Application Mapping Using Cuckoo Search Optimization With Lévy Flight for NoC-Based System

MUHAMMAD JUNAID MOHIZ¹, NAVEED KHAN BALOCH², FAWAD HUSSAIN⁴, SHAROON SALEEM¹, YOUSAF BIN ZIKRIA², (Senior Member, IEEE), AND HEEJUNG YU³, (Senior Member, IEEE)
¹Department of Computer Engineering, University of Engineering and Technology, Taxila 47050, Pakistan
²Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, South Korea
³Department of Electronics and Information Engineering, Korea University, Sejong 30019, South Korea
Corresponding authors: Yousaf Bin Zikria (yousafbinzikria@ynu.ac.kr) and Heejung Yu (heejungyu@korea.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korean Government [Ministry of Science and ICT (MSIT)], South Korea, under Grant 2019R1A2C1083988; in part by the Basic Science Research Program through the NRF grant funded by the Ministry of Education under Grant 2021R1I1A3041887; and in part by the MSIT through the Information Technology Research Center (ITRC) Support Program supervised by the Institute for Information & Communications Technology Promotion (IITP) under Grant IITP-2021-2016-0-00313.

ABSTRACT Network on chip (NoC) is a promising communication infrastructure for multiple cores on a chip to exchange data efficiently. In such NoC architecture, application mapping is a process of assigning tasks to the processing cores. An optimized application mapping technique enhances the performance of a chip and reduces the entire chip’s energy consumption. The optimization of application mapping is essential in the design of NoC. In this study, a greedy algorithm is utilized as the first technique to place the maximum communicating tasks together to give the main algorithm a head start. Then, a meta-heuristic Cuckoo Search via Lévy flight is employed further to optimize the placement of tasks on the NoC cores. The greedy algorithm furnishes a relatively pre-processed base to the cuckoo search optimization (CSO), which eventually helps in the fast convergence of the main algorithm. The analysis of the results shows that the proposed algorithm outperformed the state-of-the-art techniques in NoC application mapping in terms of various performance metrics, such as communication cost, energy consumption, and average packet latency.

INDEX TERMS Application mapping, communication cost, cuckoo search, latency, Lévy flight, network on chip.

I. INTRODUCTION

With the advancement of technology in the field of very large scale integration (VLSI), it is possible to integrate several computing elements onto a single die. As technology of semiconductor manufacturing process evolves in such a way, various processing and storage units are integrated in a single chip. The performance of these single bus-based models still suffers from signal integrity, propagation delays, and scalability on large-scale platforms. Therefore, network on chip (NoC) architecture integrating multiple cores in a single die has been emerged as an alternative.

NoC architecture is a packet-based inter-connected network that uses a packetized switching mechanism for data exchange between the cores, i.e., processing elements. The network consists of routers, physical links, and network interfaces for individual resources [1]. Researchers proposed different architectural modifications for NoC-based systems to improve the reliability and robustness of NoC platforms. In [2], the authors presented fault tolerant designs for various router components, i.e., routing computation (RC), virtual channel allocation (VA), switch allocation (SA) and crossbar (XB) stages. A fault tolerant router architecture design named as defender was presented in [3], which is capable of tolerating permanent faults in all the parts of the router. The authors in [4], [5] also presented modifications in the existing
NoC routers designs to propose a reliable on-chip network communication infrastructure. The efficiency of a network to exchange information heavily depends on the underlying topology. The latency, area, and power consumption for an NoC-based system are highly dependent on the selection of network infrastructure. Several topologies based on the interconnection network were proposed in the literature for NoC [6]. A mesh topology is the most commonly used among these topologies because it offers a regular fixed infrastructure with equal-sized communication links and high bandwidth among multiple cores.

An important step in NoC design after topology selection and flow control mechanism is the task mapping of applications on an underlying NoC topology to achieve optimal performance. The reduction of power consumption, area, and latency is more likely conceivable through competent modeling of applications onto NoC. Thus, application mapping is an important research dimension in NOC design and is considered as a non-polynomial hard (NP-hard) problem [7]. To map an application with \( n \) number of tasks requires \( n! \) computations, and this computation complexity increases exponentially as the number of tasks increases. Hence, it is not feasible to solve large combinative problems by linear programming methods. Therefore, researchers have proposed various heuristic and metaheuristic based methods to solve these complex optimization problems. Usually, algorithms based on swarm intelligence (metaheuristics) often provide a better solution by exploring possible solutions with low computational overhead than mathematical approaches based on an optimization framework. Metaheuristic algorithms are self-learning algorithms evolved from imitating the intelligent behaviors and processes adopted from nature, sociology, thinking, and other disciplines. Metaheuristics algorithms, such as genetic algorithm (GA) [8], particle swarm optimization (PSO) [9], whale optimization algorithm (WOA) [10], multi-objective marine predator algorithm (MOMPA) [11], etc., are smart enough to find the optimal or near-optimal solutions for complicated optimization problems [12].

In recent studies, researchers have widely adopted a combination of meta-heuristic algorithms with modern engineering techniques to find optimal or near-optimal solutions in various optimization problems. For example, in [13], the authors proposed an algorithm that utilizes the Lévy flight with PSO for the deployment of a wireless local area network (WLAN) in a real 3-D environment. The authors in [14], adopted PSO enhanced random forest (RF) model for the diagnosis of spontaneous rupture of ovarian endometriomas. The authors in [15], utilizes fruit fly optimization to fine-tune the parameters for support vector regression (SVR) to predict the vacant spaces in the parking lot. Moreover, researchers have also adopted meta-heuristic algorithms to aid different computing techniques to solve application mapping problems for NoC-based systems. For example, genetic algorithm (GA) and simulated annealing (SA) are utilized to map the tasks to the cores of NoC in [16], which is optimize for power consumption and performance using fuzzy rules.

In a quick review of the literature [12], [17]–[20], it can be found that the cuckoo search optimization (CSO) algorithm is a population-based algorithm similar to PSO and GA. However, CSO outperforms both PSO and GA for two primary reasons: CSO provides a fine balance between randomization and local search. The number of control parameters in CSO is less than those in the other algorithms. A good balance between the local search approach and an effective global search strategy applied on the whole search space makes the CSO a more efficient algorithm in the metaheuristic cadre. Moreover, the algorithm is controlled with only two parameters, the population size and probability of low-quality nests.

Additionally, Lévy flight is another search algorithm inspired by flies and animal behavior for a random search of food. It is considered as the best technique in finding a specific target in an unfamiliar environment [21]–[23]. Lévy flight is popular because of its inherent features such as low complexity, fewer parameters, and ease of implementation. Therefore, scientists have applied the Lévy flight independently or combined with several metaheuristic algorithms to solve various engineering problems.

Motivated from the above, the CSO algorithm, which is based on a brood parasitism behavior observed in some cuckoo species together with Lévy flight random walk, is leveraged to solve the application mapping problem in this study. The greedy algorithm is adopted to generate the initial population, and Lévy flight is utilized as a local search strategy which eventually provides the base for the fast convergence of the algorithm. To the best of our knowledge, CSO via Lévy flight is introduced for the first time for application mapping problems in a NoC-based system.

The main contribution of this research is the utilization of a metaheuristic-based CSO algorithm to solve the NP-hard application mapping problem in NoC-based systems. The algorithm is optimized by using the Greedy algorithm so that the overall computation overhead is reduced compared to the existing algorithm of its cadre. The proposed algorithm efficiently maps the application on the NoC platform, and results indicate that the proposed method provides optimal communication cost for all benchmarks. Also, it shows reasonable improvements in power consumption and latency in comparison with state-of-the-art algorithms.

The rest of the paper is organized as follows. Section II presents previous research work on application mapping in the literature. Section III articulates the problem formulations and method of analysis. In section IV, we introduce the proposed algorithm based on CSO via Lévy flight for application mapping. In section V, numerical evaluation and validation of the CSO algorithm are shown with benchmarks. Section VI presents concluding remarks.

II. RELATED WORK

Application mapping has to turn out to be an important part of NoC architecture design. In this problem, the association of tasks of an application to the cores of NoC architecture is a challenging job. Various mapping techniques
were developed based on performance metrics, like bandwidth requirement, power, energy consumption, latency, and throughput of multi-core systems. Many researchers proposed different algorithms to address the application mapping problem. In [24], the authors presented a mathematical mapping approach based on integer linear programming (ILP) for two-dimensional (2D) mesh NoC systems. Though ILP offers an optimal or near-optimal solution, it requires high computation complexity as the number of tasks in an application increases. A clustering-based relaxation for ILP formulations was proposed in [25] to reduce run-time. An ILP-based formulation with the minimized network contention was presented in [26]. It attempts to minimize energy consumption by analyzing the factors that produce network contention and shut down certain communication links in NoC. In [27], [28], authors utilize a linearized model of quadratic assignment problem (QAP) to solve application mapping problems for NoC-based systems.

In [1], a comprehensive survey on performance comparison and analysis of different application mapping approaches proposed in the last decade for NoC designs was presented. According to this survey, algorithms based on a heuristic approach are fast and usually more beneficial when the algorithm’s execution time is a critical factor for mapping an application. However, these approaches may not provide an efficient mapping solution regarding performance constraints, such as communication cost, energy consumption, and average packet latency. In [29], a near-optimal mapping technique (NMAP) was presented, which minimizes the communication delay by reducing a routing path in a mesh network based on bandwidth requirements between the tasks of an application. In [30], the author presented a heuristic algorithm, which is called CastNet, to reduce energy consumption. In this algorithm, the author also considered a bandwidth constraint for task mapping. Another systematic search algorithm, i.e., segmented brute force mapping (SBMAP) algorithm, was presented in [31]. This algorithm segmented applications tasks into multiple sections and modular searches were applied to each segment to find the optimum mapping. A branch and bound-based exact mapping (BEMAP) algorithm based on amalgam optimization technique was presented for application mapping in [32]. In BEMAP, an initial mapping step is performed with a fast branch and bound algorithm. Then a modular exact mapping algorithm is adopted to produce an efficient and optimized solution to the mapping problem.

A metaheuristic algorithm based on simulated annealing (SA) was used for application mapping on 2D NoC in [33]. Still, it requires a long execution time, i.e., high complexity, to find an optimal solution. An ant colony optimization (ACO) algorithm, which is inspired by ant population behavior to search their food, was proposed for application mapping problem [34]. In [35], application mapping based on a two-step genetic algorithm (GA), which minimizes computational overhead, was presented. At first, the tasks are allocated to different intellectual property (IP) cores, assuming that every edge has a constant delay, equivalent to the average delay of all edges. Next, the cores of NoC are rearranged, considering the actual delays to minimize the overall delay of the system. A GA-based technique in [36] considered several important factors, such as contention in a network and packet length, to reduce the average delay of the network. In 1995, Kennedy and Eberhart discovered a population-based optimization approach, i.e., particle swarm optimization (PSO), motivated by the social behavior of bird’s swarm or fish schooling. In this technique, multiple solutions may exist and work together to find an optimal target. In this algorithm, each particle is considered a candidate solution, and its fitness value assesses the quality of the solution. In [37], a PSO-based algorithm was presented to minimize the communication cost while considering the static operation of a NoC system.

From the above literature review of heuristic-based algorithms, it is observed that the applied metaheuristic algorithms for application mapping fail to balance between the finding of deterministic optima and random behavior. In the study of metaheuristic algorithms like GA and PSO, it can be found that the population in the next generation is dependent on the fitness quality of the previous population, which makes them prone to get stuck in their local optima easily. In recent research, another bio-inspired search algorithms being used in [38] for an application mapping problem based on the chicken swarm optimization technique (SCSO). The authors used the k-nearest neighbor clustering technique to generate initial mapping, and then optimized results were obtained with the SCSO algorithm. This algorithm performs well in communication cost and power reduction for standard and randomly generated graphs. We have presented a mapping algorithm using a greedy approach with Cuckoo Search Optimization (CSO), which shows optimality in results with relatively low computational overhead in fewer iterations. Also, in CSO, the probability of the worst solutions being replaced with new solutions by exploring whole search space prevents the algorithm from getting stuck in local optima.

### III. PROBLEM FORMULATION

In this section, a mapping problem for NoC is formulated. To this end, several mathematical models are introduced.

**Definition 1:** A communication task graph (CTG) $P(T, C)$ is a directed graph where $T = \{t_1, t_2, t_3, \ldots, t_n\}$ denotes a set of vertices representing tasks and $C = \{c_{i,j} | i, j = 1, 2, 3, \ldots, n\}$ denotes a set of directed edges representing the communication between the tasks (vertices) $t_i$ and $t_j$. And the weight of edge $c_{i,j}$ represents the communication bandwidth $B_{i,j}$ between the tasks $t_i$ and $t_j$. Here, $n$ stands for the total number of tasks in CTG.

**Definition 2:** An NoC topology graph (NTG) $Q(V, E)$ is a directed graph where $V = \{v_1, v_2, v_3, \ldots, v_n\}$ is a set of vertices representing nodes/tiles and $E = \{e_{i,j} | i, j = 1, 2, 3, \ldots, n\}$ denotes a set of directed edges representing a physical link between the vertices (nodes/tiles) $v_i$ and $v_j$. Here, $n$ is the total number of nodes/tiles in NTG.
The mapping of the CTG, $P(T, C)$ onto the NTG, $Q(V, E)$ is defined by one to one mapping function, represented as

$$map : T \rightarrow V, \text{ such that } map(t_i) = v_i, \forall t_i \in T, \exists v_i \in V$$

(1)

where mapping is defined when $|T| \leq |V|$. Here, $\cdot$ denotes the cardinality, i.e., size of a set.

To calculate the performance parameters such as communication cost, latency, throughput, energy, and power consumption, the following mathematical models are used as in [39]. The following equation is used to calculate the communication cost.

$$Cost = \sum_{i,j} [B_{i,j} \times N_{i,j}]$$

(2)

where, $B_{i,j}$ denotes the communication bandwidth between task $t_i$ and $t_j$, $N_{i,j}$ is the Maximum distance between two cores associated with tasks $t_i$ and $t_j$ in the NoC architecture and $n$ is the total number of tasks. Then, $N_{i,j}$ is obtained by

$$N_{i,j} = |a_i - a_j| + |b_i - b_j|$$

(3)

where $(a_i, b_i)$ and $(a_j, b_j)$ are coordinates of $v_i$ and $v_j$, which are associated with $t_i$ and $t_j$, respectively, in a NoC topology.

Latency has a dominant effect on network traffic and is also considered as the second important constraint in this research. The average latency $L_{av}$ of network is given by

$$L_{av} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{N_i} \sum_{j=1}^{N_i} L_{t_{i,j}}$$

(4)

where $L_{t_{i,j}}$ is the latency of a packet $j$ at a destination node $i$, $N$ represents the number of processors in the platform, and $N_i$ corresponds to the number of packets encountered by individual processing element $i$, after warm up time.

The average throughput of the NoC network, $Th_{av}$, is determined by

$$Th_{av} = \frac{1}{N(T_{sim} - T_{wrm})} \sum_{i=1}^{N} N_i$$

(5)

where $T_{sim}$ is the simulation time and $T_{wrm}$ is warm-up time of the simulation.

The average power of a network, $Pw_{av}$, is represented by

$$Pw_{av} = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N_i} [\alpha_{i,j} \cdot Pw_{act,j} + (1 - \alpha_{i,j}) \cdot Pw_{inact,j}]$$

(6)

where $Pw_{act,j}$ and $Pw_{inact,j}$ denotes the active and inactive power of component $j$ and $\alpha_{i,j}$ is the measurement of activeness, i.e., an active probability of component $j$ in router $i$, after warm-up time.

Additionally, the average energy consumed by each packet in a network is given by

$$En_p = \frac{(T_{sim} - T_{wrm})}{(N \cdot N_p)} \sum_{i=1}^{N} \sum_{j=1}^{N_i} [\alpha_{i,j}Pw_{act,j} + (1 - \alpha_{i,j}) \cdot Pw_{inact,j}]$$

(7)

where $N_p \left(= \sum_{i=1}^{N} N_i \right)$ is the total number of packets transferred on the network.

**IV. MAPPING USING CSO ALGORITHM**

In this section, an application mapping using the CSO algorithm for NoC will be discussed.

**A. CUCKOO BREEDING BEHAVIOR**

Cuckoo search is a metaheuristic algorithm for optimization, which is inspired by biological behaviors of cuckoo birds [12]. This optimization algorithm is based on a broad parasitism breeding strategy of some species of cuckoos along with a Lévy flight behavior of few birds and fruit flies. Cuckoos rely on other birds to raise their chicks by laying their eggs in the nest of other birds. If the host bird recognizes the cuckoo’s egg, it will either throw away the discovered egg or disown the nest and build a new one somewhere else. Some feminine parasitic cuckoos are experts in mimicking the color and pattern of selective host bird’s egg. This minimizes the probability of laid eggs being recognized by host birds, thus increasing population growth chances. In addition, cuckoos usually select a nest where eggs were just laid by the host bird. Cuckoo’s egg characteristically hatches before the host bird’s eggs, and the chick throws away the other eggs. Moreover, the cuckoo’s chick is so adaptive that it can imitate the call of host birds for feeding.

**B. LÉVY FLIGHT**

Lévy flight behavior was observed in flies and animals for a random search of food, and it is considered as the best technique in finding a specific target in an unfamiliar environment [22]. It defines random walks in a search space in which step-length follows a Lévy distribution. Lévy distribution is defined in the following equation.

$$Lévy(t) = t^{-\lambda},$$

(8)

where $1 < \lambda \leq 3$, and $t$ is the iteration number.

Lévy flight is applied on finest solutions achieved so far to get new solutions in next generation. This helps to expedite the local search process and fast convergence of the algorithm.

**C. CUCKOO SEARCH VIA LÉVY FLIGHT**

Cuckoo search via Lévy flight [17], is based on three basic rules as follows:

1) Each cuckoo lays only one egg in a randomly chosen nest.
2) A nest having good quality eggs will carry forward to the next population.
3) A fixed number of host nests are available for breeding, and $P_a \in (0, 1)$ is the probability of cuckoo’s eggs being recognized by the host bird. In this situation, the host bird will either throw away the discovered egg or disown the nest and build a new one somewhere else. In other words, fraction $P_a$ of a total number of nests are discarded and substituted with new nests (solutions).
of an egg. In our case, the egg represents the set of tasks of Algorithm 1 and mathematically represented as for CSO is generated with a greedy random algorithm shown in Figure 1 and the fitness calculation of a single nest. An initial population of nests for generation is generated by using Algorithm 1. At first, a randomly selected task in is placed to a random tile in . Then, for each task yet to be mapped, the task with the highest communication requirement with already mapped task(s) will be selected. This task is then assigned to the tile of mesh that has the least hop count with the mapped task(s) and has a minimum impact on communication cost. If there is more than one tile available which have minimum or the same cost, then, a tile from the minimum cost tileset will be selected randomly, as depicted in Figure 2. The process is repeated until all the tasks are mapped. In the end, a mapped vector is returned as a single nest.

Algorithm 1: Greedy Random Algorithm

\begin{algorithm}
\caption{Greedy Random Algorithm}
\begin{algorithmic}
\State initialize \textit{map\_vector}(W, H) to $\varphi$
\State assign a random vertex in $P(T, C)$ to \textit{init\_task}
\State assign a random vertex in $Q(V, E)$ to \textit{init\_tile}
\State map \textit{init\_task} to \textit{init\_tile}
\State remove \textit{init\_tile} from $Q$, \textit{init\_task} from $P$ and add \textit{init\_tile} to \textit{map\_vector}
\While{$|T| > 0$}
\State allocate the vertex in $P$ with maximum comm with $\forall w_j \in W$, to \textit{next\_task}$\\textit{for $\forall v_j \in Q$ and $w_i \in \text{map\_vector}$ do}$
\State $\text{comm.cost}(v_j)+ = \text{comm.next\_task, map}^{-1}(w_i) \times [\text{xdist}(w_i, w_j) + \text{ydist}(w_i, w_j)]$
\State \text{add $v_j$ with minimum cost to set \text{mincost\_tiles}}$
\State randomly select \textit{next\_tile} from \text{mincost\_tiles} set
\State map \textit{next\_task} to \textit{next\_tile}$\\textit{remove \textit{next\_tile} from $Q$, \textit{next\_task} from $P$ and add \textit{next\_tile} to \textit{map\_vector}}$
\EndWhile \State \text{return \textit{map\_vector}(W, H)}$
\end{algorithmic}
\end{algorithm}

Initial population of $N$ nests for generation $t = 0$ is generated by using Algorithm 1. At first, a randomly selected task in $P(T, C)$ is placed to a random tile in $Q(V, E)$. Then, for each task yet to be mapped, the task with the highest communication requirement with already mapped task(s) will be selected. This task is then assigned to the tile of mesh that has the least hop count with the mapped task(s) and has a minimum impact on communication cost. If there is more than one tile available which have minimum or the same cost, then, a tile from the minimum cost tileset will be selected randomly, as depicted in Figure 2. The process is repeated until all the tasks are mapped. In the end, a mapped vector is returned as a single nest. The whole population is evaluated based on the fitness function, and the best nest with the highest fitness is recorded. The highest fitness corresponds to the lowest communication cost for mapping.

Further, the position of each nest of the previous generation is updated iteratively by applying Lévy flight to generate a new population for next-generation using Algorithm 2. A new generation is then evaluated, and the best nest with fitness value is updated with new nests if the fitness of the current nest is found higher than the previous one. Additionally, nest with quality fitness is carried over to next-generation, but fraction $P_a$ of the total number of nests with the lowest fitness are discarded and substituted with new

\begin{algorithm}
\caption{CSO FORMULATION FOR NoC MAPPING}
\begin{algorithmic}
\State $x_i^{t+1} = x_i^t + \alpha \odot \text{Lévy}(t)$ (9)
\end{algorithmic}
\end{algorithm}

where $\alpha > 0$ is a step size, which is subjected to the magnitude of the concern problem and usually the value is taken as $\alpha = 1$. The product $\odot$ means entry-wise multiplications. A random step length is drawn from a Lévy distribution (8), which is more effective in exploring the search space.

D. CSO FORMULATION FOR NoC MAPPING

Formulation of CSO algorithm for application mapping problem to reduce the communication cost is discussed in this section. In mapping problem, individual nest represents a single solution for a cuckoo (mapping). We use the communication cost given by (2) as our main objective function for the fitness calculation of a single nest. An initial population for CSO is generated with a greedy random algorithm shown in Algorithm 1 and mathematically represented as

\begin{equation}
 x_i^t = x_i^1, x_i^2, \ldots, x_i^N \text{ for } i = 1, 2, \ldots, N \quad (10)
\end{equation}

where $x_i^t$ represents the $i^{th}$ individual nest in the $t^{th}$ generation, $N$ represents the population size, and $n$ is the dimension of an egg. In our case, the egg represents the set of tasks for specific application and the nest represents the set of tiles with associated tasks. For simplicity, we assumed the dimension of the nest equal to the dimension of an egg for one-to-one mapping. The basic steps of the CSO algorithm for application mapping are summarized as the pseudocode in Algorithm 2.

\begin{algorithm}
\caption{Greedy Random Algorithm}
\begin{algorithmic}
\State initialize \textit{map\_vector}(W, H) to $\varphi$
\State assign a random vertex in $P(T, C)$ to \textit{init\_task}
\State assign a random vertex in $Q(V, E)$ to \textit{init\_tile}
\State map \textit{init\_task} to \textit{init\_tile}
\State remove \textit{init\_tile} from $Q$, \textit{init\_task} from $P$ and add \textit{init\_tile} to \textit{map\_vector}
\While{$|T| > 0$}
\State allocate the vertex in $P$ with maximum comm with $\forall w_j \in W$, to \textit{next\_task}$\\textit{for $\forall v_j \in Q$ and $w_i \in \text{map\_vector}$ do}$
\State $\text{comm.cost}(v_j)+ = \text{comm.next\_task, map}^{-1}(w_i) \times [\text{xdist}(w_i, w_j) + \text{ydist}(w_i, w_j)]$
\State \text{add $v_j$ with minimum cost to set \text{mincost\_tiles}}$
\State randomly select \textit{next\_tile} from \text{mincost\_tiles} set
\State map \textit{next\_task} to \textit{next\_tile}$\\textit{remove \textit{next\_tile} from $Q$, \textit{next\_task} from $P$ and add \textit{next\_tile} to \textit{map\_vector}}$
\EndWhile \State \text{return \textit{map\_vector}(W, H)}$
\end{algorithmic}
\end{algorithm}

\begin{algorithm}
\caption{CSO FORMULATION FOR NoC MAPPING}
\begin{algorithmic}
\State $x_i^{t+1} = x_i^t + \alpha \odot \text{Lévy}(t)$ (9)
\end{algorithmic}
\end{algorithm}

where $\alpha > 0$ is a step size, which is subjected to the magnitude of the concern problem and usually the value is taken as $\alpha = 1$. The product $\odot$ means entry-wise multiplications. A random step length is drawn from a Lévy distribution (8), which is more effective in exploring the search space.
Algorithm 2: Cuckoo Search via Lévy Flight

Input: \( P(T, C), Q(V, E) \)

Output: Mapping (best_nest)

Begin

Generate initial population of \( N \) host nests \( x_0^i \) for \( i = 1, 2, \ldots, N \) using algo-1;

Evaluate Fitness \( F_0 \) using (2) and record best nest to best_nest;

while \( (t < \text{Max Generation}) \) do

Carry out Lévy flight to get new nests for \( x_t^i \);

Evaluate its quality/fitness \( F_t^i \);

Assign the best among \( N \) nests to new_nest for \( F_t^i \);

if \( F_t^i < F_{t-1}^i \) then

Replace by new solution: best_nest = new_nest;

else

\[ F_t^i = F_{t-1}^i; \]

A fraction \( P_a \) of worse nests are abandoned and new ones are built using algo-1;

Evaluate and keep the best_nest (or nests with quality solutions);

End

Return the best_nest;

End

V. SIMULATION RESULTS

In this section, the performance of the proposed CSO algorithm over performance parameters is evaluated against the existing mapping algorithms.

NoCtweak simulator [39] has been explored in literature review and is utilized in this research work for analysis of applied mapping technique. It has an ability to give multiple performance constraints like communication bandwidth, latency, throughput, power and energy consumption using (2)-(7).

The NMAP algorithm is already available in NoCtweak. We have implemented the proposed CSO, and the CastNet algorithms in NoCtweak simulator for application mapping, and the results are analyzed using the built-in functions in the simulator. Intel Core-i7 with a base frequency of 3.2 GHz platform with 8GB RAM (main memory) is utilized to run the simulator. The NoC performance parameters are calculated using the commercial 65nm CMOS standard cell library model embedded in NoCtweak [39].

To analyze and assess the effectiveness of the presented mapping approach, various experiments are conducted on real-world embedded NoC Benchmarks collected from literature like VOPD, MPEG4, MWD, PIP, MMS, 802.11arx, CAVL, DVOPD, TELECOM, 263enc3dec, mp3enc3dec, and 263dec3dec. We have set up the simulator with 1000 MHz operating frequency, packet length of 10 flits with an injection rate of 0.1 flits per cycle, and x-y ordered routing algorithm. Four stage pipelines with a buffer length of 10 flits are selected for simulation in NoCtweak.
TABLE 1. Square mesh size.

| Benchmark | No. of Nodes | Square Mesh (2D) |
|-----------|--------------|-----------------|
| VOPD      | 16           | 4x4             |
| MPEG4     | 12           | 4x4             |
| MWD       | 12           | 4x4             |
| PIP       | 8            | 3x3             |
| MMS       | 25           | 5x5             |
| 802.11arx | 24           | 5x5             |
| CAVLC     | 16           | 4x4             |
| DVOPD     | 32           | 6x6             |
| Telecom   | 16           | 4x4             |
| 263encMP3dec | 12   | 4x4             |
| MP3encMP3dec | 13   | 4x4             |
| 263decMP3dec | 14   | 4x4             |

TABLE 2. Rectangular mesh size.

| Benchmark | No. of Nodes | Rectangular Mesh (2D) |
|-----------|--------------|-----------------------|
| MPEG4     | 12           | 4x3                   |
| MWD       | 12           | 4x3                   |
| PIP       | 8            | 4x2                   |
| 802.11arx | 24           | 6x4                   |
| DVOPD     | 32           | 8x4                   |
| 263encMP3dec | 12   | 4x3                   |
| MP3encMP3dec | 13   | 7x2                   |
| 263decMP3dec | 14   | 7x2                   |

A. COMMUNICATION COST AND COMPUTATION TIME

We have selected a 2D mesh topology for application mapping and performed experiments on two different types of mesh sizes. Square mesh in which the number of routers in a row is equal to the number of routers in a column in the NoC structure. In this case, routers may exceed the number of nodes in an application for a few embedded benchmarks. Table 1 shows the square mesh size utilized against each benchmark.

The Greedy algorithm is leveraged to generate the initial population by placing maximum communicating tasks in close vicinity. In the initial population, a relatively pre-processed base is furnished to the main algorithm, and Lévy flight is utilized as a local search strategy which eventually provides the foundation for the fast convergence of the algorithm. Moreover, the probability of worst solutions $P_a$, abound balances the global and local search, which implicitly generates new quality solutions as the algorithm progresses iteratively. Therefore, the CSO algorithm provides an optimal solution against each benchmark in relatively low computational overhead, which makes the proposed algorithm more efficient among other state-of-the-art algorithms.

Table 3 represents the estimation of the communication cost (Manhattan distance \times Bandwidth) in terms of MB/s for several benchmarks with different mapping algorithms on the 2D square mesh. The obtained results indicate that the proposed algorithm, i.e., CSO, performs better than NMAP, GA, CastNet, SMBAP, BEMAP, and in some cases better than PSO and SCSO. CSO provides the lowest communication cost against each benchmark, like ILP based exact mapping technique. It is considered one of the best methods for application mapping in terms of communication cost. CSO has 3.54% improvement in cost over NMAP, 2.4% over GA, and 0.39% over CastNet for VOPD for square mesh as listed in Table 4. The cost-saving of the proposed algorithm for MPEG4 is 2.94% for NMAP, 5.75% for GA, 7.99% for CastNet, 25.43% for SMBAP, and 5.76% for BEMAP. For MWD, CSO has 17.14% better results in cost than NMAP, 17.95% than GA, 14.29% than CastNet, SMBAP, and BEMAP 0.18% than SCSO algorithm. CSO also performs better than PSO and SCSO in some applications. For example, for DVOPD, CSO has 2.42% savings in cost compared to PSO and 0.18% compared to SCSO for MWD application. On average, CSO has an improvement of 11.02% over NMAP, 5.33% over GA, 4.28% over CastNet, 5.95% over SMBAP, 2.16% over BEMAP, 0.30% over PSO, and 0.03% over SCSO algorithm. BEMAP performs better in 802.11arx and CAVLC applications but has more computation time than the proposed algorithm.

Apart from this, results are also examined for rectangular mesh structures. In rectangular mesh, the number of routers in rows and columns is not equal. We have experimented with benchmarks with a tendency to map on rectangular mesh and whose number of task counts are less than the number of routers in the square mesh. For example, MPEG4 and MWD both have 12 nodes each. Square mesh for these applications is of $4 \times 4$ size with 16 routers in total. This will cost 4 extra routers in structure and will have a significant effect on the power and energy consumption of the system. An alternative approach we have adopted is rectangular mesh. For MWD and MPEG4, we have used a $4 \times 3$ mesh size with 12 routers in total equal to the number of nodes in an application. Hence, we can save power and energy consumption, especially in battery-powered NOC systems. Table 2 shows the mesh size we have used for each benchmark in rectangular configurations.

Table 5 shows the communication cost comparison of proposed CSO algorithm with NMAP and CastNet for mapping onto the rectangular mesh. According to the results obtained, CSO algorithm outperforms both algorithms in rectangular arrangement of mesh. Network cost savings of CSO as compared to NMAP are 50%, 7.097%, 13.16%, 29.72%, 8.60%, 20.27%, 5.84%, and 32.91%, and 10%, 6.04%, 10.53%, 5.26%, 2.07%, 0%, 9.58%, 2.52% as compared to CastNet for PIP, MPEG4, MWD, 802.11arx, DVOPD, 263encMP3dec, MP3encMP3dec and 263decMP3dec applications, respectively.

Computation time for CSO is compared with the algorithms of its cadre, listed in Table 6. Result shows that the CSO algorithm saves computation overhead from existing bio-inspired algorithms like GA, PSO, and SCSO. Computation time savings of the proposed algorithm for VOPD,
TABLE 3. Communication cost for square mesh.

| Benchmark   | ILP | NMAP | GA | PSO | CastNet | SBMAP | BEMAP | CSO | Proposed Algo. (CSO) |
|-------------|-----|------|----|-----|---------|-------|-------|-----|---------------------|
| VOPD        | 4119| 4265 | 4218| 4119| 4135    | 4125  | 4119  | 4119| 4119                |
| MPEG4       | 3567| 3672 | 3772| 3567| 3852    | 4474  | 3772.5| 3567| 3567                |
| PIP         | 640 | 704  | 704 | 640 | 640     | 640   | 640   | 640 | 640                 |
| MWD         | 1120| 1312 | 1321| 1120| 1280    | 1280  | 1280  | 1122| 1120                |
| MMS         | 66752| 66752| 66752| 66752| 66752   | 66752 | 66752 | 66752| 66752               |
| 802.11ax     | 16185.6| 16185.6| 16185.6| 16185.6| 16185.6| 16185.6| 16185.6| 16185.6| 16185.6          |
| CAVLC        | 6971 | 6971 | 6971 | 6971 | 6719    | 6719  | 6719  | 6719| 6719                |
| DVOPD        | 9908 | 9908 | 9908 | 9908 | 9602    | 9602  | 9602  | 9602| 9602                |
| Telecom      | 14834| 14834| 14834| 14834| 15478   | 15050 | 13054 | 13054| 13054               |
| 263encMP3dec | 230470| 230470| 230470| 230470| 230470  | 230470| 230470| 230470| 230470             |
| MP3encMP3dec | 17021 | 18041 | 17133 | 17021 | 17021   | 17021 | 17021 | 17021| 17021              |
| 26decMP3dec  | 19823 | 19843 | 19911 | 19823 | 19823   | 19823 | 19823 | 19823| 19823              |

TABLE 4. CSO saving in communication cost for square mesh.

| Benchmark   | Percentage saving of CSO for square Mesh |
|-------------|-----------------------------------------|
| NMAP        | 10                                      |
| GA          | 10                                      |
| CastNet     | 0.39                                    |
| SBMAP       | –                                       |
| BEMAP       | –                                       |
| PIP         | 2.94                                    |
| MPEG4       | 5.57                                    |
| MWD         | 17.14                                   |
| VOPD        | 2.40                                    |
| MMS         | 2.30                                    |
| 802.11ax    | 27.10                                   |
| CAVLC       | 3.72                                    |
| DVOPD       | 4.05                                    |
| Telecom     | 13.64                                   |
| 263encMP3dec| 47.71                                   |
| MP3encMP3dec| 0.66                                    |
| 26decMP3dec | 0.10                                    |

CSO Average % saving in Communication Cost

| Over NMAP | Over GA | Over CastNet | Over SBMAP | Over BEMAP | Over PSO | Over CSO |
|-----------|---------|--------------|------------|------------|----------|----------|
| 11.02%    | 5.33%   | 4.28%        | 5.95%      | 2.16%      | 0.30%    | 0.03%    |

MPEG4, MWD, 263encMP3dec, MP3encMP3dec and 26decMP3dec are 121.75%, 169.5%, 202.65%, 242.47%, 172.99%, and 223.88% respectively, in comparison with GA. Savings of CSO in computation time with respect to PSO are 113.84%, 188.75%, 203.72%, 242.47%, 172.99%, and 225.31% for above mentioned applications. Likewise, the computation time savings of CSO over SCSO are 26.04%, 67.5%, 76.64%, 64.19%, 52.56% and 54.18% respectively for these standard benchmarks.

B. POWER AND ENERGY

The power and energy estimation of the proposed algorithm is carried out using (6) and (7). Standard CMOS cell library embedded in NoCtweak is used to compute network latency, throughput, power, and energy expenditure of NoC network for the CSO algorithm. NoCtweak uses the standard CMOS cell data to compute power utilization and estimation of other performance metrics for NoC. Register Transfer Level designs of all components of the router in Verilog were synthesized with Synopsys Design Compiler in NoCtweak. Cadence SoC Encounter uses the CMOS cell library to place and route RTL Designs. NoCtweak uses enfolded post-layout data of these designs for performance assessment of NoC systems [39]. The performance at certain traffic is estimated according to the activities of these elements in the network.
Usually, the router’s energy and power consumption is estimated on the active percentage of router components while routing the packets on the network. The power consumption of the network increases as the packets travels longer in the network. The proposed CSO algorithm provides optimum placement of tasks in the network, which reduces packet traversed paths. Hence, it provides solutions with optimal performance in power and energy consumption.

Power estimation of NMAP, CastNet, SBMAP, and BEMAP, along with the proposed algorithm for standard benchmarks normalized with NMAP, are shown in Figure 3. Results indicate that the CSO performs better than the mentioned algorithms. On average, CSO consumes 24.87% less power than NMAP, 20.15% less than CastNet, 8.69% less than SBMAP, and 10.76% less than BEMAP.

Similarly, CSO provides better results in terms of network energy. Figure 4 represents the energy comparison of the CSO for eight standard benchmarks. We have found, on average, the energy savings of 18.94% over NMAP, 13.58% over CastNet, 2.40% over SBMAP, and 4.77% over BEMAP by the proposed algorithm.

C. LATENCY AND THROUGHPUT

Latency is the time required in transmitting a packet from the initiation of a packet at the source node to its complete reception at the destination node. It has a dominant effect on network traffic and is also considered to be an important constraint in this research. Equation (4) embedded in NoCtweak is utilized for latency estimation. The average network latency of PIP, VOPD, 263encMP3dec, and MP3encMP3dec normalized with NMAP is represented in Figure 5. On average, the proposed algorithm’s improvement in latency is 1.18%, 0.66%, 1.02%, 0.95% compared to NMAP, CastNet, SBMAP, and BEMAP, respectively.
The maximum traffic received by a network in a unit time is known to be throughput. A good mapping results in high throughput and maximizes the performance of the network. Throughput of the network is calculated using (5) with NoCtweak. We have compared the throughput of the proposed algorithm with existing heuristics for standard applications with different injection rates. Figure 6a-b represent the performance of throughput for different flit injection rates in the network. The results show that the proposed algorithm gives better throughput than NMAP, CastNet, SBMAP, and BEMAP at a higher traffic injection rate in most cases.

VI. CONCLUSION
In this paper, we have proposed a Cuckoo Search Optimization algorithm for the application mapping problem on the NoC systems. CSO algorithm is inspired by obligate brood parasitism of some cuckoo species by laying their eggs in the nests of other host birds and the Levy Flight behavior of some birds and fruit flies. In this technique, the initial population is generated with a Greedy random algorithm which is then optimized iteratively by applying Cuckoo Search via Lévy flight to generate a new population to achieve optimal fitness value. The proposed algorithm optimally maps the embedded...
application benchmarks on defined 2D mesh NoC structures. CSO provides the lowest communication cost against each benchmark, like ILP based exact mapping technique. The comparative analysis revealed that the proposed CSO beats the other bio-inspired heuristic algorithms like GA, PSO, and SCSO with minimum communication cost at low computational overhead.

Furthermore, the performance analysis indicates that the CSO delivers better results in communication cost, power, energy consumption, latency, and throughput than the existing heuristic algorithms, i.e., NMAP, CastNet, SBMAP, and BEMAP. In addition, we have also assessed the proposed algorithm for application mapping on the NoC platform, with reduced mesh size (rectangular mesh) for few embedded application benchmarks. The results indicate that the proposed algorithm shows optimality on existing heuristic-based algorithms.

In the proposed algorithm, heavily communicating tasks are mapped in close vicinity in the structure, which may introduce congestion in the network depending on the communication pattern and volume of data exchange among the tasks in an application. In future consideration, this shortcoming can be addressed by using an adaptive dead-lock free routing algorithm and proposed mapping methodology. Moreover, the proposed mapping approach can be extended to 3D and wireless NoC systems.

REFERENCES

[1] 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. [Online]. Available: https://ieeexplore.ieee.org/document/9044826/

[2] C. L. H. J. Choudhury, “A survey on application mapping for network-on-chip systems,” J. Syst. Archit., vol. 64, no. 3, pp. 394–405, May 2018.

[3] W. Amin, F. Hussain, S. Anjum, Z. Nain, and S. W. Kim, “NodeGuard: A reliable network-on-chip router architecture,” Electron. Lett., vol. 9, no. 2, p. 342, Feb. 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/2/342

[4] N. K. Baloch, M. I. Baig, and M. Danesh talab, “Defender: A low overhead and efficient fault-tolerant mechanism for reliable on-chip router,” IEEE Access, vol. 7, pp. 142843–142854, 2019.

[5] M. Rashid, N. K. Baloch, M. A. Shahid, F. Hussain, S. Saleem, Y. B. Zikria, and S. W. Kim, “Fault-tolerant network-on-chip router architecture design for heterogeneous computing systems in the context of Internet of Things,” Sensors, vol. 20, no. 18, p. 5355, Sep. 2020. [Online]. Available: https://www.mdpi.com/2412-8220/2018/18/5355

[6] M. Ibrahim, N. K. Baloch, S. Anjum, Y. B. Zikria, and S. W. Kim, “An energy efficient and low overhead fault mitigation technique for Internet of Thing edge devices reliable on-chip communication,” Softw. Pract. Exper., early access, pp. 1–18, Feb. 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/spe.2796
doi: 10.1002/spe.2796.

[7] R. Marculescu, U. Y. Ogras, L.-S. Peh, N. E. Jerger, and Y. Hoskote, “Outstanding research problems in NoC design: System, microarchitecture, and circuit perspectives,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 28, no. 1, pp. 3–21, Jan. 2009. [Online]. Available: http://ieeexplore.ieee.org/document/4723644/

[8] 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, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762112000092

[9] J. H. Holland, “Outline for a logical theory of adaptive systems,” J. Assoc. Comput. Mach., vol. 9, no. 3, pp. 297–314, 1962, doi: 10.1145/321127.321128.

[10] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in Proc. IEEE ICNN, vol. 4, Nov./Dec. 1995, pp. 1942–1948.
