IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 1 May 2023, accepted 15 May 2023, date of publication 24 May 2023, date of current version 1 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3279501

RESEARCH ARTICLE

HyDra: Hybrid Task Mapping Application Framework for NOC-Based MPSoCs

WAQAR AMIN[1], FAWAD HUSSAIN[1], (Member, IEEE), SHERAZ ANJUM[2], SHAROON SALEEM[1], WAQAR AHMAD[1], AND MUBASHIR HUSSAIN[3]

[1]Department of Computer Engineering, University of Engineering and Technology at Taxila, Taxila 47050, Pakistan
[2]Department of Computer Science, COMSATS University Islamabad, Wah Campus, Wah Cantt 47040, Pakistan
[3]School of Information Technology, King's Own Institute, Sydney, NSW 2000, Australia

Corresponding author: Waqar Amin (waqar.ameen45@gmail.com)

ABSTRACT Multiprocessor System-On-Chip (MPSoCs) with Networks-on-Chip (NoCs) has been proposed to address the communication challenges in modern dynamic applications. One of the key aspects of design exploration in NoC-based MPSoC is application mapping, which is critical for the parallel execution of multiple applications. However, mapping for dynamic workloads becomes challenging due to the unpredictable arrival times of applications and the availability of resources. In this work, we propose a hybrid task mapping approach, HyDra, that combines design-time mapping and efficient runtime remapping to reduce communication and energy costs. The proposed approach generates multiple application mappings during the design phase by minimizing latency, energy, and communication costs. The diverse mapping possibilities produced at design time consider multiple performance metrics. However, we cannot predict the arrival time of applications and the availability of resources at design time. To further optimize the MPSoC performance, our dynamic mapping phase re-configures the design time mappings based on the runtime availability of resources and applications. The simulation results show that HyDra reduces communication costs by 14% while using 15% less energy for small and large NoCs compared to state-of-the-art task mapping techniques. Furthermore, our approach provides an average of 19% reduction in end-to-end latency for applications. Our hybrid task allocation and scheduling approach effectively addresses communication issues in NoC-based MPSoCs for dynamic workloads. HyDra achieves improved performance by combining design-time and runtime mapping, providing a promising solution for future MPSoC design.

INDEX TERMS Hybrid application mapping, multiprocessors, network-on-chip, particle swarm optimization, simulated annealing, task graph for free, directed acyclic graph, dynamic task mapping, design-time mapping, K-means, elbow method.

I. INTRODUCTION

Multicore embedded systems are now conceivable, thanks to the advancements in VLSI technology that allow designers to combine a large number of Processing Elements (PEs), Intellectual Property (IP) PEs, and memory units (MUs) onto a single chip. Increased parallelism in these systems requires a fast communication framework to meet the communication needs between the PEs. The NoC architecture offers an efficient and scalable communication infrastructure across several processor PEs [1]. Multimedia, automotive, cloud computing, and avionic applications are the few sectors where NoC-based multicore systems are used [2]. The allocation of application executable tasks to available resources is critical because it influences system performance.

Important factors in the mapping process include the communication limitations between tasks and the physical placement of the PEs on a specific multicore system. The validity of outcomes in real-time applications is determined not only by logical accuracy but also by the time it takes to receive

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

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

the output. As a result, it is critical to keep in mind their time limits when executing the tasks, because most embedded systems are battery-powered and have short operating life [3].

The position of tasks and their communication with neighboring PEs have an impact on the amount of energy used during data packet exchange on an NoC-based architecture. The situation becomes more complex when multiple applications are executed simultaneously to improve resource utilization on multicore platforms.

On the provided NoC platform, running real-world applications have variable workloads. These variable workloads are impossible to forecast precisely throughout the design phase. The offline task mapping approaches provide sub-optimal results due to changes in application workloads during the runtime execution. For embedded applications, task allocation algorithms that can adapt to changing runtime environments have become crucial.

To achieve optimal mapping for the applications, the majority of design-time application mapping approaches use heuristic techniques. However, these methods are resource intensive and cannot cope with the runtime variable behavior of the applications, where some applications may operate concurrently over time. Therefore, for embedded applications, dynamic task allocation approaches (also known as runtime task allocation) are required, which can handle the variable workloads. Task distribution for new applications during the runtime has been proposed using several efficient online techniques [4], [5] [6]. However, optimal mapping may not always be guaranteed by these algorithms because of time constraints and limited computing resources at runtime.

In recent years, hybrid task allocation techniques have grown in popularity as a way to address the inadequacies of design time and runtime mapping techniques. In this type of technique, the mapping solution found at design time is used during the runtime [7] [8], [9] [10]. However, the mapping under process may fail to achieve the expected performance for a variety of unanticipated reasons during execution. For example, (a) an application may stop execution due to the unavailability of one or more allocated resources due to hardware failure. Moreover, (b) an application's performance needs may change over time, which is frequent with streaming multimedia applications. Also, (c) At some moment during the execution of any application, the predefined mapping may cause the creation of thermal hot spots which may entail turning off a section of the platform and stopping the application. Likewise, (d) unexpected changes in an application's input (workload) may cause the mapping in use to become insufficient to meet the application's performance demands, such as in image processing. Hybrid mapping strategies have been proposed to address the limitations of the runtime mapping approaches. To accomplish dynamic task mapping, these methods use pre-computed results produced offline.

In this paper, we present an enhanced hybrid (design time and runtime) task mapping and a re-configurable technique for NoC to achieve better task mapping for real-time applications. Multiple optimal mappings were generated with the best values of latency and energy against different applications. Instead of storing complete mappings for applications with different sizes and workloads, only the top task is stored with the associated information e.g. task id, average communication weight, and the number of neighbors, etc is stored.

The initial allocation information is then used at runtime to generate an appropriate allocation configuration for arriving applications depending on the existing condition of the system. To do this, we provide an enhanced runtime method for dynamically selecting and changing design-time allocation options while fulfilling task deadlines and lowering communication energy usage. The proposed approach takes advantage of the pre-computed task mapping options to offer effective decisions for runtime resource allocation.

The following is a list of the contributions of this research work.

1) The proposed approach offers an efficient design-time technique that has been evaluated for its performance on both real-world benchmark applications and synthetic applications of varying sizes. The proposed design-time approach achieved optimal results in communication cost, latency, and energy consumption for all test cases.
2) The proposed HAM technique reduces the workload of finding the most suitable mapping for incoming applications at runtime by utilizing a variety of pre-calculated mappings that were generated during the design phase. This minimizes the overhead associated with mapping multiple applications of varying sizes.
3) The proposed technique is tested on various scenarios to evaluate its efficiency both at design time and runtime. Its design time performance was assessed in terms of its ability to reach an optimal cost, while its runtime mapping capability was evaluated based on different runtime workloads and test scenarios.
4) The comparison between the proposed hybrid application mapping approach and current dynamic application mapping algorithms reveals that the proposed HAM algorithm performs better in terms of energy consumption, latency, and communication cost.

II. RELATED WORK

Dynamic mapping and design time mapping are two methodologies used to assign and sort tasks according to the available time. The dynamic mapping works while applications are in use, attempting to detect any performance issues and spread the workload evenly. This can lead to a more optimal solution but also can slow down the execution of the application and require more energy. Static mapping, meanwhile, is completed prior to the application's execution, aiming to determine the best task placement for a particular application and communication infrastructure [22], [23].

TABLE 1. Summary of application mapping approaches for NoC.

Proposed Technique	Classification	Mapping approach	Advantages	Disadvantages
[11]	Dynamic Mapping	Dynamic mapping technique with three stages that operate at runtime and is based on a single-move remapping	Examines the benefits and costs of potential migrations	Comparison of migration cost and performance enhancement
[12]	Dynamic Mapping	Task mapping for NoC platforms using Virtual Regions	Improves Latency and communication cost and computation time for small applications	Computation time may increase for large applications
[6]	Dynamic Mapping	Highly communicative activities are mapped to similar or nearby PEs. in the following order: left, down, top, and right	Reduced communication overhead and Energy are achieved	Some leaf tasks could fill up some PEs. As a result, such PEs are underutilized since they are not shared with any other processes
[13]	Dynamic Mapping	Real-time mapping is done online together with analysis of the communication status of applications	Communication energy saving	Lightly communicative tasks could be ignored or put distant
[5]	Dynamic Mapping	The impact of user behavior improves how well the system reacts to real-time changes	Reduced Communication energy	Increased Computational overhead
[14]	Dynamic Mapping	A more reliable reaction to real-time changes is achieved by taking into account user behavior while allocating resources	Communication energy savings	Increased computation requirements
[15]	Dynamic Mapping	Tasks are mapped out in a spiral pattern	Reduced reconfiguration time	During spiral mapping, certain PEs that may give better mapping may be overlooked
[16]	Dynamic Mapping	Agents can manage mapping and resources, store resource status information, and engage in inter-agent negotiations	Minimize the volume of traffic required to assess the network's status	Computational and communication overhead
[17] [18]	Dynamic Mapping	At runtime, the tasks are mapped while considering the communication requests and link load	Reduced execution times and congestion, smaller packet latency	This heuristic applies only to single-task processors
[19]	Static Mapping	Minimize inter-tile network contention by using integer linear programming (ILP)	Decrease in packet latency	High computation time and somewhat higher communication energy overhead
[20]	Static Mapping	ILP-based application mapping	Reduced energy consumption	The approach has a high computation time and does not take into account bandwidth restrictions
[21]	Static Mapping	Cluster-based hybrid application mapping	Produces optimal results	The trade-off between better performance and computation speed

Design time mapping techniques employ various offline heuristics and non-heuristics techniques to find out mapping with the best results. The optimal mapping is considered which offers the lowest communication cost, latency, and energy. In the runtime application mapping, during the execution state, the incoming applications are mapped either by using a pre-calculated mapping during the design time, or tasks are mapped one by one by identifying the best placement for mapping at runtime. The application will run on the specified mapping until it is terminated once it is launched.

An effective mapping procedure is needed to assign tasks to the appropriate PEs to make the most use of the NoC resources. In NoC, the process of assigning tasks from an application to PEs while bearing in mind the optimization criteria, such as a decrease in energy usage, overall execution time, congestion, and communication overhead is referred to as task mapping [24]. Decisions on task mapping have a direct impact on how well the system performs overall in terms of the aforementioned metrics. It is NP-hard to map applications onto the NoC architecture [25], [26].

As a result, several heuristics-based solutions to task mapping issues have been proposed. To discover an effective task mapping for NoC, a lot of effort has been made and is currently being performed. We briefly discuss some task

mapping strategies proposed for NoC in the following paragraphs. In addition, Table 1 provides a quick comparison of the methodologies covered.

Mapping multiple applications onto NoC [6] presented a communication-aware runtime heuristic. The model looks at the resources that exist and assigns the communicating tasks to the same or closest PE in the left, down, top, and right order. This method has a significant drawback that it can map leaf tasks on such PEs that are not being used by any other tasks, which leads to the ineffective use of some PEs and leads to higher energy consumption.

An energy-aware runtime mapping technique for NoC was presented in [13] The algorithm's goal was to reduce the energy used in communication. The first phase of the algorithm is to examine how applications are communicating while they are running and choose the task with the maximum communication volume. The chosen task is first mapped. The subsequent phases involve sorting and mapping the nearby tasks to the closest PEs based on how much communication they get from the selected activity. The mapping carried out by this algorithm places activities that communicate often near to one another, whilst tasks that communicate seldom may be ignored and placed far away.

A runtime task allocation mechanism was proposed in [5] while taking user behavior into account. The system can respond to real-time changes more quickly thanks to the effect of user behavior. Additionally, it enables the system's dynamic adaptation to variable user requirements. The series of actions that a user takes when interacting with the system shapes their behavior. To process the user behavior, the algorithm has to apply machine learning techniques, which adds additional computational overhead.

A user-dependent dynamic mapping approach was presented in [14]. Resource allocation specifically takes into account user behavior, enabling better responsiveness to real-time changes. When a user interacts with a system or network, their behavior may be characterized as a series of related events that overlap and follow one another. Since user behavior might differ from one user to the next, the system must be clever enough to handle user behavior that necessitates additional processing demands.

A dynamic spiral mapping approach was proposed [15]. From the center out to the edge of NoC architecture, the tasks are mapped in a spiral manner. The communicative tasks are positioned nearer to one another. Additionally, the dynamic mapping and task migration times are cut down, which reduced communication time. Even though the tasks are spirally mapped onto the NoC architecture, it is possible that the PEs that may offer a better mapping were overlooked during the spiral mapping.

In [16] introduced a runtime agent-based mapping approach. A small task called an agent is one that may run on any NoC node. Agents are capable of managing resources, storing resource status information, and negotiating with one another to select the most appropriate processing element for a given task. There are two categories of agents: global agents (GA) and cluster agents (CA). When given a task, cluster agents first analyze their clusters before negotiating with global agents who have access to global knowledge about all clusters. The negotiation aid in producing effective mappings. While the computational burden is raised due to agent interactions, the proposed method reduced the overall bandwidth required to ascertain the network's present status.

Authors in [17] ana [18] developed dynamic task mapping heuristics for heterogeneous MPSoCs based on NoCs. The performance of the mapping heuristic is observed for dynamic workloads. A dynamic mapping phase follows the initial task mapping phase. The methodology that is being described aims to reduce congestion while improving NoC performance. At runtime, the tasks are mapped while taking the communication requests and link load into account. This heuristic only works with processors that can handle a single task. It may also be used to multi-task processors.

For heterogeneous NoC architectures [27] introduced an energy-aware mapping method to reduce energy usage during task allocation and scheduling. To create a global optimum mapping, an extension of integer linear programming (ILP) was devised that takes both processing and communication energy into account. However, Its execution takes a long time. The authors also presented a Simulated Annealing with Timing Adjustment (SA-TA) strategy that aimed to speed up the optimization procedure that was extremely near to the global optimum to get around the limitations of the ILP-based approach. The presented model's primary flaw is that it only takes into account a simplified energy model without taking leakage power into account.

A contention-aware application mapping was suggested in [19]. To reduce inter-tile network contention, this method uses integer linear programming (ILP). Though the communication energy overhead is marginally raised, the proposed scheme reduces packet delay. This method was only applicable to 2-D mesh NoCs with XY routing. ILP was also proposed as part of an application mapping approach [20]. Although the approach reduces energy usage for several benchmarks, bandwidth restrictions are not taken into account. The algorithm's computing time is quite high.

A quantitative comparison of the advantages and disadvantages of static and dynamic mapping techniques was done in [11]. The study's findings indicate that because dynamic algorithms can cope with real-time scenarios, they are preferable to static ones. Dynamic algorithms, on the other hand, only take into account a portion of the application graph because they only address the problem of communicating with their caller. Static algorithms, on the other hand, can utilize sophisticated methods to take into account all tasks and resources and have a broad perspective of the entire system.

Another comparison analysis of runtime mapping techniques for heterogeneous NoC was published in [28]. In this study, eight alternative heuristics were evaluated, although the comparison was based on a relatively small number of factors and the deadlines were also missed throughout the

simulation. To produce quality results, it is necessary to concentrate on the deadlines as well.

In [29], the authors offer a move-based approach for dynamic mapping on heterogeneous NoC that targets data flow actors. Using worst-case timing and schedule updating with actual execution time at runtime. A distributed runtime resource management framework for multicore systems is provided in [12] by allocating distinct responsibilities to the available computer resources, the method combined the concepts of distributed management with parallel applications. The architecture given here is based on the concept of local controllers and managers, with decision distribution ensured via an on-chip intercommunication.

In [30], a novel technique for task mapping termed Virtual Regions PBIL (VRPBIL-NoC) is introduced, which is based on the PBIL algorithm. This method divides the platform into virtual regions to support the search for high-quality solutions. To achieve reconfiguration predictability, the proposed approach in [31] uses a design-time reconfiguration analysis process to discover efficient reconfiguration solutions among the mappings and determine the worst-case reconfiguration latency. The RM receives the design-time analysis as well as the migration routes to conduct predictable mapping re-configurations at runtime.

A new HAM approach was presented in [32] which combines elements of a scenario-based design during the design phase, and a three-step data-driven selection process during the runtime phase, to optimize soft real-time applications. The proposed algorithm in [33] for dynamic task allocation and scheduling with contention awareness for NoC-based multicore systems provides a novel approach for improving the performance of real-time applications. The algorithm aims to select an appropriate processor for task execution based on the link utilization of the target multicore platform. It also dynamically selects a route in order to mitigate network contention during the task execution.

In [34], a mapping strategy was proposed that explores power patterns and their footprints. Three metrics (power peak, power range, and regional power density) were defined and tested when used as mapping objectives along with communication costs. [35] introduces a hierarchical and dependency-aware (HDA) task mapping that considers both spatial mapping and the dependencies between tasks to increase the flexibility of the task mapping. Application mapping framework which consists of three mapping algorithms, Horological Mapping (HorMAP), Rotational Mapping (RtMAP) and Divide and Conquer Mapping (DACMAP) proposed in [36]. A new algorithm called Adaptive Core Mapping (ACM) [37] has been tested and simulated using Vivado Design Suite 2018.3 and validated on a Kintex-7 FPGA board. Its performance parameters including area, power consumption, and throughput were analyzed to show how this algorithm can help systems recover from failed PEs and maintain optimal functionality.

TABLE 2. Parameters used in the evolution model and the proposed algorithm.

Notations	Explanation	Units
L_{av}	Avg latency of the network	cycles
$L_{i,j}$	Packet j latency in processor i	cycles
P_{avg}	The average power of the network	W
$P_{act,j}$	Power of an active component j	W
$P_{inact,j}$	Power of an inactive component j	W
$\alpha_{i,j}$	Percentage of active components	%
E_{avg}	Average energy per packet	pJ/packet
T_{sim}	Simulation Time	seconds
T_{wrm}	Warm-up time	seconds
T_{pav}	Average throughput	packets/cycle
N	Total processor in the platform	-
N_i	Total packets process by i processor	-
N_p	The total packets transmitted over the network. $N_p = \sum_{i=1}^{N} N_i$	-
B_{t_i,t_j}	Bandwidth constraint between task i and j	MB/s
N_h	Manhattan distance	hops
S_t	Suspension time of the task	seconds
C_{st}	Core search time for migration	seconds
M_{gt}	Migration time	seconds

III. PROBLEM FORMULATION

In this section, the task mapping problem on NoC platforms is described, along with a set of considerations and prerequisites for establishing a task mapping approach. We proposed some metrics that the mapping approach should take into account, as well as the conceptual domain for modeling applications.

A. PERFORMANCE EVALUATION MODEL

The performance evaluation for 2D NoC includes latency, communication cost, power, and energy models. The proposed algorithm is evaluated using the mathematical models discussed in this section. The standard notations used during modeling are described in Table 2.

We have implemented the energy model described in [38]. The power and energy model uses data from a 65 nm CMOS-based system. The average energy consumption relates to the energy consumed per packet by different NoC components, such as switches, internal buffers, connectivity wires inside the fabrication, and connections, both active and idle.

Power and energy calculations are constructed by the sum of the energy consumed in the execution of tasks on the processor N and the energy consumed by the router components. Let $P_{act,j}$, and $P_{inact,j}$ be the active and inactive power of router component j. Let $\alpha_{i,j}$ be an active proportion of component j in router i (after the warm-up time), so the average power of router i is presented in Equation 1

$$P_i = \sum_{j=1}^{N} [\alpha_{i,j} P_{act,j} + (1 - \alpha_{i,j}) P_{inact,j}] \tag{1}$$

The average Power P_{avg} of the network is represented in Equation 2 and 3

$$P_{avg} = \frac{1}{N} \sum_{i=1}^{N} P_i \tag{2}$$

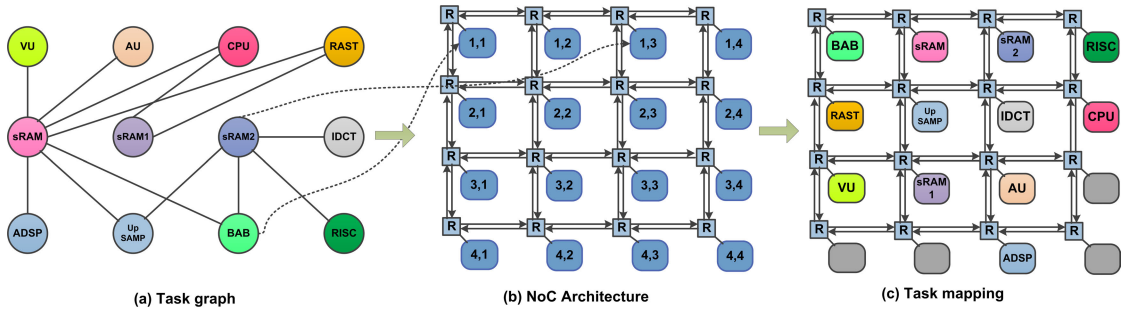


FIGURE 1. Application mapping onto NoC graphically.

$$P_{avg} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}[\alpha_{i,j}P_{act,j} + (1-\alpha_{i,j})P_{inact,j}] \quad (3)$$

Equation 4 shows the average energy consumed by each packet in the network.

$$E_{avg} = \frac{(T_{sim} - T_{wrm})}{(NN_p)} \times \sum_{i=1}^{N}\sum_{j=1}^{N}[\alpha_{i,j}P_{act,j} + (1-\alpha_{i,j})P_{inact,j}] \quad (4)$$

The communication cost system is specified in Equation 5.

$$Cost = \sum_{i,j}^{N}[B_{t_i,t_j} \times N_h] \quad (5)$$

where, B_{t_i,t_j} represent the bandwidth between tile t_i and t_j and N_h represents the Manhattan distance. The Manhattan distance between the source node (x_i, y_i) and the destination node (x_j, y_j) in the NoC architecture is indicated by

$$N_h = |x_i - x_j| + |y_i - y_j| \quad (6)$$

Equation 7 reflects the average latency of the network:

$$L_{avg} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{N_i}\sum_{j=1}^{N}L_{i,j} \quad (7)$$

where $L_{i,j}$ denotes the delay of the *jth* packet, N denotes the number of PEs in the network, and N_i is the number of packets processed by one processor after a warm-up period.
Reconfiguration time is the most important parameter in the dynamic mapping technique because slow reconfiguration time can affect the overall performance of time-critical applications. It is crucial to do a thorough analysis of the worst-case timings of reconfiguration to make sure that the reconfiguration does not disrupt the real-time requirements of the application. The reconfiguration time is summarized in Equation 8.

$$R_t = S_t + C_{st} + M_{gt} \quad (8)$$

where S_t = suspension time of the task; C_{st} = Core search time for migration; and M_{gt} = Migration time

Suspension time S_t measures how long it takes to stop all tasks from running and transport the messages they create to the target PEs, ensuring that no application messages are currently being sent. Core search time is the time required to search for a suitable core for reconfiguration. Migration time is the time to migrate the task from the current core to the next identified core.

B. WORKING APPLICATION MODEL

To address the mapping issue of NoC architecture, we use Directed Acyclic Graph (DAG) = G(T,E,W) to represent a real-time application as shown in Figure 1. Where $T = \{t_1, t_2, t_3, \hat{a}, \ldots.., t_n\}$ denotes a set of tasks and $E \subseteq C \times C$ is a set of directed edges. While each edge $(c_i, c_j) \in E$ describes the data flow and connectivity between two tasks in MB/s. For instance, if task c_i and task c_j are connected by an edge, c_i is the antecedent of c_j and sends data to c_j, while c_j is the descendant of c_i and receives data from c_i. B_{t_i,t_j} is the edge-weight of the edges that displays the amount of data (in bits) transmitted from c_i to c_j, we presume that D_n is the common deadline shared by all application tasks.

C. COMMUNICATION MODEL

We assume a 2D-mesh topology with homogeneous PEs connected through NoC as a communication architecture for the proposed technique. The routing channel $(R_{i,j})$ is in charge of creating a physical path for packets to go from the source core to the destination core. Each router contains five ports, of which four are used to interact with nearby routers and one is reserved specifically for processor communication. A link is used to connect two routers or any router to the adjacent processor. We assume that all links are full duplex, identical. The data bandwidth requirements (B_{t_i,t_j}) are linked to the routing channel. We consider the Wormhole switching technique (WHST) for the proposed technique. Figure 1 depicts the problem design for application mapping onto NoC graphically.

The 2D-mesh topology, with its regular connections, scalability, and direct communication, in conjunction with Wormhole switching, provides a simple and efficient architecture for NoC systems. For routing protocol, our choice is the well-known XY deterministic routing method. The XY routing is a simplistic yet efficient method. Its ability to avoid deadlocks is another one of its many features. These design considerations can contribute to effective application

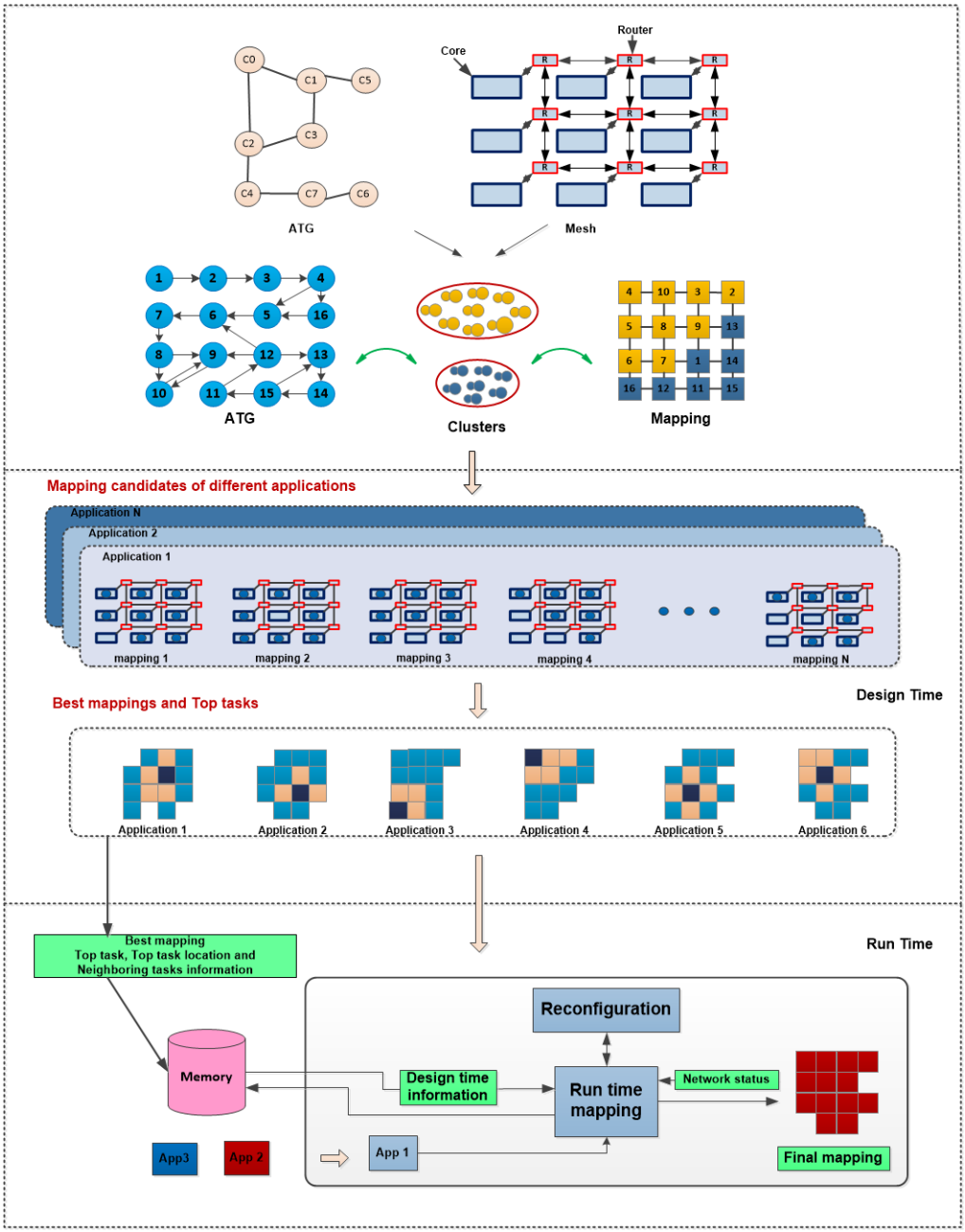


FIGURE 2. Proposed hybrid mapping framework HyDra.

mapping, reduced communication overhead, and improved overall system performance.

IV. HYDRA

This section presents the proposed hybrid application mapping and reconfiguration framework HyDra, which included dynamic mapping and reconfiguration DRA and design-time mapping approaches for the NoC-based multicore platform. The design-time mapping approach iHPSA [21] generates mappings with optimal communication costs having minimum latency and energy consumption. The design time stage produces different mappings that can have optimal communication costs. However, these mappings can have different placements of tasks on the NoC, making different shapes of mapped regions. Detailed analysis was carried out on the optimal mappings to identify the best possible shapes against the different real-world benchmark applications.

From the analysis we found out that the area with the greatest number of closely related activities is practically circular, making it the most contiguous area. Close mapping of an application's tasks will result in minimal expected costs and latency. However, external congestion in a mesh network is increased when a circular region is used for mapping an

application. Because nearby (circular) areas exchange network links, this happens in a mesh topology. As a substitute, when tasks are organized into a rectangular area of a network with limited routing, most traffic will be routed within that area, preventing external congestion. The square was found to be the best region for optimum mapping based on the evaluation. For any given application the task having the maximum communication weight with the neighboring tasks is mapped in the square. In this article, we consider the task with maximum communication weight as the top task τ_p.

The proposed hybrid scheme identifies the top task τ_p in all mappings generated by the design time approach. The top task has the maximum average communication weight. Instead of saving the complete mappings against different applications, only the information related to the top task τ_p such as the total number of tasks in the application κ_n, top task location τ_{loc}, the number of neighboring tasks in the square τ_{nbr}, and their location on the NoC is stored and further used during the runtime phase.

At the runtime mapping stage upon arrival of the application, the mapping information such as the top task τ_p and its mapping location on NoC is retrieved by identifying the application size. The top task τ_p is mapped on the location identified during the design time followed by the neighboring tasks on the adjacent PEs forming the square area around the first task. The tasks with the lowest weights are mapped at a minimum distance from the main region.

runtime contention status is monitored during the complete execution of the application and if the current mapping is causing congestion at any location, reconfiguration is done to find the more suitable mapping and to avoid congestion. The proposed hybrid approach is described in detail in the sections to follow. Figure.2 shows the hybrid mapping framework for design time and runtime application mapping and reconfiguration.

A. DESIGN-TIME MAPPING FRAMEWORK

We used our earlier work iHPSA [21] an improved hybrid Particle Swarm and Simulated the Annealing based application mapping technique for 2D NoC as a design-time mapping framework. In the iHPSA, the PSO with strong global search capabilities and the SA algorithm with strong local search capabilities are combined. The combination of PSO and SA helps the proposed method escape from the local optimum and maintain quick convergence for the vast majority of the time. It compensates for the shortcomings of each methodology when used in conjunction with the K-means clustering technique and an efficient first-stage mapping strategy. The fitness function which in this case is communication cost is the main function in a problem that we aim to reduce using the proposed technique. It assists in determining if the environment is suitable for each solution being considered. If the global best solution has remained the same for the past K iterations, the algorithm is likely trapped at the best local value. By applying SA over PSO, the proposed methodology breaks out of a local minimum in a relatively short amount of time. But if SA is used with PSO at every iteration, the execution time will rise and the rapid convergence capability of PSO will decrease at the same time. SA has applied to PSO in every K iteration till the optimal result is obtained to effectively integrate SA with PSO. Because of the PSO and SA, the proposed technique can often maintain quick convergence and escape from the local optimum. SA is added to the best possible result in the swarm decoded thus far in order to allow PSO to hop out of the local optimum.

The global and local searches were balanced by the addition of the inertia weight ω. The inertia weight ω is used to regulate the swarm's ability to explore and exploit, eliminating the need to use velocity clamping. The inertia weight ω adjusts the momentum of the particle, deciding how much the prior flight direction will impact the new velocity by considering the contribution of the old speed. The inertia weight ω shows how the new velocity will be influenced by the amount of recollection from the previous flight direction. When the inertia weight ω is greater than 1, the velocity will eventually decrease, allowing the particle to reach its full speed, and consequently making the swarm spread out. Conversely, for values less than 1, the particle's velocity will reduce until it reaches zero. A higher value of ω will make exploration easier and a lower value will promote exploitation [39]. The definition of the particle swarm process equations is

$$\beta = random(0, 1) \tag{9}$$

$$\gamma = 1 - \omega - \beta \tag{10}$$

$$v_{(i+1)} = \sum \omega\, v_i * \beta * (pbest_i - x_i) * \gamma * (gbest_i - x_i) \tag{11}$$

$$x_{i+1} = x_i + v_{i+1} \tag{12}$$

where random (0, 1) is an integer produced at random within the range [0, 1]. The social and cognitive component effects are governed by the learning factors β and γ As shown in equation 9 to 12, the values of β and γ are determined at each iteration. The proposed algorithm's stop criteria are defined as the moment when all evaluation functions are exhausted or the desired result is not achieved. Following thorough assessment and comparison, the algorithm's input parameters are chosen. Additionally, the choice of parameters depends on the nature of the problem; one set of parameters may be effective for one problem but not another.

1) POPULATION GENERATION

The PSO method begins with a collection of swarm particles and iteratively improves to achieve the optimal solution. In our case, a particle consists of the number of tasks of the application and their initial communication cost. For example, the particle for an application with n tasks is defined as:

$$\delta_i = [X, t_1, t_2, t_3, \ldots.., T_n] \tag{13}$$

where δ_i represents a particle consisting of a number of tasks of an *ith* application and X is the initial communication cost

of the particle. The complete generation consists of G x δ_i, where G is the generation size The particles are updated every generation using Equations 9-12 till the best value is achieved.

2) FITNESS FUNCTION

For efficient 2D mapping in this work, communication cost and power-based optimization have been implemented. Furthermore, the fitness function is designed so that none of its components will predominate during the optimization process [40]. It is possible to formulate the fitness function F for minimization.

$$F = (P_{avg})^{\eta P}(En_p)^{\eta E} \tag{14}$$

where ηP and ηE stand for the restraining elements of power and energy. By setting the appropriate control factor to zero, any fitness function's efficacy may be eliminated.

3) PARAMETER SETTINGS DESIGN TIME

To get an effective solution, the parameter values are chosen following several simulations and positive outcomes. It is important to note that the values of these parameters need not be the same for all applications.

The optimal parameter values are determined based on the specific requirements and characteristics of each application. Small values of ω force PSO to converge quickly, leaving PSO trapped at local maxima; in contrast, big values extend the exploration area and can cause PSO to take too long to converge on a global optimum.

A modified version of the PSO algorithm uses adaptable values of β and γ that are modified after every iteration. Different values of parameters, including population size and iterations, are used for different applications. The value of inertia weight ω linearly decreased from 0.9 to 0.4.

In the SA algorithm, the factor $\lambda = 0.99$ represents the decrease in temperature. The process of temperature reduction is shown as $T_{i+1} = \lambda T_i$, where $i = 0, 1..$ and $T = 10ln|P|$. The temperature is reduced by a factor of $\lambda = 0.99$ in the SA method. The approach produces satisfactory results, per earlier research [41], when the initial temperature is set at $10ln|P|$, where P is the total number of PEs in the network. For applications with few PEs, the SA algorithm's inner loop or iteration numbers are set to $|P|^2$, while for applications with large PEs, they are set to $|P|$. where $|P|$ is the mesh's number of tiles.

4) COMPLEXITY ANALYSIS

The complexity of the design time algorithm is approximately $O(itrPSO * popsize * networksize^2)$. Where the initialization step has a complexity of $O(networksize^2)$ since it involves creating and evaluating the initial population. The inner for loop from 1 to *itrPSO* has a complexity of $O(itrPSO{*}popsize)$ since it involves iterating over the population and updating the velocity of each particle. The second inner for loop from 1 to popsize has a complexity of $O(popsize * networksize^2)$ since it involves evaluating the communication cost for each particle in the population. The global best fitness and solution are updated after each iteration, which has a complexity of $O(1)$. If the global best fitness is equal to the optimal communication cost, then the final mapping is set to the global best solution, which has a complexity of $O(networksize^2)$. The overall complexity of the runtime mapping algorithms is $O(\kappa_n * (E + VZ_n))$, where n is the number of tasks in the application, E is the number of edges in the task graph, and VZ_n is the size of the virtual zone.

5) OPTIMAL COMMUNICATION COST CONVERGENCE ANALYSIS

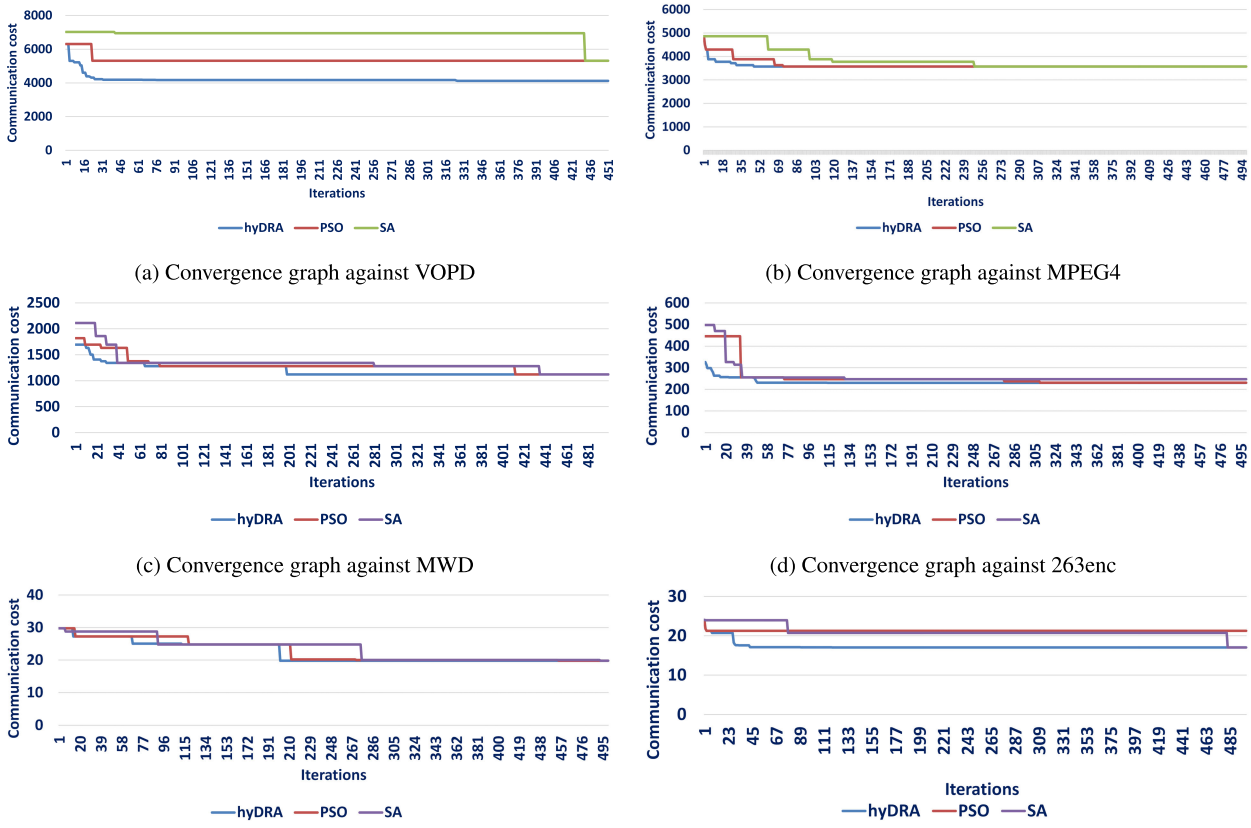
The combination of PSO and SA into a hybrid algorithm was found to be more successful in quickly reaching the optimal cost of communication. This was reflected by achieving the optimal result in fewer iterations than would have been possible with either algorithm alone, with some cases having undesired outcomes when the algorithms were used separately. We executed the PSO, SA, and proposed design time hybrid algorithm against each real-world benchmark application.

Figure 3 shows the convergence to optimal communication cost of the proposed hybrid algorithm and PSO, SA algorithms against six real-world benchmarks. From figure 3, we can observe that the proposed hybrid algorithm converges to optimal communication cost much faster by improving the current result in a few iterations than the PSO and SA algorithms. In some cases PSO and SA if executed separately converged faster to an undesired value. The hybrid configuration of PSO and SA improved the convergence capability of the proposed algorithm to achieve the optimal communication cost in less number of iterations.

6) ANALYSIS OF MAPPING SHAPES AND TOP TASK SELECTION

In this part, we will discuss the analysis of optimal mapping shapes for applications of different sizes to find out the top task selection. This analysis will be used in the runtime mapping when multiple applications will be mapped on the same network. The design time mapping approach produced optimal mappings for both real-world benchmarks and large synthetic applications. There are multiple mapping results with optimal communication cost but with a different set of task placements on NoC. The different task placement on the NoC creates different shapes containing the area of mapped PEs of the NoC.

Figure.4 shows different mappings with the same communication cost value but having different task placements on the NoC. The different placement of tasks on NoC produces variations in the overall latency and energy of the network. For small applications the variation in the latency and energy is negligible but for large applications and multiple applications on the same NoC can increase the latency and energy.



(a) Convergence graph against VOPD

(b) Convergence graph against MPEG4

(c) Convergence graph against MWD

(d) Convergence graph against 263enc

(e) Convergence graph against 263dec

(f) Convergence graph against mp3enc

FIGURE 3. Optimal communication cost convergence analysis.

The energy and execution times of four final mappings are also given in Figure.4. We termed these different task groupings on NoC as mapping shapes. We performed an offline comparison and analysis of execution time, latency, and energy to select the best mapping shapes for different real-world benchmarks. In Figure.4, we used 4×4 2D mesh NoC architecture for mapping 12 x tasks of MPEG4 real-world benchmark application. If the number of tasks of the application is less than the network size, as in the case of MPEG4 where we have 12 tasks to be mapped on a 4×4 network. Then the mapping shapes will include only mapped PEs and leave unmapped PEs. The mapping shape is further analyzed to find out the position of the Top task and the closely related neighboring tasks forming a square area inside the mapping shape are identified.

It is very important to find out efficient mapping shapes during the design time because the selected mapping shape will be used as initial mapping during the runtime mapping. The mapping calculated at the design time may fail to achieve the expected performance for a variety of unanticipated reasons during execution. However, detailed analysis during the design time and a list of different shapes help in the selection of another possible mapping if there is some issue during the runtime. Algorithm 1, presents the design time mapping process.

B. DYNAMIC MAPPING AND RECONFIGURATION APPLICATION MAPPING (DRA)

We employ the design-time results to create a lightweight task assignment mechanism for runtime. It involves identifying the best combination of PEs for mapping an application's tasks and executing them at runtime. The runtime situations cannot be predicted since the arrival time of applications is unknown in advance. The real-time task assignment is affected by the following factors: (a) the availability of PEs for the execution of new applications, and (b) execution time. Design-time decisions may not always be acceptable for runtime operations due to these reasons. As a result, when an application is executed during runtime, the task assignment of the application must be modified.

Virtual zones are created within the NoC during the runtime mapping. The size of the NoC, as well as the size and quantity of applications, all influence the number of virtual zones. The multicore platform application being mapped uses less energy for communication thanks to the mapping on virtual zones. A suitable mapping template may be chosen from the repository to do this. The search process is repeated until a region size that will fit in the set of PEs that are currently accessible on the multicore platform at runtime is discovered. The first step is to examine the areas whose size corresponds to the number of PEs on the platform. The simultaneous

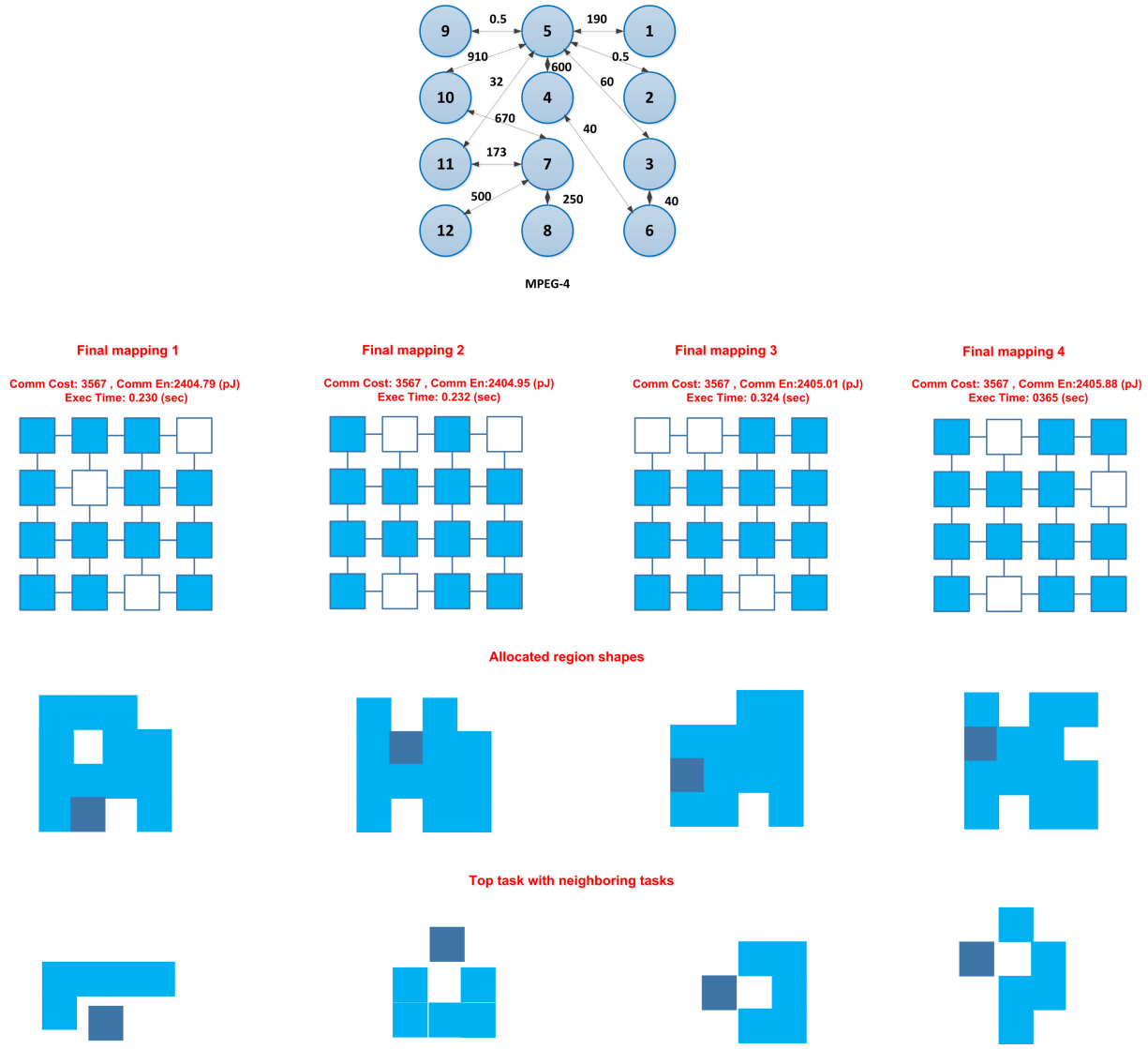


FIGURE 4. Process of mapping shapes and top task selection.

execution of tasks contained in a particular application is impacted by the platform's free PE availability. Due to the availability of PEs, larger allocation regions provide a higher possibility for various activities to be executed concurrently. This will reduce the time to find out the best mapping placements during runtime as we already know that this mapping has the best results during the design time. If the number of tasks is more than the available PEs in the free virtual zone, the free PEs of other virtual zones can be shared for mapping.

Algorithm 2 presents a Dynamic Mapping and Reconfiguration (DRA) technique for runtime mapping based on design-time findings and runtime resource availability. At the start, the number of tasks κ_n of the incoming application is used to fetch the design time information such as τ_{loc} the best mapping location of the τ_p of the application having the same size and the number of neighboring tasks τ_{nbr} of the τ_p in the square. The τ_p which has a high communication bandwidth with the neighboring tasks τ_{nbr} is selected as the first task to map. The virtual zone VZ_n is selected to map the application and the location of the core to map the first task is set as the top task location τ_{loc}. τ_p is placed on the core selected and the rest of the tasks are mapped exactly on the tiles as per the design time information. After the mapping of all tasks, the end-to-end latency and communication cost of the mapped network is calculated and compared with the predefined latency threshold C_{th} and communication cost threshold C_{th}. The reconfiguration process is invoked if the values of C_{th} and C_{th} are above the threshold.

C. WORKING EXAMPLE

We selected 8 × 8 mesh NoC for the implementation of the proposed dynamic approach. The 8×8 mesh is further divided into four virtual zones of 4×4 size each. The first application will be mapped on virtual zone 1, the second application on the second virtual zone, and so on. All applications will

Algorithm 1 Design time mapping process

```
Data: initMapping, taskGraph,popSize, itrPSO,
      itrSA, T, ω, itrCompleted, optimalCommCost
Result: optimalcosts, optimalmappings, topTaskLoc,
        commDependentedTasks

Find pop, v, fitness
initpopfitness(initMapping, popsize, networksize);
Find gbestpop, gbestfitness, pbestpop, pbestfitness
getinitbest(fitness, pop);
while stopCriterion do
    for i  1 to itrPSO do
        for j ← 1 to popsize do
            β = random(0, 1);
            γ = 1 − ω − β
            Find v using 11;
            pop[j] = add(pop[j], v[j])
        end
        for j ← 1 to popsize do
            calCommCost(networksize, pop[ j],
              taskGraph);
        end
        gbestfitness = min(pbestfitness);
        gbestpop = min(pop);
    end
    if gbestfitness == optimalCommCost then
        finalMapping ← gbestpop ;
        optimalCommCost ← gbestfitness ;
    else
        Apply SA to global best solution
        itrSA ← 0 ;
        T ← 10 × ln[P] ;
        currentbest ← gbestpop ;
        currentbestcost ← gbestfitness ;
    end
    while itrSA do
        pbestSA create(currentbest);

        Find best cost SA
        calCommCost(networksize, pbestSA,
          taskGraph);
        itrSA = itrSA+1;
        update(gbestfitness, finalMapping);
        optimalcost= gbestfitness;
        optimalmapping = finalMapping;
        findandstoreTopTask(optimalmapping);
    end
end
```

execute independently of each other, if there is communication between applications then all communicative applications will be considered as a single application. If the mapping application has fewer or equal tasks than the available PEs in the free virtual zone then the information related to optimal mapping shapes and top tasks generated and stored during the design phase is retrieved during runtime based on the size of the applications

Figure 5 illustrates the runtime mapping process of the proposed mapping scheme, which divides the 8 × 8 NoC architecture into four virtual zones comprising of 16 PEs each. Figure 5 shows the initial state at time instance T1, when all the virtual zones are free and the first application A-1 arrives for mapping. In this case, Virtual Zone 1 (vz_1) is chosen for task allocation due to the number of tasks in A-1 being fewer than the maximum PEs available in vz_1. The top task τ_p shown in black is placed alongside the other tasks, with the position based on the design time mapping information that was obtained. At time instance T2, the second application A-2, with 13 tasks, arrives and VZ_2 is selected for the task allocation since the number of tasks in A-2 is less than the maximum PEs available for mapping in VZ_2. For applications, such as A-3, with a size larger than the available size of a single virtual zone, other zones can be used for mapping, as depicted in Figure 5 at time instance T3. The predefined mappings at design time may not be optimal for the PEs available on the platform, leading to unacceptable latency and energy results. If this occurs, the mappings must be modified at runtime, as represented by the arrows at time instance T6 in Figure 5 where task allocation is reconfigured to new positions.

V. EXPERIMENTAL SETUP

The proposed algorithm's performance is evaluated in an experimental setting. The experimental framework and results are described in the following subsections. To execute application mapping on NoC, we developed a hybrid simulation framework that consists of a simulated environment developed in Python and a modified version of Noctweek [38], a systemC-based simulator. The Noctweek simulator has been modified to incorporate the proposed hybrid task mapping approach. We explore NoC with different sizes for executing the simulations using benchmark applications such as PIP, MPEG4, VOPD, MWD, MP3ENC, 263DEC, 263ENC, and MMS [42] as shown in Table 3. To construct task graphs with a large number of tasks, we employ synthetic applications produced by the TGFF tool [43]. Simulation has been carried out using an Intel i5 platform with a 2.5 GHz clock frequency and 8 GB of main memory. For the design-time mapping technique, we used a Python environment and for runtime mapping and scheduling, we modified the System C-based NoC simulator Noctweek [38]. We used the parameters given in Table 5 for the evaluation of the proposed algorithm. For avoiding unnecessary complexity we used 2D- mesh network with an XY routing algorithm. We evaluated the performance by mapping multiple applications from a minimum of 2 applications to a maximum of 6 applications on different sizes of mesh networks as given in Table 3. The applications are synthetically generated by a modified TGFF tool [43] with a variable number of tasks per application.

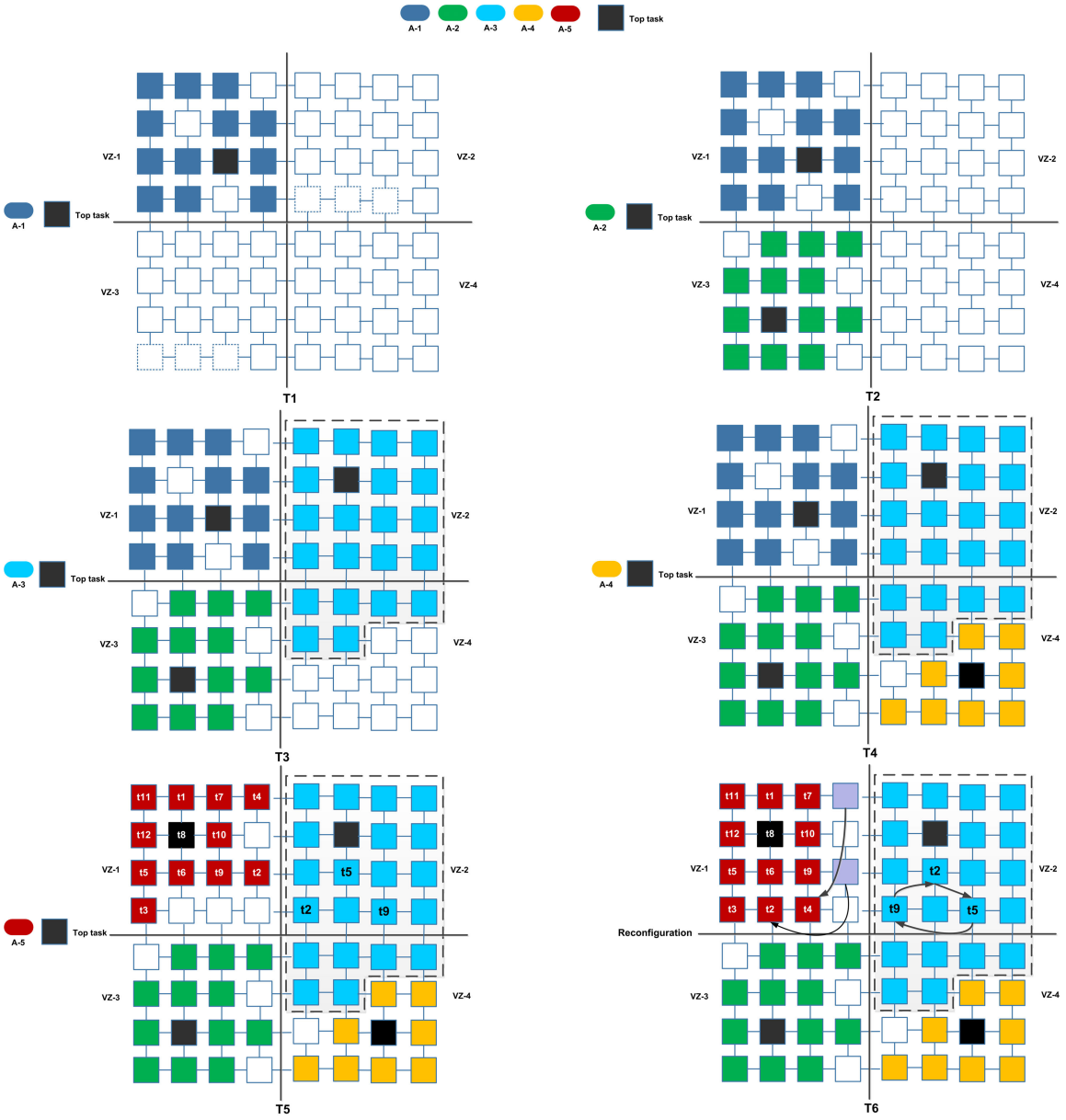


FIGURE 5. Dynamic mapping and reconfiguration process.

A. EVALUATION OF PROPOSED HYBRID TECHNIQUE

The effectiveness of the proposed hybrid application mapping technique was assessed and compared with the state-of-the-art. Different performance criteria, including communication cost, consumption of energy, latency, and channel load, were considered for both design time and runtime techniques.

B. DESIGN-TIME PERFORMANCE COMPARISON

This research uses a hybrid of particle swarm optimization (PSO) and simulated annealing (SA) algorithms to allocate tasks onto processors within a Network-on-Chip based multicore platform during the design phase. The performance of the proposed design time approach is compared with different algorithms. Table 6 shows the results of the optimal communication costs for various real-world benchmark applications, as well as a comparison of the proposed design time approach against other algorithms in terms of exploration time. The results show that the proposed design time approach usually achieves the optimal communication cost and does so with shorter computation times.

C. RUNTIME PERFORMANCE COMPARISON

We measure the performance of our algorithm for dynamic task allocation in terms of communication energy of allocated applications, finish time of mapped tasks, and average packet latency of tasks that belong to the applications. To test the adaptivity of our proposed approach to runtime conditions, applications arrive randomly at any time with changing amounts of PEs and virtual zones as depicted in the different test scenarios as shown in table 7. We selected an 8×8 mesh

Algorithm 2 Dynamic Mapping and Reconfiguration (DRA) Algorithm

Data: ATG (T, E), Core graph, Latency threshold Lth, commCost threshold Cth
Result: optimized dynamic mapping and reconfiguration
$\kappa_n \leftarrow$ getAppSize(ATG);
$\tau_{loc}, \tau_{nbr} \leftarrow$ getDesignTimeMappingInfo(κ_n);
$VZ_n \leftarrow$ assignVirtualZone(κ_n, VZ_{status});
$VZ_{status} \leftarrow VZ_n$;
$C_1 \leftarrow$ getFirstCore(VZ_n, τ_{loc});
$\tau_p \leftarrow$ find task T with max communication bandwidth;
$C_1 \leftarrow \tau_p$;
TaskList $\leftarrow$ get the list of tasks in ATG other than Tmax;
while *TaskList !=0* **do**
  Search for already mapped neighbor of current task $\tau_{current}$;
  if *noOfnbrMapped>0* **then**
    Find core C for mapping task $\tau_{current}$ close to already mapped neighbor;
    VZ_n[C] $\leftarrow \tau_{current}$
  end
  if *noOfnbrMapped == 0* **then**
    find core C for mapping task $\tau_{current}$ from available PEs with min MHD;
    VZ_n[C] $\leftarrow \tau_{current}$
  end
  Remove Tc from TaskList;
  Update TaskList ;
  if *TaskList == 0* **then**
    finalMapping $\leftarrow$ mesh with mappedTasks;
    finalCost $\leftarrow$ communication cost of current mapping;
  end
end
Lavg $\leftarrow$ caclulateAvgNetworkLatency (finalMapping);
if *Lavg >= L_{th} or finalCost >= c_{th}* **then**
  Find free core C for reconfiguring the task mapping;
  Reconfigure task to free PEs available;
end
finalMapping $\leftarrow$ mesh with mappedTasks;
finalCost $\leftarrow$ communication cost of current mapping;

NoC to implement the proposed dynamic approach, which is further split into four 4 × 4 virtual zones. In the first test scenario, all virtual zones and their respective PEs are idle, whereas, in the second test scenario, one virtual zone is allocated to the first arriving application. Similarly, test scenarios 3, 4, and 5 assume that all virtual zones are occupied by applications, while the sixth test scenario is where all virtual zones are taken up and task reassignment is required

TABLE 3. Standard real-world benchmarks applications.

Benchmarks	Number of edges	Number of Nodes	2D Mesh Size
PIP	8	8	3x3
CABLC	22	16	4x4
VOPD	21	16	4x4
MPEG4	26	12	4x4
MWD	13	12	4x4
MP3Enc/MP3Dec	14	13	4x4
263dec/MP3dec	12	12	4x4
263dec/MP3dec	15	14	4x4
MIMS	38	25	5x5

TABLE 4. Synthetic applications.

Benchmarks	Number of tasks	2D Mesh Size
TGFF G1	32	6 x 6
TGFF G2	64	8 x 8
TGFF G3	96	10 x 10
TGFF G4	128	12 x 12

TABLE 5. Description of the simulation environment.

Settings	Details
Type of network	2-D mesh
Mesh dimension	3 x 3 ,4 × 4, 5 x 5, 6 × 6, 8 × 8, 10 × 10, 12 × 12
Applications per Mesh	1,2,3,4,5
Total Tasks for large applications	6, 8,9,12,13,14,16,22,32 64,128
Platform	Embedded and synthetic
Packet mode	Without ACK
ACK Sending procedure	Optimally send ACK
Packet delivery	Exponential
Length of packet	10 (flits)
Flit rate of injection	0.1 (flits/cycle/node)
Type of router	WORM HOLE PIPELINE
Routing algorithm	XY
Buffer Size	1 (flit)
Inter router link length	1000 (um)
Pipeline type	8
Pipeline stages	4
Input voltage	1(V)
Operating clock frequency	1000 (MHz)
Warm-up time	20000 cycles

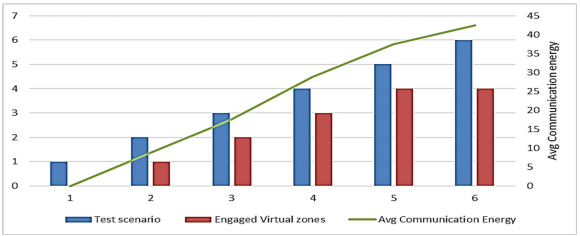


FIGURE 6. Evaluation of Communication Energy.

to account for changes in network conditions. We mapped applications with varying numbers of tasks to different virtual zones and simulated 100 times in order to obtain average results for all test scenarios.

D. COMMUNICATION ENERGY

Here, we assess the effectiveness of the energy-aware version of our dynamic task allocation strategy based on the communication energy consumed by the allocated applications. The communication energy consumed by the mapped applications

TABLE 6. Communication cost (Bw X Nh), design time performance comparison against real-world benchmark applications.

Algorithms	VOPD	CAVLC	MMS	MPEG4	MWD	Mp3EncMp3Dec	263encMP3dec	263decMP3dec	PIP
ILP	4119	-	-	3567	1120	17.021	230.407	19.823	-
ONMAP	4119	-	663379	3567	-	-	-	-	640
GA	4141	-	-	3567	1321	17.133	230.69	19.911	-
SA	4125	-	-	3567	1451	-	-	-	-
BEMAP	4119	6701	664636	3567	-	-	-	-	640
ACO	-	-	-	3670	-	17.231	-	-	-
PSO	4119	-	-	3567	1120	17.021	230.45	19.823	-
BA	4119	-	-	3567	1120	17.834	231.45	19.936	-
HDPSO	4119	-	-	3567	-	-	-	-	-
SCSO	4119	-	-	3567	1122	17.021	230.407	19.823	-
Proposed	4119	6691	652637	3567	1120	17.021	230.407	19.823	640

TABLE 7. Test scenarios for dynamic mapping.

Test scenarios	Initial configuration
Test scenario 1	All virtual zones free
Test scenario 2	1 x virtual zone engaged
Test scenario 3	2 x virtual zone engaged
Test scenario 4	3 x virtual zone engaged
Test scenario 5	All virtual zones engaged
Test scenario 6	All virtual zones engaged , Tasks positions reconfigured

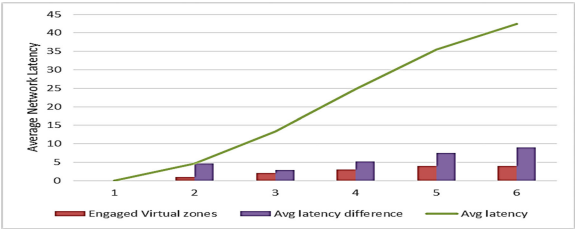


FIGURE 7. Evaluation of average network Latency.

during their execution is illustrated in Figure 6. On average, test scenarios 2, 3, 4, and 5 result in 8.9%, 17.6%, 28.8%, and 37.5% more communication energy consumption when compared to test scenario 1, respectively. Test scenario 6 has the highest communication energy consumption of 42.4% due to task relocation. Test scenario-1 gives the least communication energy for allocated applications since all PEs are assumed to be free for task assignment. As more PEs are progressively occupied from test scenario-2 to scenario-5, the mapped region becomes less dense due to the lack of available PEs in the regular area for task assignment. This increases the communication energy of the mapped applications.

E. AVERAGE NETWORK LATENCY

The average network latency of the assigned applications under different test scenarios is presented in Figure 7. We can see that as the PE occupancy increases from the beginning of the test scenarios, the latency of the allocated application increases by 9.02%. As more PEs are located in different virtual zones of the platform, the allocated regions become non-uniformly shaped. Tasks communicate with PEs that are spaced farther apart, resulting in increased communication costs. As data packets travel from source to destination PEs, more network resources, such as routers and links, are used for communication between dependent tasks. Furthermore, PEs located in regular regions have reduced communication energy compared to PEs that are spread out across the platform. In general, the allocated applications have 23.5% higher latency in test scenarios with more active virtual zones.

F. COMPARISON WITH EXISTING WORKS

In this section, we evaluate the performance of the proposed hybrid runtime approach HyDra to the performance of the selected state-of-the-art runtime mapping approaches for NoC-based multicore systems published in the literature. The Nearest-Neighbor (NN) [18] approach, which is regarded as the standard way, is used to represent the findings as normalized values. Figures 8 to 10, represent the improvement in reducing the performance metrics in percentage as compared to the baseline algorithm NN. Here, we want to clarify that our study focuses on constantly running applications, in which a collection of periodic activities is carried out frequently during the application. Examples of these applications include software radios, streaming apps, audio, video encoders, and decoders, to mention a few. A single iteration also refers to the whole cycle during which all of an application's recurring activities are carried out. The results presented here are based on the average of all the execution cycles.

1) COMMUNICATION COST

The distance between the source and destination PEs where communicative tasks are allocated and route congestion has an impact on the average packet latency. It might be very cost-effective to assign independent and communicative tasks to the same or nearby PEs. Figure 8 displays the communication costs for various methods for lower and larger NoC sizes. Because they do not perform any pre-processing and do not account for overhead when mapping the tasks, the FF and NN algorithms have the greatest communication costs for both small and large NoC sizes. According to the experimental findings, the proposed algorithm can lower communication costs in comparison to CPNN, especially in bigger mesh NoCs. Particularly, for small and large mesh NoCs. The proposed method achieves 6% and 8% reduced communication cost in comparison to CPNN. Similar to this, the proposed technique reduces communications costs over FF by 35% and 28% for smaller and larger mesh NoCs, respectively. According to experimental findings, the proposed algorithm may greatly lower communication costs, which not only

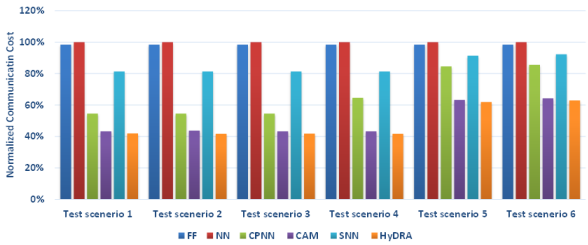


FIGURE 8. Comparison of communication cost.

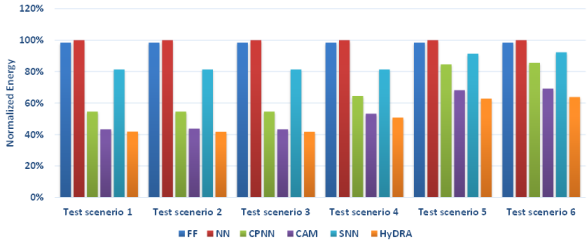


FIGURE 9. Comparison of energy Consumption.

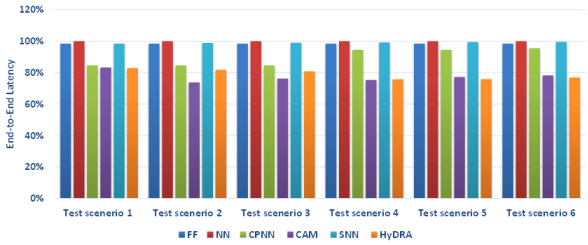


FIGURE 10. Comparison of end-to-end latency.

lowers energy consumption but also enhance application performance.

2) ENERGY CONSUMPTION

The energy usage values for smaller and larger mesh sizes are shown in Figure.9. Experimental results show that, among all methods for lower mesh NoCs, the proposed algorithm consumes the least energy. More precisely, the proposed approach uses 47% less energy than the FF algorithm and 6% less energy than the CPNN for lower mesh NoCs. For bigger mesh NoCs, the proposed algorithm's energy consumption is 1% to 2% higher than that of CPNN.

3) APPLICATIONS-SPECIFIC END-TO-END LATENCY

Because dependent processes don't have to wait as long for the needed data, a decrease in application-specific end-to-end latency can significantly increase application throughput and speed up task execution. Figure.10, displays the findings of a comparison of application-specific end-to-end latency. In comparison to prior techniques for both smaller and higher NoC sizes, the proposed algorithm has reduced application-specific end-to-end latency. When compared to CPNN, the proposed algorithm's reduction in latency is more noticeable for large sizes. More precisely, for smaller NoCs 7% less and 31% less for bigger NoCs.

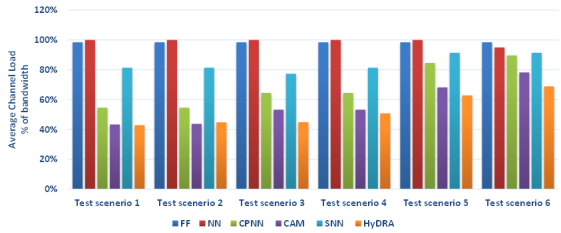


FIGURE 11. Comparison of average channel load.

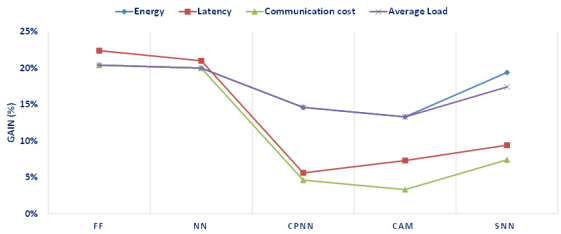


FIGURE 12. Gain of proposed clustering approach over non-clustering approaches.

4) AVERAGE CHANNEL LOAD

The amount of activity in the Network-on-Chip (NoC), which reflects how much the NoC is being utilized, can be determined by measuring the average channel load. To calculate this load, the usage of all channels is recorded at regular intervals during the execution of all applications. The channel load is influenced by the communication overhead involved in transmitting information between tasks and the amount of traffic generated by tasks that communicate with each other across different nodes. The average channel load under different injection rates for all simulation scenarios is illustrated in Figure.11. When the proposed technique is applied, the average channel load is noticeably decreased. Furthermore, the proposed technique demonstrates superior performance in comparison to other methods.

5) PROPOSED APPROACH VS. NON-CLUSTERING APPROACHES

The proposed approach is based on design time clustering and runtime virtual channels. The clustering strategy involves first sorting applications by the number of tasks contained inside them and then mapping the initial tasks of applications at the center of the clusters sorted by their processing capabilities. Whereas, applications are not sorted and their initial tasks are mapped at any random position in the non-clustering technique. For various mapping techniques, Figure.12 illustrates the gain achieved by the proposed design time clustering approach over the non-clustering approaches in terms of energy consumption, average latency, communication costs, and average channel load.

VI. CONCLUSION

In this work, a hybrid mapping and reconfiguration strategy for NoC has been presented. Our approach requires allocation exploration at design time, runtime mapping, and

reconfiguration using the best design time mapping. The best mappings having the lowest latency and energy values are identified during the design time. The optimal mappings against each application selected and placement of core with the top task stored along with the information of communication dependent tasks. The proposed approach uses various solutions during runtime, subject to the availability of required resources and application timing needs, to adapt to the dynamic of the application workload. We have assessed our method and compared it with various task mapping methods that have been mentioned in the literature. The effectiveness of the proposed approach has been demonstrated in experiments by the reduced latency, communication cost, and communication energy consumption of the assigned applications. The proposed method is therefore appropriate for real-time workload distribution of dynamic workloads with multiple applications on NoC-based multicore systems. In future work, We believe that with few modifications, the proposed technique might be used with more sophisticated NoC-based MPSoC platforms, including heterogeneous core systems and hierarchical wireless nodes.

REFERENCES

[1] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. 38th Design Autom. Conf.*, Jun. 2001, pp. 684–689.

[2] B. Gaide, D. Gaitonde, C. Ravishankar, and T. Bauer, "Xilinx adaptive compute acceleration platform: Versal TM architecture," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2019, pp. 84–93.

[3] J. Hu and R. Marculescu, "Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints," in *Proc. Design, Autom. Test Eur. Conf. Exhib.*, Feb. 2004, pp. 234–239.

[4] N. Chatterjee, S. Paul, P. Mukherjee, and S. Chattopadhyay, "Deadline and energy aware dynamic task mapping and scheduling for network-on-chip based multi-core platform," *J. Syst. Archit.*, vol. 74, pp. 61–77, Mar. 2017.

[5] C.-L. Chou and R. Marculescu, "Run-time task allocation considering user behavior in embedded multiprocessor networks-on-chip," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 1, pp. 78–91, Jan. 2010.

[6] A. K. Singh, T. Srikanthan, A. Kumar, and W. Jigang, "Communication-aware heuristics for run-time task mapping on NoC-based MPSoC platforms," *J. Syst. Archit.*, vol. 56, no. 7, pp. 242–255, Jul. 2010.

[7] G. Mariani, P. Avasare, G. Vanmeerbeeck, C. Ykman-Couvreur, G. Palermo, C. Silvano, and V. Zaccaria, "An industrial design space exploration framework for supporting run-time resource management on multi-core systems," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2010, pp. 196–201.

[8] W. Quan and A. D. Pimentel, "A hybrid task mapping algorithm for heterogeneous MPSoCs," *ACM Trans. Embedded Comput. Syst.*, vol. 14, no. 1, pp. 1–25, Jan. 2015.

[9] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Resource and throughput aware execution trace analysis for efficient run-time mapping on MPSoCs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 1, pp. 72–85, Jan. 2016.

[10] C. Ykman-Couvreur, P. Avasare, G. Mariani, G. Palermo, C. Silvano, and V. Zaccaria, "Linking run-time resource management of embedded multi-core platforms with automated design-time exploration," *IET Comput. Digit. Techn.*, vol. 5, no. 2, pp. 123–135, 2011.

[11] E. Carvalho, C. Marcon, N. Calazans, and F. Moraes, "Evaluation of static and dynamic task mapping algorithms in NoC-based MPSoCs," in *Proc. Int. Symp. Syst. Chip*, Oct. 2009, pp. 87–90.

[12] V. Tsoutsouras, I. Anagnostopoulos, D. Masouros, and D. Soudris, "A hierarchical distributed runtime resource management scheme for NoC-based many-cores," *ACM Trans. Embedded Comput. Syst.*, vol. 17, no. 3, pp. 1–26, May 2018.

[13] B. Xie, T. Chen, W. Hu, X. Tang, and D. Wang, "An energy-aware online task mapping algorithm in NoC-based system," *J. Supercomput.*, vol. 64, no. 3, pp. 1021–1037, Jun. 2013.

[14] C.-L. Chou and R. Marculescu, "User-aware dynamic task allocation in networks-on-chip," in *Proc. Design, Autom. Test Eur.*, Mar. 2008, pp. 1232–1237.

[15] A. Mehran, A. Khademzadeh, and S. Saeidi, "DSM: A heuristic dynamic spiral mapping algorithm for network on chip," *IEICE Electron. Exp.*, vol. 5, no. 13, pp. 464–471, 2008.

[16] M. A. Al Faruque, R. Krist, and J. Henkel, "ADAM: Run-time agent-based distributed application mapping for on-chip communication," in *Proc. 45th Annu. Design Autom. Conf.*, Jun. 2008, pp. 760–765.

[17] E. Carvalho, N. Calazans, and F. Moraes, "Heuristics for dynamic task mapping in NoC-based heterogeneous MPSoCs," in *Proc. 18th IEEE/IFIP Int. Workshop Rapid Syst. Prototyping (RSP)*, May 2007, pp. 34–40.

[18] E. L. D. S. Carvalho, N. L. Calazans, and F. G. Moraes, "Dynamic task mapping for MPSoCs," *IEEE Design Test Comput.*, vol. 27, no. 5, pp. 26–35, Sep. 2010.

[19] C.-L. Chou and R. Marculescu, "Contention-aware application mapping for network-on-chip communication architectures," in *Proc. IEEE Int. Conf. Comput. Design*, Oct. 2008, pp. 164–169.

[20] S. Tosun, O. Ozturk, and M. Ozen, "An ILP formulation for application mapping onto network-on-chips," in *Proc. Int. Conf. Appl. Inf. Commun. Technol.*, Oct. 2009, pp. 1–5.

[21] W. Amin, F. Hussain, and S. Anjum, "IHPSA: An improved bio-inspired hybrid optimization algorithm for task mapping in network on chip," *Microprocessors Microsyst.*, vol. 90, Apr. 2022, Art. no. 104493.

[22] R. Pop and S. Kumar, "A survey of techniques for mapping and scheduling applications to network on chip systems," School Eng., Dept. Electron. Comput. Eng., Jönköping Univ. Jönköping, Sweden, Res. Rep. 04:4, 2004.

[23] P. K. Sahu and S. Chattopadhyay, "A survey on application mapping strategies for network-on-chip design," *J. Syst. Archit.*, vol. 59, no. 1, pp. 60–76, Jan. 2013.

[24] A. K. Singh, W. Jigang, A. Kumar, and T. Srikanthan, "Run-time mapping of multiple communicating tasks on MPSoC platforms," *Proc. Comput. Sci.*, vol. 1, no. 1, pp. 1019–1026, May 2010.

[25] S. Tosun, "Cluster-based application mapping method for network-on-chip," *Adv. Eng. Softw.*, vol. 42, no. 10, pp. 868–874, Oct. 2011.

[26] J. Hu and R. Marculescu, "Energy-aware mapping for tile-based NoC architectures under performance constraints," in *Proc. ASP-DAC Asia South Pacific Design Autom. Conf.*, Jan. 2003, pp. 233–239.

[27] J. Huang, C. Buckl, A. Raabe, and A. Knoll, "Energy-aware task allocation for network-on-chip based heterogeneous multiprocessor systems," in *Proc. 19th Int. Euromicro Conf. Parallel, Distrib. Netw.-Based Process.*, Feb. 2011, pp. 447–454.

[28] L. Moller, L. S. Indrusiak, L. Ost, F. Moraes, and M. Glesner, "Comparative analysis of dynamic task mapping heuristics in heterogeneous NoC-based MPSoCs," in *Proc. Int. Symp. Syst. Chip (SoC)*, Oct. 2012, pp. 1–4.

[29] T. D. Ngo, K. J. M. Martin, and J.-P. Diguet, "Move based algorithm for runtime mapping of dataflow actors on heterogeneous MPSoCs," *J. Signal Process. Syst.*, vol. 87, no. 1, pp. 63–80, Apr. 2017.

[30] L. G. G. Morales, J. E. A. Cobo, and N. Bagherzadeh, "A new approach to the population-based incremental learning algorithm using virtual regions for task mapping on NoCs," *J. Syst. Archit.*, vol. 97, pp. 443–454, Aug. 2019.

[31] B. Pourmohseni, S. Wildermann, M. Glaß, and J. Teich, "Hard real-time application mapping reconfiguration for NoC-based many-core systems," *Real-Time Syst.*, vol. 55, no. 2, pp. 433–469, Apr. 2019.

[32] J. Spieck, S. Wildermann, and J. Teich, "Scenario-based soft real-time hybrid application mapping for MPSoCs," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.

[33] S. Paul, N. Chatterjee, and P. Ghosal, "Dynamic task allocation and scheduling with contention-awareness for network-on-chip based multi-core systems," *J. Syst. Archit.*, vol. 115, May 2021, Art. no. 102020.

[34] N. Dahir, A. Karkar, M. Palesi, T. Mak, and A. Yakovlev, "Power density aware application mapping in mesh-based network-on-chip architecture: An evolutionary multi-objective approach," *Integration*, vol. 81, pp. 342–353, Nov. 2021.

[35] C.-H. Huang, "HDA: Hierarchical and dependency-aware task mapping for network-on-chip based embedded systems," *J. Syst. Archit.*, vol. 108, Sep. 2020, Art. no. 101740.

[36] A. Kumar, V. K. Sehgal, G. Dhiman, S. Vimal, A. Sharma, and S. Park, "Mobile networks-on-chip mapping algorithms for optimization of latency and energy consumption," *Mobile Netw. Appl.*, pp. 1–15, 2021.
[37] A. S. Kumar and T. V. K. H. Rao, "An adaptive core mapping algorithm on NoC for future heterogeneous system-on-chip," *Comput. Electr. Eng.*, vol. 95, Oct. 2021, Art. no. 107441.
[38] A. T. Tran and B. Baas, "Noctweak: A highly parameterizable simulator for early exploration of performance and energy of networks on-chip," VLSI Comput. Lab, ECE Dept., Univ. California, Davis, Davis, CA, USA, Tech. Rep. ECE-VCL-2012-2, 2012.
[39] D. P. Rini, S. M. Shamsuddin, and S. S. Yuhaniz, "Particle swarm optimization: Technique, system and challenges," *Int. J. Comput. Appl.*, vol. 14, no. 1, pp. 19–26, 2011.
[40] A. A. Morgan, H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Unified multi-objective mapping and architecture customisation of networks-on-chip," *IET Comput. Digit. Techn.*, vol. 7, no. 6, pp. 282–293, Nov. 2013.
[41] C. A. M. Marcon, E. I. Moreno, N. L. V. Calazans, and F. G. Moraes, "Comparison of network-on-chip mapping algorithms targeting low energy consumption," *IET Comput. Digit. Techn.*, vol. 2, no. 6, pp. 471–482, Nov. 2008.
[42] W. Amin, F. Hussain, S. Anjum, S. Khan, N. K. Baloch, Z. Nain, and S. W. Kim, "Performance evaluation of application mapping approaches for network-on-chip designs," *IEEE Access*, vol. 8, pp. 63607–63631, 2020.
[43] R. P. Dick, D. L. Rhodes and W. Wolf, "TGFF: Task graphs for free," in *Proc. 6th Int. Workshop Hardw./Softw. Codesign (CODES/CASHE)*, Seattle, WA, USA, 1998, pp. 97–101, doi: 10.1109/HSC.1998.666245.

WAQAR AMIN received the B.Sc. degree in computer engineering and the M.S. degree from the University of Engineering and Technology at Taxila (UET Taxila), Pakistan, in 2007 and 2014, respectively, where he is currently pursuing the Ph.D. degree. He has vast experience in the research and development of cellular protocols. His research interests include fault-tolerant systems, network-on-chip (NoC) system designs, machine learning, and cellular and satellite communications. He is also working on the low-cost application mapping on NoC.

FAWAD HUSSAIN (Member, IEEE) received the B.Sc. degree in computer engineering, the M.Sc. degree in electrical engineering, and the Ph.D. degree in computer engineering from the University of Engineering and Technology (UET) at Taxila, Pakistan, in 2005, 2009, and 2015, respectively. He is currently an Assistant Professor with the Computer Engineering Department, UET at Taxila. His research interests include speech and audio processing, computer vision, and network-on-chip (NoC).

SHERAZ ANJUM received the M.Sc. degree in electronics, the M.Sc. degree in computer engineering, and the Ph.D. degree in engineering from the Institute of Microelectronics, GUCAS, Beijing, China, in 1999, 2005, and 2008, respectively. Currently, he is an Associate Professor with COMSATS University Islamabad, Wah Campus, Pakistan. His research interests include networks-on-chip, machine learning, multi-processor heterogeneous computing, wired and wireless networks, and edge computing.

SHAROON SALEEM received the B.Sc. degree in computer engineering and the M.S. degree from the University of Engineering and Technology at Taxila (UET Taxila), Pakistan, in 2007 and 2015, respectively, where he is currently pursuing the Ph.D. degree. He is a Lecturer with the Computer Engineering Department, UET Taxila. He had been involved in various projects in the domain of embedded systems development in his career. His research interests include network-on-chip (NoC), fault tolerant systems, and reconfigurable digital system designs. He is also working on the low-complexity and low-cost application mapping techniques in network-on-chip.

WAQAR AHMAD received the Ph.D. degree in electronics and communication in Europe. He has more than 15 years of teaching, research, training, and development experience in computing and engineering fields. He has worked in various professional capacities across academia, such as the lab engineer, a lecturer, and an assistant professor of artificial intelligence, machine learning, and deep learning. He has a strong professional industrial collaboration with top-notch companies of artificial intelligence and hardware designs. He has supervised more than 40 students at B.S., M.S., and Ph.D. level in their research and development activities. He is a certified deep learning specialist with a focus on natural language processing (NLP), computer vision (CV), and speech recognition (SR). His research interests include natural language processing (NLP), Python programming, computer vision, artificial intelligence, machine learning, deep learning, digital system designs, microelectronics, field programmable gate arrays (FPGAs), and very large scale integration (VLSI).

MUBASHIR HUSSAIN received the Ph.D. degree from the University of New South Wales, Sydney, Australia, and the M.Sc. degree from the University of Engineering and Technology at Taxila, Pakistan. He is currently a Senior Lecturer with the School of Information Technology, King's Own Institute, Sydney. He has authored many scientific publications in the field of hardware security and network-on-chips. His research interests include embedded system designs, hardware security designs, network-on-chip, wireless sensor networks, and energy optimization.