

## Article

# An Optimized Nature-Inspired Metaheuristic Algorithm for Application Mapping in 2D-NoC

Saleha Sikandar <sup>1</sup>, Naveed Khan Baloch <sup>1</sup>, Fawad Hussain <sup>1</sup>, Waqar Amin <sup>1</sup>, Yousaf Bin Zikria <sup>2,\*</sup>  
and Heejung Yu <sup>3,\*</sup>

<sup>1</sup> Computer Engineering Department, University of Engineering and Technology, Taxila 47050, Pakistan; salu.malik16@gmail.com (S.S.); naveed.khan@uettaxila.edu.pk (N.K.B.); fawad.hussain@uettaxila.edu.pk (F.H.); waqar.ameen45@gmail.com (W.A.)

<sup>2</sup> Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, Korea

<sup>3</sup> Department of Electronics and Information Engineering, Korea University, Sejong 30019, Korea

\* Correspondence: yousafbinzikria@ynu.ac.kr (Y.B.Z.); heejungyu@korea.ac.kr (H.Y.)

**Abstract:** Mapping application task graphs on intellectual property (IP) cores into network-on-chip (NoC) is a non-deterministic polynomial-time hard problem. The evolution of network performance mainly depends on an effective and efficient mapping technique and the optimization of performance and cost metrics. These metrics mainly include power, reliability, area, thermal distribution and delay. A state-of-the-art mapping technique for NoC is introduced with the name of sailfish optimization algorithm (SFOA). The proposed algorithm minimizes the power dissipation of NoC via an empirical base applying a shared  $k$ -nearest neighbor clustering approach, and it gives quicker mapping over six considered standard benchmarks. The experimental results indicate that the proposed techniques outperform other existing nature-inspired metaheuristic approaches, especially in large application task graphs.

**Keywords:** network-on-chip; sailfish hunting; metaheuristic optimization



**Citation:** Sikandar, S.; Baloch, N.K.; Hussain, F.; Amin, W.; Zikria, Y.B.; Yu, H. An Optimized Nature-Inspired Metaheuristic Algorithm for Application Mapping in 2D-NoC. *Sensors* **2021**, *21*, 5102. <https://doi.org/10.3390/s21155102>

Academic Editor: Francisco J. González-Castaño

Received: 14 July 2021  
Accepted: 26 July 2021  
Published: 28 July 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

The overall performance and scalability of the system-on-chip (SoC) are degraded because of the increasing number of intellectual property (IP) cores embedding on the SoC. For the improvement of overall performance and flexibility of the SoC, new promising solutions have been proposed, and they are called network-on-chip (NoC) [1]. NoC is an on-chip, packet-based communication switching network which is created for interaction between IP cores of the SoC designs [2]. Routers (switch fabric) are linked in some standard topology for communications among IP cores. A router is available for every IP core in an NoC. The router is a basic building block of the NoC architecture; a fault-resilient router architecture is necessary for reliable on-chip communication. The authors of [3–6] did some architectural modifications in the existing NoC routers designs to propose a reliable on-chip network communication infrastructure. A message passing technique is used for the exchange of data between IP cores. As per the multi-core system principle, the contribution of NoC in power consumption of the total system is around 40%, and this has a vital role in network performance [1,7]. The power, latency and area of NoC-based systems are conspicuously impacted by the selection of an on-chip interconnection architecture [7]. Depending on the interconnection networks, numerous standard topologies are established for the NoC. The most renowned topology out of all prevailing conventional topologies of the NoC architecture is a mesh topology [8].

In the mesh topology, there are short paths for communication between IP cores and high bisection width. The interconnected structure is regular and fixed, and the links are of equal size. Considering this context, various techniques for applications mapping have been proposed using search-based and exact optimization methods. Additionally, proper

modeling via an analytical approach has been investigated to reduce the area, latency and power in NoCs.

Because computation time to solve the mapping problem increases with the size of the application to be mapped, it is known that an application mapping is a non-deterministic polynomial-time (NP)-hard problem. To obtain the optimal solution over NoC performance metrics, search-based optimization techniques have been considered. Therefore, the solution of NP-hard problems is significantly dependent on the choice of the best heuristic or metaheuristic technique.

In practical systems, resources are limited so that an efficient utilization of given resources is a critical issue. Optimization techniques can be employed in a wide range of areas, including engineering, finance, resource planning and Internet routing. Using a mathematical model of the social and political progression, metaheuristic algorithms provide an effective algorithm to solve the given optimization problems. These algorithms can obtain a universal solution by facilitating interaction between high level approaches and local improvement methods.

Furthermore, a metaheuristic algorithm can be efficient if it offers a realistic equilibrium between experimentation and exploitation on a provided optimization problem, which is critical. Intensification (i.e., exploitation) is associated with local search, while diversification (i.e., exploration) is associated with global search. Diversification tends to find out diverse solutions globally (i.e., global search). On the other hand, intensification focuses on searching local regions with the knowledge of the current best solution from this region (i.e., local search). There is no initial solution required for global search, while local search starts from an initial candidate solution. The mobility of candidate solutions should be randomized as far as possible during the exploration phase. On the other hand, the exploitation process entails thorough investigation of the promising area(s). The most dominant difference between current metaheuristic algorithms, in general, is how they balance the discovery and exploitation phases. Depending upon the context mentioned above, sailfish optimization (SFO) is considered in this study.

SFO provides a suitable equilibrium between intensification (exploitation) and diversification (exploration) to avoid early convergence. To examine the performance metrics of NoC, the novel metaheuristic optimization algorithm used in this paper, that is, SFO, is described in [9]. The SFO algorithm is modeled after a sailfish group targeting a school of sardine prey in a series of attacks. To begin, SFO uses two assortments of prey and predator species to replicate the technique of group hunting. Second, the presented algorithm breaks down the mutual security of grouping prey by alternating attacks. Third, prey mobility can be changed across the search region, allowing the hunter to capture the right prey and improve its fitness. The effectiveness of the SFO algorithm is verified by examining the optimal mapping for eight NoC benchmarks for the two-dimensional (2D) mesh topology.

The remainder of the paper is structured as follows. The related work is given in Section 2. The inspiration for the sailfish optimization algorithm is described in Section 3. The mapping using SFO, models used for the analysis of metrics and the proposed algorithm are described in Sections 4–6, respectively. The experimental setup along with considered benchmarks and results are summarized and analyzed in Section 7. Section 8 ends with some conclusive remarks.

## 2. Related Work

In [10], Araki and Yoshihiro presented a multi-path reliable distance-vector routing strategy by utilizing multiple paths for the extension of reliable distance-vector routing (RDV) for the improvement of communication performance, decreased delivery delay, higher load-balancing and more substantial network capacity. In comparison to RDV, fault tolerance is also greater against the topology modifications. In [4], Rashid et al. proposed a reliable on-chip network communication architecture by making some architectural improvements in the existing NoC routers' designs. In [11], a router's controllers design

based on finite-state machine (FSM) is presented for the minimization of error propagation, aiming at low utilization of logical resources.

In [12], Wu and Cai presented a Fibonacci tree optimization strategy (FTOS) for the scheduling query of wireless sensor networks. The proposed algorithm provided less energy consumption and optimization of detection efficiency. In [13], Rhee et al. presented an artificial neural network (ANN) model combined with the genetic algorithm (GA) for the cost-effective operation of a silo. The combined technique gave the optimized results with the improvement in the accuracy of internal level prediction of the silo, and an efficient number of sensors and their positions of installation are determined. In [14], the authors presented a comprehensive overview of the algorithms of machine learning for embedded systems and mobile computing space. In [15], the authors presented a heuristic technique based on the moth-flame optimization (MFO) algorithm for resolving the weak exploration problem of the k-means data clustering algorithm.

The problem of application mapping has stimulated the research community because of the expeditious growth in NoC. Tosun et al. proposed integer linear programming (ILP) as an exact mapping method for the mesh-based two-dimensional NoC with an energy minimization principle in [8]. In [16], Hu and Marculescu presented a branch and bound (BB) mapping solution for the topological allocation of IP cores on a NoC platform for the minimization of the total consumption of energy with the limitation of bandwidth of the link. In [17], Lei et al. presented a two-step genetic algorithm (GA) based on delay for the communication of NoC. The prime function for the scheduling and mapping of IPs was the minimization of overall execution time. Murali and Micheli proposed a heuristic approach based on a mapping algorithm for cores mapping on 2D mesh topology with the restraint of bandwidth reservation in [18]. In [19], Lu et al. presented a clustering algorithm based on simulated annealing for reducing the simulation time of an annealing process of a large system. The process of clustering compromised the optimum results but accelerated the computation time. In [20], Radu and Vintan proposed an optimized simulated annealing (OSA) algorithm for 2D mesh mapping by optimizing the parameters of the annealing process for producing the optimum outcomes with less time than the conventional simulated annealing schemes. Ascia et al. [21] presented a multi-objective GA for mapping of IP cores in a 2D mesh topology for optimizing the power consumption and network performance. In [22], Jena and Sharma presented a heuristics search based multi-objective GA for the mapping of IP cores on a 2D mesh topology for the optimization of link bandwidth, the performance of the network and power dissipation. Sepulveda et al. also presented a multi-objective adaptive immune algorithm (MAIA) for the problem of application mapping of NoC architecture [23]. In [24], Harmanani and Farah proposed an algorithm for assigning tasks to the nodes of a 2D mesh network based on simulated annealing. Hu et al. proposed a task mapping technique for the NoC architecture with a constraint of bandwidth [25]. This technique was energy aware and expedited the run-time of the process of task mapping, but it shows trade-off in the network performance results.

Ye et al. derived the power models for connectivity wires, switch and inbuilt buffer in [26]. In [27], the authors provided a well-accepted mathematical term for 2D NoC interconnect energy models. Kahng et al. [28] and Ost et al. [29] created a practical power model for 2D NoC as a follow-up to the one in [27]. In [28], the power model takes into account architecture-level power as well as region modeling and router capacity for the router. The power modeling in [28] was validated and checked by Ost et al. [29]. The authors of [30] calculated the efficiency of mesh-dependent 2D and 3D NoCs based on the comprehension of energy depletion between the cores and the routing area. The thesis by Sahu and Chattopadhyay [31] takes advantage of a comprehensive review of framework mapping techniques for NoC and examines various mapping methods proposed during the last period. As per Sahu and Chattopadhyay [31], a heuristic-based mapping strategy provided a better end result in terms of network output metrics optimization.

In [32], a simulated annealing (SA) algorithm is implemented as a metaheuristic approach to create an efficient mapping with IP connectivity specifications as a restriction

for 2D NoC. The authors of [33] implemented mapping by scheduling with an ant colony optimization (ACO) approach for 2D NoC. In [34], a particle swarm optimization (PSO) is used as a mapping technique on both 2D and 3D NoCs, with the connectivity metric as the objective function. To tackle the problem addressed in [32], a power-aware mapping technique for 2D NoC utilizing SA with the taboo quest (SAT) was proposed by Alagarsamy and Gopalakrishnan [35]. In [36], a mapping technique for a 2D NoC is presented. The foremost objective is to build a chain of linked cores that can be used to construct a new mapping system. In comparison to similar ones, the authors of [36] attempted to use less bandwidth. In [37], Tosun presented a heuristic approach for a mesh 2D NoC in which a priority list based on overall and average communication bandwidth was established.

In [38], a reliability-aware technique is presented. The featured graph is divided into two sub-graphs, which are used to reduce transmission flow. As a result, transmission flow between the two sub-graphs is reduced, while traffic within every graph increases. Niknam and Amiri presented a novel hybrid PSO-based approach to address the clustering issue in [39]. For better performance, ACO and  $k$ -means techniques were used. The presented approach was tested and validated on various publicly available datasets, and the preliminary observations are optimistic. The suggested hybrid approach was shown to coincide with an optimal solution in the majority of instances. Junior et al. [40] also presented an ACO-based approach for finding and maximizing directions in a mesh-based NoC. Routed optimization was achieved by reducing the total delay in packet transmission between activities. The visionary conclusions showed the efficiency of the ACO-based technique. In addition, Xie et al. proposed an online mapping protocol to refine task mapping methodology for minimizing connection power consumption [41]. First, the run-time interconnection point of applications was investigated. Secondly, this method measured the mapping assignment and used real-time web mapping.

### 3. Sailfish Optimizer

In this section, the key inspiration for the SFO algorithm (SFOA) is discussed. The suggested algorithm and mathematical models are then thoroughly explained.

#### *Inspiration*

Shadravan et al. [9] recently introduced a new metaheuristic technique called SFO, which incorporates the action of both a predatory group of sailfish and a prey group of sardines. The sailfish is known as a social predator since it attacks and catches its prey in groups. Predators use various killing techniques in cooperative hunting. The class of sailfish, for example, is distinguished by the alternation of attack techniques. It entails that each member of the group attacks the school of prey (sardine) alone at a given time, injuring or hunting some of them while the other group members conserve their strength. Whenever a sailfish attacks a school of prey, it will update its location concerning them. Furthermore, the sailfish will update their location to occupy vacant space around the prey school and imitate circling the prey. When a member of the sardine group (prey) is wounded, the sardine group changes direction to avoid the sailfish's subsequent attacks. The general procedure of the sailfish optimizer algorithm is defined in the subsections that follow.

Group hunting is an intriguing illustration of collective activity in communities of invertebrates, fishes, birds and mammals. Compared to hunting alone, predators do not require a lot of power to kill their prey while hunting in groups.

Predators in the most basic type of group hunting aim to finish off the prey by step-by-step planning of the attack, whereas predators under the more sophisticated class of group hunting practice specialized positions to mob and capture the prey [42]. The alternation of attacks is one of the most complicated group hunting techniques. This tactic allows the hunter to save strength when other predators are injuring the prey. Sailfish hunting in groups that alternate attacks on the schooling sardines is an illustration of this kind of method [43,44].

The most expeditious fish in the ocean, sailfish can attain speeds up to 62 miles per hour. They hunt in clusters, herding schools of smaller fish, such as sardines, near the surface. Sailfish find the sardines' mobility and speed during the assault very difficult. The sailfish either slashes multiple sardines with its rostrum or taps a single sardine, causing it to become unstable. Sardines cannot float quickly enough to dodge the tip of the sailfish's rostrum and are incapable of responding to this community hunting because the sailfish has one of the fastest accelerations ever observed in a floating creature. According to sardine experimental action, wounded sardines would be isolated from the prey shoal and unable to travel with the shoal, resulting in their capture by the sailfish [42].

The majority of sailfish attacks do not result in sardine deaths, and only a small percentage of sardines are directly caught. However, as sailfish attacks become more common, an increasing number of sardines are injured. Animals who hunt in groups, such as wolves, are more likely to engage in this form of hunting. On the other hand, these sailfish parties split up and regroup with new affiliates daily. During an assault, a sailfish preserves its big back flipper and sacral flippers upright to maintain its body strength. Often, right before an attack, they transform their body color from the usually bluish-silver parallel edges deepening to nearly black. The purpose for the color change is unclear, but it appears to be a form of communication between sailfish [42]. Sailfish use shifts in their body to signal which should move first, allowing them to avoid being injured by a companion. The attack-alternation technique of sailfish party hunting is the key inspiration for the SFO algorithm. The natural actions of sailfish and sardines are mathematically represented in the following subsection, and an optimization approach based on this mathematical model is developed.

## 4. Mapping Using SFOA

### 4.1. Problem Formulation

An application is characterized by a directed graph of the network in NoC, which is later scheduled by the scheduler using another directed core graph of the network on the existing IP-cores. The directed core graph is transmuted and depicted via an effective mapping method on the NoC topological architecture using an architecture graph.

**Definition 1. Directed Task Graph (DTG):** The task graph of the network is a directed acyclic graph  $DTG(P, E)$ , where every node of the graph symbolizes a task of the computational process of the application. In addition, the directed edges or links represent the communication or data volume among the tasks communicating.

$$DTG(P, E) \quad (1)$$

where  $P$  and  $E$  are the sets of nodes, which correspond to the processes or tasks, and links or edges, respectively, and  $p_i \in P, e_{i,j} \in E$  for  $i, j = 1, 2, 3, \dots$ .

**Definition 2. Directed Core Graph (DCG):** The core graph of the NoC architecture is a directed graph  $DCG(C, D)$ , where every node of the graph symbolizes the IP cores in the topology. The directed edges represents the direct communication among the nodes (i.e., IP cores,  $d_i$  and  $d_j$ ).

$$DCG(C, D) \quad (2)$$

where  $C$  is the set of IP cores or processing elements and  $D$  denotes the set of links or edges with communication directions in the architecture graph. Elements in  $C$  and  $D$  are defined as  $c_i \in C$  and  $d_{i,j} \in D$  for  $i, j = 1, 2, 3, \dots$ .

### 4.2. SFOA for NoC Mapping

The initial sailfish and sardine populations are generated using the initial mapping and weight of the task graph given at time  $t = 0$ . Considering the settings of parameters of the proposed algorithm, the fitness value, which is the communication cost (CC) of the best sailfish (i.e., mapping solution), is computed. (For CC, refer to Equation (8) which

is defined in Section 5). Later, the positions of sailfish and sardine are updated in consideration of attack power (AP). (For position updates of sailfish and sardine, refer to Equations (21) and (27), respectively. For AP, refer to Equation (24) in Section 5). After updating the positions, the optimized result of mapping (sailfish) can be obtained.

#### 4.3. Parameters Setting for SFOA

The proposed algorithm requires the setting of a few basic parameters to verify the efficiency of group hunting. In the proposed algorithm, the fitness function under consideration is the cost for communication, which is denoted by CC. The population size is 300, the number of iteration is equal to 150 and pp is the rate between the sailfish and sardine (where pp is defined as the fraction of the sardine population which forms the initial sailfish population), which is set to 0.1; these values are set for the application mapping on 2D NoC. These values are set based on the number of iterations run and optimization acquired for deducing an optimal solution. They also differ as per the properties of the application considered for mapping.

For the analysis of the performance parameters of an NoC such as energy, power and communication cost computation along with latency and average throughput, two models are used in this work. These two models are named the Bit Energy model and CMOS cell library model, and their mathematical expressions are explained in detail and in the next section.

### 5. Models Used for Analysis of Metrics

For analyzing the performance metrics of an NoC, two models are considered in the presented work [25,43]. An effective trade-off between the faster mapping over 2D mesh and performance metrics of NoC is presented by SFOA in this study.

#### 5.1. Bit Energy Model

For the estimation of consumption of power of the router in the network, an energy model [25] is considered as follows:

$$E_B = E_{SB} + E_{LB}, \quad (3)$$

where  $E_B$  is the energy used up for transferring 1 bit of data from the source node to the destination node, which comprises the energy of the switch ( $E_{SB}$ ) and energy of the link ( $E_{LB}$ ) of the NoC network. The average network energy consumption  $E_{B(p_i,p_j)}$  for transferring 1 bit of data from a source node  $p_i$  to the destination node  $p_j$  is calculated by the following equation:

$$E_{B(p_i,p_j)} = H_{count} \times E_{SB} + (H_{count} - 1) \times E_{LB}, \quad (4)$$

where  $H_{count}$  is the Manhattan distance between the source node  $(a_i, a_j)$  and the destination node  $(b_i, b_j)$ , which is obtained by

$$H_{count} = |a_i - b_i| + |a_j - b_j|. \quad (5)$$

Therefore, the total energy consumption of the network ( $E_T$ ) is calculated by using the average network energy and the link bandwidth,  $BW_{(p_i,p_j)}$ , between nodes  $p_i$  and  $p_j$ .

$$E_T = \sum_{i,j} (E_{B(p_i,p_j)} \times BW_{(p_i,p_j)}) \quad (6)$$

Substituting Equation (4) into Equation (6),  $E_T$  can be rewritten by

$$E_T = \sum_{i,j} \left[ (H_{count} \times E_{SB} + (H_{count} - 1) \times E_{LB}) \times BW_{(p_i,p_j)} \right]. \quad (7)$$

Moreover, the cost of communication is defined by

$$CC = \sum_{i,j} H_{count} \times BW_{(p_i,p_j)}. \quad (8)$$

Different mapping results generate different energy and cost values. The prime concern is to obtain a mapping function that provides minimal cost for the whole network. The communication cost of the applications of NoC is considered the performance measure for distinct applications in this research work.

### 5.2. CMOS Cell Library Model

The proposed SFO algorithm utilizes the standard CMOS cell library model [43] for the calculation of network power, latency, energy consumption of packets and throughput of an NoC system. For the computation of average latency of the network via this model, the following equation is used:

$$Lat_{avg} = \frac{1}{N} \sum_{i=1}^N \left( \frac{1}{N_i} \sum_{k=1}^{N_i} (Lat_{(i,j)}) \right), \quad (9)$$

where  $N$  is the total number of processor or cores in the network,  $N_i$  is the total numbers of received packets by the core  $i$  and  $Lat_{(i,j)}$  is the latency of packet  $j$  at destination node  $i$ .

The average throughput of the network,  $TP_{avg}$ , is evaluated as follows:

$$TP_{avg} = \frac{1}{N(T_S - T_W)} \sum_{i=1}^N N_i, \quad (10)$$

where  $T_W$  is the warm-up time of the simulation and  $T_S$  is the simulation time.

The network average power,  $P_{N_{avg}}$ , is computed by

$$P_{N_{avg}} = \frac{1}{N} \sum_{i=1}^N \sum_{k=1}^{N_i} \left[ \alpha_{(i,k)} P_{N(act,k)} + (1 - \alpha_{(i,k)}) P_{N(inact,k)} \right] \quad (11)$$

where  $\alpha_{(i,k)}$  is the active probability of component  $k$  in router  $i$  after  $T_W$ . Moreover,  $P_{N(act,k)}$  and  $P_{N(inact,k)}$  are the post-layout active and inactive power of the component  $k$ .

Finally, the network average energy consumption by every packet is given by

$$E_{P_{avg}} = \frac{T_S - T_W}{NN_{pack}} \sum_{i=1}^N \sum_{k=1}^{N_i} \left[ \alpha_{(i,k)} P_{N(act,k)} + (1 - \alpha_{(i,k)}) P_{N(inact,k)} \right] \quad (12)$$

where  $N$  is the total number of cores available in the network.  $N_{pack} (= \sum_{i=1}^N N_i)$  is the total number of packets injected in the network. For a certain number of experiments,  $N$  remains the same, and  $N_{pack}$  can be changed by increasing or decreasing the packet injection rate.

## 6. The Proposed Algorithm: SFOA

The proposed SFOA takes the inputs, directed task graph,  $DCG$ , and directed network graph,  $DNG$ , and effectively performs the mapping of the task onto the cores of the 2D NoC topological architecture.

### 6.1. Empirical Base for Initial Mapping

To create the empirical base for the initial mapping, the following five steps of the self-adaptive chicken swarm optimization (SCSO) algorithm [44] are considered. Furthermore, Figure 1 shows the flowchart for initial mapping procedure.

- Step 1: From DCG, randomly select the IP-Core

$$Rand(c_i), \text{ for } c_i \in C \quad (13)$$

- Step 2: Use the DC matrix to find the presence of direct connection of the selected core with each core.

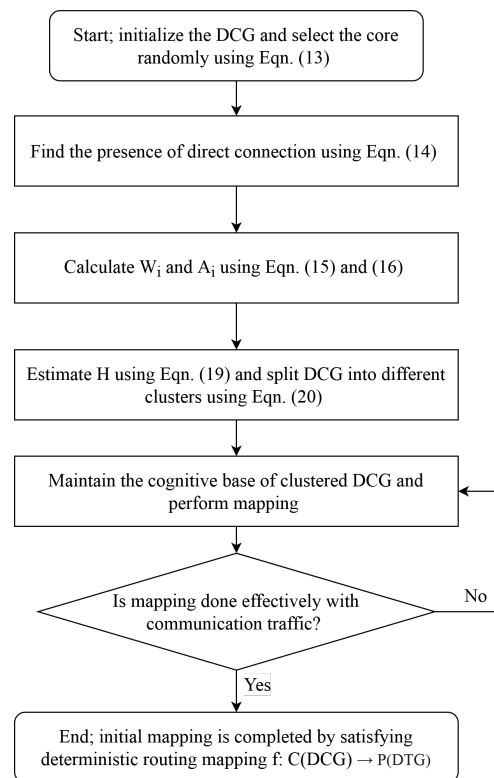
$$DC = \begin{cases} 1 & ; \text{ if } (c_i, c_j) = d_{ij} \in D \\ 0 & ; \text{ otherwise} \end{cases} \quad (14)$$

- Step 3: Calculate the average CC ( $A_i$ ) and weight ( $W_i$ ) for each core ( $c_i$ ) as follows:

$$W_i = \sum_{d_{ij} \in D} w_{ij} \quad (15)$$

$$A_i = \sum_{d_{ij} \in D} \frac{w_{ij}}{|N(c_i)|}, \quad (16)$$

where  $w_{ij}$  is the weight between cores  $c_i$  and  $c_j$  and  $N(c_i)$  is the open neighborhood of  $c_i$ .



**Figure 1.** Flowchart for initial mapping.

For the identification of neighbors, use the following equation:

$$N(c_i) = \{c_j \in C \mid (c_i, c_j) = d_{ij} \in D\} \quad (17)$$

- Step 4: For the identification of hop counts among the source node  $c_i$  and sink node  $c_j$ , use the following matrix:

$$H = [H_{ij}] \quad (18)$$

where  $[H_{ij}]$  means that  $(i, j)$  element of matrix  $H$  is given by  $H_{ij}$ . Matrix  $H$  indicates the minimum probable links for communication between the source and sink nodes. Considering  $d(c_i, c_j)$  is the shortest path between the cores  $c_i$  and  $c_j$ ,  $N(c_i, c_j)$  is the number of hops in the shortest path.

$$H_{ij} = \min(N(c_i, c_j)) \quad (19)$$



- Step 5: Using the shared  $K$ -nearest neighbor clustering approach, form a diverse cluster. If  $c_i$  and  $c_j$  have each other in their closest  $K$ -nearest neighbors list, then an edge exists between them. The strength of this edge is evaluated using:

$$str(c_i, c_j) = \sum (K + 1 - o) \times (K + 1 - p) \tag{20}$$

where  $K$  is the size of the neighbor’s list,  $o$  is the position of shared near-neighbor in  $c_i$  list and  $p$  is the position of shared near-neighbor in  $c_j$  list. Hence,  $c_{io}$ , i.e., the shared near-neighbor in  $c_i$  list, is equal to  $c_{jp}$ , that is the shared near-neighbor in  $c_j$  list.

After Step 5, an empirical base is created with clustered DCG. Figures 2–4 show the standard NoC video object plane decoder (VOPD) benchmark, clustering of VOPD task graph and its initial mapping on a  $4 \times 4$  mesh, respectively.

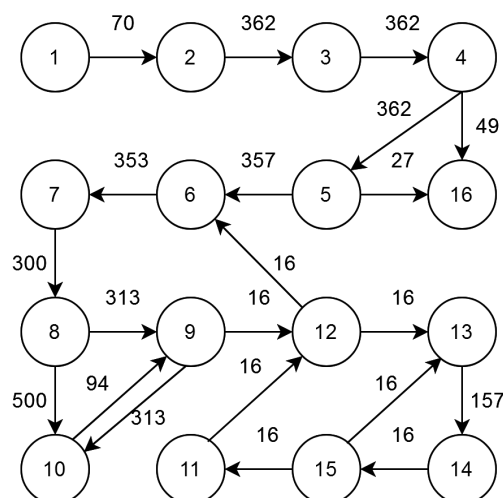


Figure 2. Standard NoC VOPD benchmark.

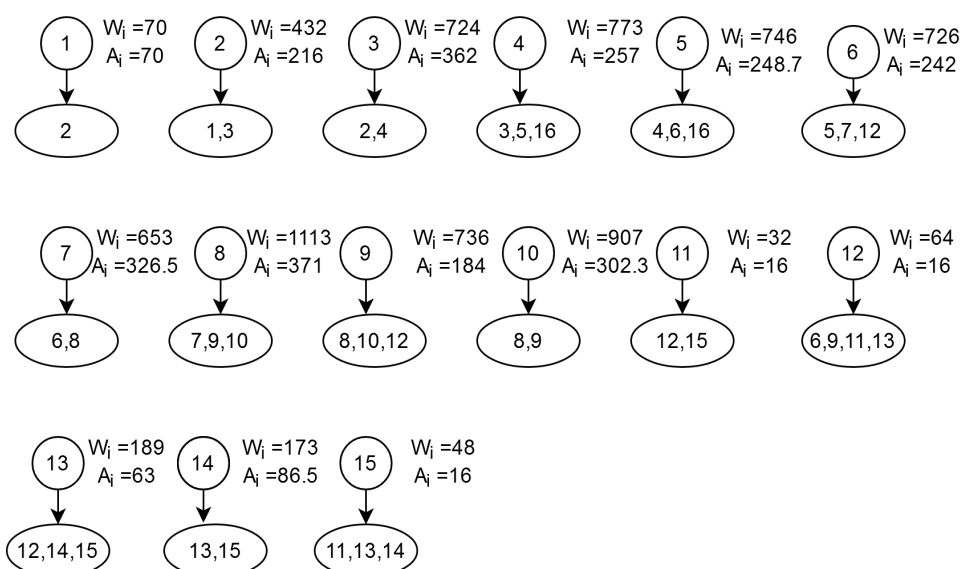
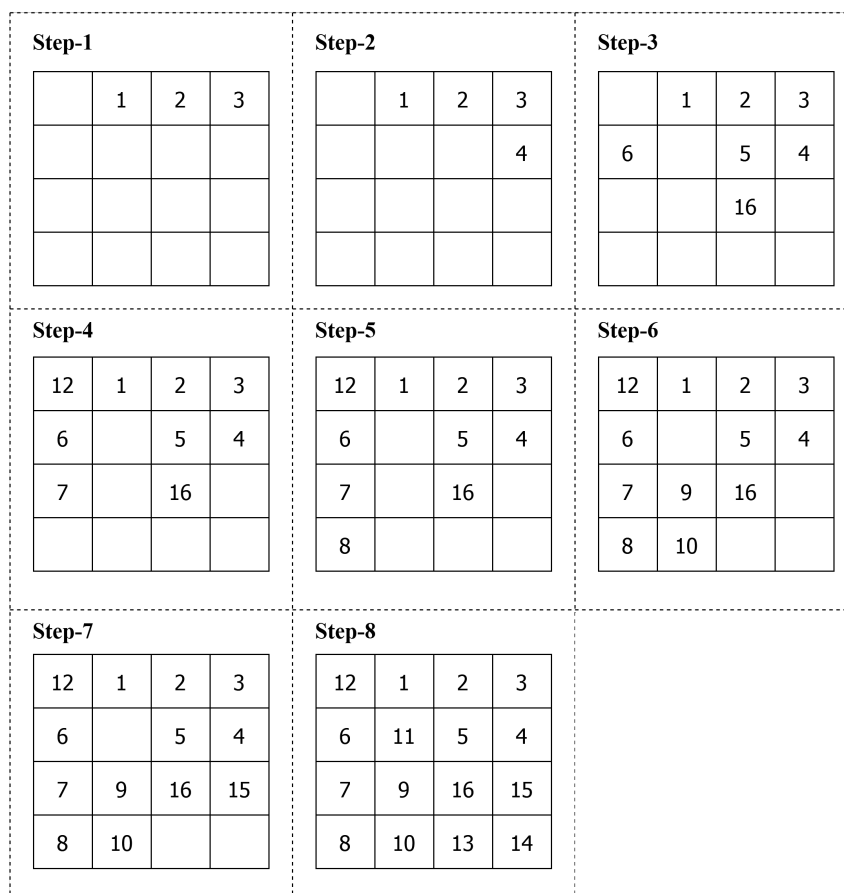


Figure 3. Clustering of VOPD task graph.



**Figure 4.** Initial mapping of VOPD task graph on  $4 \times 4$  mesh.

### 6.2. Video Object Plane Decoder

Video object plane decoder (VOPD) is an application comprising several sub-tasks: run-length decoder, downsampler, quantizer, etc. These sub-tasks require communication among themselves at the rates specified in MBs on the edges between them. Figure 5 represents the architectural diagram of the VOPD, while Figure 2 illustrates the graphical representation of the VOPD tasks. VOPD consists of 16 sub-tasks having 21 edges labeled with distinct communication bandwidth.

For the initial phase of mapping, a random procedure is adopted as a mapping strategy. The outcome of this initial mapping is considered the input for the proposed SFOA to minimize the consumption of power and communication cost of 2D NoC. Figure 6 represents the flowchart for SFOA.

### 6.3. SFOA Algorithm

#### 6.3.1. Initialization

The first step of SFOA comprises initialization of the sailfish and sardine populations. The population generation/initialization is random. Variable position vectors represent that the sailfish can search in multiple dimensions. In this algorithm, the candidate solution considered is sailfish and the positions of sailfish in the search space are the variables of the problem. Firstly, the sailfish and sardine populations are randomly initialized as  $XSF_i^{itr}$  and  $XS_j^{itr}$ , which are the position of sailfish and sardine populations where the subscripts  $i$  and  $j$  are the indices of sailfish and sardine from the initialized population and the superscript  $itr$  denotes the index of iteration.



Aristocracy involves finding the best search agent via the best sailfish fitness value and, for the sardines, the best fitness value of injured sardine and replicating the unchanged best solutions to the next generation. The best position of the search agent (sailfish) is kept in every iteration and measured as an Elite. The best or the fittest sailfish acquired until now is the Elite sailfish. It would be the one affecting the maneuverability and speeding up of sardines during the attacking. The location of any injured sardine is also saved in every iteration, which the sailfish will consider for group hunting as the best target selected.

Secondly, the fitness of each sailfish and sardine in the population is calculated using the fitness function (i.e., CC in the proposed algorithm). Based on this, Elite (i.e., the best sailfish) and injured sardine are acquired. The best sailfish is the one having the smallest fitness function value at iteration  $itr$ .

$$XSF_{best}^{itr} = \{XSF^{itr} | \text{sailfish with the smallest fitness value}\}$$

Similarly, the injured sardine is the one which has been attacked and injured by the sailfish and having the smallest value of CC.

$$XS_{inj}^{itr} = \{XS^{itr} | \text{sardine injured by the best sailfish}\}$$

### 6.3.3. Attack-Alternation Technique

Sailfish promote the success rate of hunting their prey with the help of attacking in coordination technique. Sailfish chase their prey and herd them, change their own position conferring to the position of the other hunting sailfish, without even directly communicating with each other. Through this attack-alternation technique, sailfish injure more sardines during the first phase of hunting, which leads to a higher rate of success in capturing the prey at advanced phases of group hunting.

Afterward, the termination condition is checked. If the condition is not satisfied, the position of sailfish is updated with the following equation:

$$XSF_{new}^{itr} = XSF_{best}^{itr} - \delta_{itr} \times \left( \varphi \times \left( \frac{XSF_{best}^{itr} + XS_{inj}^{itr}}{2} \right) - XSF_{old}^{itr} \right). \quad (21)$$

The symbols in the above update equation are defined as follows:  $XSF_{new}^{itr}$  is the updated position of sailfish,  $XSF_{best}^{itr}$  is the position of best sailfish,  $\delta_{itr}$  is the coefficient at iteration  $itr$ ,  $\varphi$  is a random number between 0 and 1,  $XS_{inj}^{itr}$  is the position of injured sardine and  $XSF_{old}^{itr}$  is the current position of sailfish.

$$\delta_{itr} = 2 \times \varphi \times P. \quad (22)$$

where  $P$  denotes the prey density.

The prey density represents the quantity of prey at each iteration. It is an important factor when updating the position of sailfish because the number of prey (i.e., sardines) will decline in group hunting as follows:

$$P = 1 - \left( \frac{nSF}{nSF + nS} \right) \quad (23)$$

where  $nSF$  and  $nS$  denote the numbers of sailfish and sardine, respectively, in each iteration.

After using Equation (21) for updating the position of sailfish, the attacked power of sailfish,  $AP$ , at iteration  $itr$  is calculated with

$$AP = C \times (1 - (2 \times itr \times \epsilon)), \quad (24)$$

where  $C$  and  $\epsilon$  are the coefficients for linearly decreasing  $AP$ .

#### 6.3.4. Hunting Prey

The observation of a complete massacre of sardine is very sporadic at the beginning of the group hunting. In more than 90% of the cases, the scales of sardines would be removed after the sailfish strikes their bodies. At the start of the hunting phase, the energy level of sailfish for hunting and catching its prey is higher, and the sardines are also not really drained and injured. This is the reason that sardines have excessive escape speed and high maneuverability. Sailfish's attacking power would decline steadily over the time of hunting.

The position of every sardine in the population is also updated based on the current position of sailfish and AP at every iteration. The following formula is used for updating the position of sardine:

$$XS_{new}^{itr} = \varphi_1 \times (XS_{best}^{itr} - XS_{old}^{itr} + AP), \quad (25)$$

where  $XS_{new}^{itr}$  and  $XS_{old}^{itr}$  are the updated and previous positions of sardine and  $\varphi_1$  is a random number between 0 and 1.

Considering the value of AP, if the attack power of sailfish is less than 0.5, only  $S$  number of sardines positions will be updated. Otherwise, all the sardines' positions will be updated. Here,  $S$  is determined by

$$S = nS \times AP. \quad (26)$$

Next, the fitness value (i.e., CC) of all the sardines and sailfish is recalculated as per their updated positions and population is sorted.

#### 6.3.5. Catching Prey

Alongside the reducing attacking power of sailfish, the energy levels of sardines would be decremented because of the recurrent powerful attacking of sailfish. The attacks also affect the maneuverability as it reduces the prey's ability to detect the directional information regarding the position of sailfish. This will result in pulling away the sardines from the school after being slashed by the sailfish's rostrum, and they would be quickly captured then.

In the last phase of hunting, the pulled away sardines are quickly captured by the sailfish. In this algorithm, it is considered that, if any sardine becomes fitter than the sailfish, it is removed from its population. The sailfish will update to the position of the corresponding sardine as follows:

$$XS_{itr}^{itr} = XS_{itr}^{itr}, \text{ if } CC(S_{itr}) < CC(SF_{itr}), \quad (27)$$

where  $CC(S_{itr})$  and  $CC(SF_{itr})$  denote the fitness values (i.e., CC values) of sardines and sailfish at iteration  $itr$ .

Thereafter, the position of best sailfish and injured sardine is also updated at every iteration.

#### 6.3.6. Deducing Optimal Sailfish

The injured sardine that pulled away from the school would quickly be captured. In SFOA, it is considered that, when a sardine becomes weak, its respective sailfish catches its prey. The hunted sardine's position replaces the sailfish's position, elevating the probabilities of new prey's hunting. After satisfying the termination condition, the best sailfish is acquired along with its fitness value, that is CC.

## 7. Results and Discussion

This section presents the results of the performance analysis of SFOA for 2D NoC for six standard NoC benchmarks, as shown in Table 1. Network size is standard  $4 \times 4$  for all considered benchmarks. For a fair comparison with previous state-of-the-art architectures, the network size is the same. VOPD application consists of 16 sub-tasks. These

sub-tasks can be mapped on a  $4 \times 4$  mesh network. However, in the case of MPEG4, MWD, MP3encMP3dec, 263encMP3dec and 263decMP3dec 4, 4, 3, 4 and 1 routers are idle, respectively.

**Table 1.** Standard NoC benchmarks details with mesh sizes.

Benchmark	Nodes	Edges	2D Mesh Size
VOPD [35]	16	21	$4 \times 4$
MPEG4 [31]	12	26	$4 \times 4$
MWD [35]	12	13	$4 \times 4$
MP3encMP3dec [31]	13	14	$4 \times 4$
263encMP3dec [31]	12	12	$4 \times 4$
263decMP3dec [31]	14	15	$4 \times 4$

### 7.1. Experimental Setup

To evaluate the performance of the proposed SFOA, different standard NoC benchmarks were considered and various experiments were conducted. The proposed algorithm was verified for 2D NoC with other nature-inspired algorithms such as ACO, PSO, GA, SA and CSO. The code for the proposed SFOA algorithm was written in Python and implemented on NoC Tweak Simulator [43]. All experiments were run on a PC Intel(R) Core (TM) i7-16GB RAM, 2.30 GHz processor. Table 2 depicts the details of the NoC Tweak platform for simulation.

**Table 2.** Simulation environment description.

Network Type	2D Mesh
Type of Platform	EMBEDDED
Embedded applications	VOPD, MPEG4, MWD, MP3encMP3dec, 263encMP3dec, 263decMP3dec
Mapping algorithm	SFOA, CSO, ACO, PSO, SA
Type of Router	WORMHOLE-PIPELINE
Routing algorithm	XY DIMENSION-ORDERED
Arbitration Policy	VIRTUAL CHANNEL ARBITRATION
Packet delivery type	WITHOUT ACK
Packet distribution	EXPONENTIAL
Sending ACK policy	SEND ACK OPTIMALLY
Packet length (fixed)	10 (flits)
Injection rate (flit)	0.1 (flits/cycle/node)
Output channel selection	XY-ORDERED
Buffer size	8 (flits)
Inter-route link length	10,000 ( $\mu\text{m}$ )
Pipeline type	8
Pipeline stages	4
Input clock frequency	1000 (MHz)
Operating clock frequency	1000 (MHz)
Warm-up time	20,000 cycles

### 7.2. Average Power Dissipation Analysis

To evaluate the efficiency of the proposed algorithm, power minimization analysis was also performed. It shows that SFOA outperformed other existing mapping techniques

and the average percentage of improvement on power minimization with other nature-inspired algorithms.

Table 3 shows the results for total power consumption in watts (W) of 2D  $4 \times 4$  mesh for six standard NoC benchmarks. From the results in Table 3, it is evident the average improvement of power minimization of our proposed algorithm SFOA is 3.63%, 23.7%, 18.70%, 22.14%, 27.25%, 18.66%, 12.08% and 4.73% over ILP, ACO, PSO, SAT, SA, GA, BA and CSO, respectively.

**Table 3.** Total Power (W) for 2D NoC  $4 \times 4$  Mesh for standard NoC benchmarks.

Mapping Algorithm	VOPD	MPEG4	MWD	MP3encMP3dec	263encMP3dec	263decMP3dec
ILP	1.528	1.137	1.012	1.228	1.286	1.211
ACO	1.920	1.423	1.218	1.498	1.599	1.738
PSO	1.841	1.357	1.112	1.507	1.445	1.561
SAT	1.856	1.370	1.236	1.524	1.563	1.624
SA	1.971	1.478	1.256	1.590	1.697	1.877
GA	1.843	1.356	1.109	1.507	1.445	1.561
BA	1.634	1.247	1.110	1.486	1.323	1.313
CSO	1.518	1.219	1.023	1.228	1.286	1.198
Proposed Algorithm	1.311	1.201	1.015	1.228	1.286	1.148

### 7.3. Communication Cost and Computation Time Analysis

The execution analysis of the proposed SFOA compared to other present nature-inspired mapping algorithms is presented in this section. Table 4 depicts the evaluation of average communication cost ( $H_{count} \times BW$ ) from Equation (8) for VOPD [34] and MPEG4 [30] standard NoC benchmarks for two-dimensional NoC.

As ILP [8] is regarded as one of the most competent algorithms in the exact mapping method for communication cost estimation, our proposed SFOA is explicitly compared with ILP as well, along with other algorithms. SFOA provides the same results for communication cost, as shown by the results in Table 4. The values of a few parameters are missing in Tables 4–6 for some benchmarks as they were not provided by the authors in the base papers of ACO [33] and SA [32].

**Table 4.** Estimation of communication cost for 2D NoC.

Mapping Algorithm	Communication Cost (Hops $\times$ Bandwidth) in MB/s	
	VOPD	MPEG4
ILP [8]	4119	3567
ACO [33]	-	3633
PSO [34]	4119	3567
SA [32]	4231	3567
GA [21]	4218	3772
BA [45]	4119	3567
CSO [44]	4119	3567
Proposed Algorithm	4119	3567

The percentage deviation from the exact mapping method based on ILP over heuristic-based mapping techniques for 2D NoC is shown in Table 5. However, the proposed SFOA gives the best results compared with other nature-inspired algorithms, as specified by the results in Table 6. In comparison with other existing mapping techniques, the proposed SFOA takes 69% less computation time. Table 6 represents the estimations for computation time in seconds and communication cost in MB/s of two-dimensional  $4 \times 4$  mesh for six standard NoC benchmarks.

**Table 5.** Percentage deviation over ILP based mapping techniques.

Mapping Algorithm	Percentage of Communication Cost Deviation	
	VOPD	MPEG4
ACO	-	1.9
PSO	0.0	0.0
SA	2.7	-
GA	2.4	5.7
BA	0.0	0.0
CSO	0.0	0.0
Proposed Algorithm	0.0	0.0

**Table 6.** Communication cost and computation time of 2D NoC  $4 \times 4$  mesh for standard NoC benchmarks.

Mapping Algorithm	VOPD		MPEG4		MWD	
	Communication Cost (Hops $\times$ Bandwidth) in MB/s	Computation Time in Seconds	Communication Cost (Hops $\times$ Bandwidth) in MB/s	Computation Time in Seconds	Communication Cost (Hops $\times$ Bandwidth) in MB/s	Computation Time in Seconds
ILP	4119	4679.341	3567	22.340	1120	210.021
ACO	-	-	3633	18.652	-	-
PSO	4119	3.785	3567	3.465	1120	3.432
SA	4231	3878.527	3567	-	1451	197.541
GA	4218	3.925	3772	3.234	1321	3.420
BA	4119	2.231	3567	2.925	1122	2.894
CSO	4119	2.231	3567	2.010	1122	1.996
Proposed Algorithm	4119	1.98	3567	1.96	1120	1.886

Mapping Algorithm	MP3encMP3dec		263encMP3dec		263decMP3dec	
	Communication Cost (hops $\times$ bandwidth) in MB/s	Computation Time in Seconds	Communication Cost (hops $\times$ bandwidth) in MB/s	Computation Time in Seconds	Communication Cost (hops $\times$ bandwidth) in MB/s	Computation Time in Seconds
ILP	17.021	1435.012	230.407	193.035	19.823	4897.210
ACO	17.231	1196.856	-	-	-	-
PSO	17.021	3.194	230.407	3.185	19.823	3.188
GA	17.133	3.194	230.698	3.185	19.911	3.174
BA	17.834	2.653	231.450	2.345	19.936	2.350
CSO	17.021	1.785	230.407	1.527	19.823	1.511
Proposed Algorithm	17.021	1.585	230.407	1.227	19.823	1.011

#### 7.4. Average Network Latency Analysis

For the analysis of the performance of the proposed SFOA, the impact of average network latency was also scrutinized with different types of traffic patterns on mesh topological architecture. The considered distinct types of traffic patterns are a uniform random traffic pattern and tornado traffic pattern. These traffic patterns are a method for defining the communication between the IP-cores of the NoC.

In the case of uniform random traffic patterns, it distributes the traffic uniformly, balances the load and each source is equally likely to communicate with each destination. In the case of tornado traffic patterns, it is devised as a combatant for torus topologies.

The performance analysis of the considered  $4 \times 4$  mesh-based NoC architecture was done using the XY-routing algorithm via the NoCTweak simulator [43]. The average network latency of the proposed algorithm, i.e., SFOA, was evaluated for the above-considered two types of traffic patterns compared with other existing nature-inspired heuristics algorithms.

Figure 7 depicts the graphical results of the average network latency in contrast to different rates of injection load under uniform random traffic patterns. It is evident from this graph that SFOA outperformed PSO, GA, BA and CSO by 11.23%, 16.40%, 8.65% and 4.42%, respectively, for uniform random traffic pattern. Furthermore, Figure 8 illustrates the graphical results of the average network latency compared to different injection load



rates under tornado traffic patterns. It can be seen from this graph that SFOA outperformed PSO, GA, BA and CSO by 24.06%, 25.45%, 13.89% and 5.82%, respectively, for tornado traffic patterns.

SFOA gives the best latency in comparison with other existing nature-inspired algorithms considered such as PSO, GA, BA and CSO using minimum hops count mapping technique.

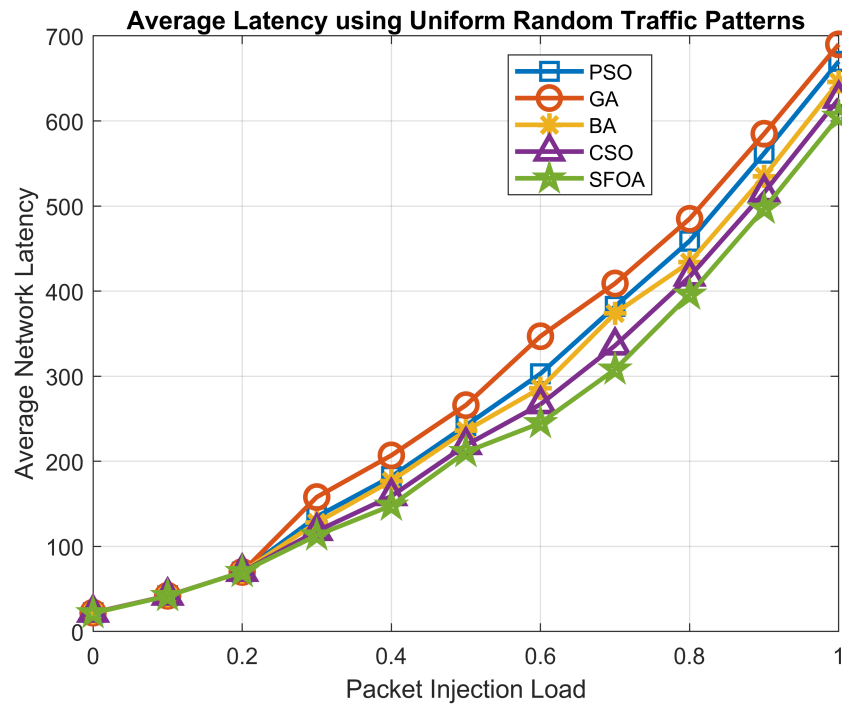


Figure 7. Average network latency for uniform random traffic patterns.

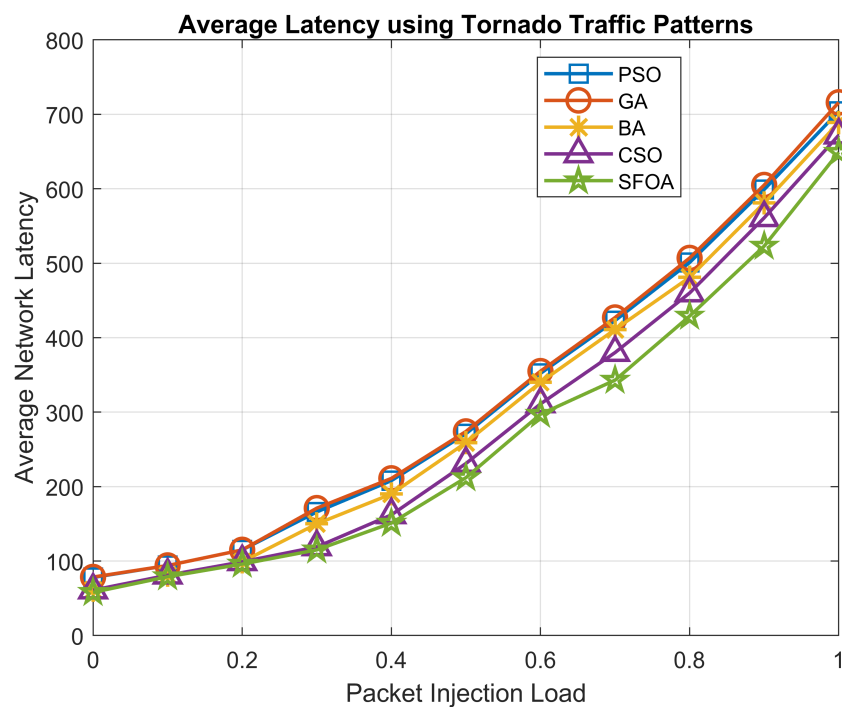


Figure 8. Average network latency for Tornado traffic patterns.

The mapping results of the proposed SFOA clearly indicate that it is more efficient than other existing nature-inspired algorithms. The results in figures and tables show the improvement in performance analysis parameters. It indicates the reduction in average network power consumption, computation time, communication cost and average network latency.

## 8. Conclusions

This paper presents a state-of-art nature-inspired metaheuristic algorithm, i.e., SFOA, which mainly comprises two advantages. The first advantage is high-speed convergence by strengthening the searching process used for the best sailfish group. The second advantage is robust optimization by strengthening the search space intended for the diversity of the sardine population. SFOA is used for the optimized mapping of the application task graph on a two-dimensional NoC with mesh topology. The efficiency of the proposed approach was assessed based on the results of the performance analysis parameters for six standard NoC benchmarks. The evaluation of the proposed SFOA proficiency was done via multiple experiments on alternative heuristic algorithms such as ACO, PSO, SA, GA, BA and CSO. The results shown in the previous section indicate that the average improvement of power minimization of the proposed algorithm SFOA is 3.63%, 23.7%, 18.70%, 22.14%, 27.25%, 18.66%, 12.08% and 4.73% over ILP, ACO, PSO, SAT, SA, GA, BA and CSO, respectively. In contrast to other existing mapping techniques, the proposed SFOA takes 69% less computation time. It is evident from the average network latency graphs that SFOA outperformed PSO, GA, BA and CSO for two distinct standards of traffic patterns for NoC by 11.23%, 16.40%, 8.65% and 4.42% for uniform random traffic patterns and 24.06%, 25.45%, 13.89% and 5.82% for tornado traffic patterns, respectively. The experiments results reveal that SFOA outperformed other nature-inspired algorithms to minimize power consumption, computation time, communication cost and latency. Moreover, this work can be continued in various ways, e.g., some hybrid algorithms can be introduced to reduce computation time further. This algorithm can also be implemented on 2D and 3D NoC architectures with different topologies.

**Author Contributions:** Conceptualization, S.S. and N.K.B.; Formal analysis, S.S., N.K.B., F.H. and W.A.; Investigation, S.S., N.K.B., F.H., W.A., Y.B.Z. and H.Y.; Methodology, S.S., N.K.B., F.H., W.A., Y.B.Z. and H.Y.; Resources, Y.B.Z. and H.Y.; Software, S.S. and N.K.B.; Validation, N.K.B., Y.B.Z. and H.Y.; Visualization, S.S., N.K.B., F.H. and W.A.; Writing—original draft, S.S. and N.K.B.; Writing—review and editing, Y.B.Z. and H.Y. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (2019R1A2C1083988), in part by Basic Science Research Program through the NRF funded by the Ministry of Education (2021R1I1A3041887) and in part by MSIT, Korea, under the ITRC (Information Technology Research Center) support program (IITP-2021-2016-0-00313) supervised by the IITP (Institute for Information & communications Technology Promotion).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Bjerregaard, T.; Mahadevan, S. A survey of research and practices of network-on-chip. *ACM Comput. Surv. (CSUR)* **2006**, *38*, 1–es. [[CrossRef](#)]
2. Tsai, W.C.; Lan, Y.C.; Hu, Y.H.; Chen, S.J. Networks on chips: Structure and design methodologies. *J. Electr. Comput. Eng.* **2012**, *2012*, 4. [[CrossRef](#)]
3. Baloch, N.K.; Baig, M.I.; Daneshmand, M. Defender: A low overhead and efficient fault-tolerant mechanism for reliable on-chip router. *IEEE Access* **2019**, *7*, 142843–142854. [[CrossRef](#)]

4. Rashid, M.; Baloch, N.K.; Shafique, M.A.; Hussain, F.; Saleem, S.; Zikria, Y.B.; Yu, H. Fault-Tolerant Network-On-Chip Router Architecture Design for Heterogeneous Computing Systems in the Context of Internet of Things. *Sensors* **2020**, *20*, 5355. [[CrossRef](#)] [[PubMed](#)]
5. Shafique, M.A.; Baloch, N.K.; Baig, M.I.; Hussain, F.; Zikria, Y.B.; Kim, S.W. NoCCGuard: A Reliable Network-on-Chip Router Architecture. *Electronics* **2020**, *9*, 342. [[CrossRef](#)]
6. Ibrahim, M.; Baloch, N.K.; Anjum, S.; Zikria, Y.B.; Kim, S.W. An energy efficient and low overhead fault mitigation technique for internet of thing edge devices reliable on-chip communication. *Softw. Pract. Exp.* **2020**. [[CrossRef](#)]
7. Bononi, L.; Concer, N. Simulation and analysis of network on chip architectures: Ring, spidergon and 2D mesh. In Proceedings of the Design Automation & Test in Europe Conference, Munich, Germany, 6–10 March 2006; Volume 2, p. 6.
8. Tosun, S.; Ozturk, O.; Ozen, M. An ILP formulation for application mapping onto network-on-chips. In Proceedings of the 2009 International Conference on Application of Information and Communication Technologies, Baku, Azerbaijan, 14–16 October 2009; pp. 1–5.
9. Shadravan, S.; Naji, H.; Bardsiri, V.K. The Sailfish Optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. *Eng. Appl. Artif. Intell.* **2019**, *80*, 20–34. [[CrossRef](#)]
10. Araki, D.; Yoshihiro, T. A Distance-Vector-Based Multi-Path Routing Scheme for Static-Node-Assisted Vehicular Networks. *Sensors* **2019**, *19*, 2688. [[CrossRef](#)]
11. Melo, D.R.; Zeferino, C.A.; Dilillo, L.; Bezerra, E.A. Maximizing the inner resilience of a network-on-chip through router controllers design. *Sensors* **2019**, *19*, 5416. [[CrossRef](#)]
12. Wu, L.; Cai, H. Energy-Efficient Adaptive Sensing Scheduling in Wireless Sensor Networks using Fibonacci Tree Optimization Algorithm. *Sensors* **2021**, *21*, 5002. [[CrossRef](#)]
13. Rhee, J.H.; Kim, S.I.; Lee, K.M.; Kim, M.K.; Lim, Y.M. Optimization of Position and Number of Hotspot Detectors Using Artificial Neural Network and Genetic Algorithm to Estimate Material Levels Inside a Silo. *Sensors* **2021**, *21*, 4427. [[CrossRef](#)] [[PubMed](#)]
14. Ajani, T.S.; Imoize, A.L.; Atayero, A.A. An Overview of Machine Learning within Embedded and Mobile Devices—Optimizations and Applications. *Sensors* **2021**, *21*, 4412. [[CrossRef](#)] [[PubMed](#)]
15. Singh, T.; Saxena, N.; Khurana, M.; Singh, D.; Abdalla, M.; Alshazly, H. Data Clustering Using Moth-Flame Optimization Algorithm. *Sensors* **2021**, *21*, 4086. [[CrossRef](#)]
16. Hu, J.; Marculescu, R. Energy-and performance-aware mapping for regular NoC architectures. *IEEE Trans.-Comput.-Aided Des. Integr. Circuits Syst.* **2005**, *24*, 551–562.
17. Lei, T.; Kumar, S. A two-step genetic algorithm for mapping task graphs to a network on chip architecture. In Proceedings of the Euromicro Symposium on Digital System Design, Belek-Antalya, Turkey, 1–6 September 2003; pp. 180–187.
18. Murali, S.; De Micheli, G. Bandwidth-constrained mapping of cores onto NoC architectures. In Proceedings of the Proceedings Design, Automation and Test in Europe Conference and Exhibition, Paris, France, 16–20 February 2004; Volume 2, pp. 896–901.
19. Lu, Z.; Xia, L.; Jantsch, A. Cluster-based simulated annealing for mapping cores onto 2D mesh networks on chip. In Proceedings of the 2008 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, Bratislava, Slovakia, 16–18 April 2008; pp. 1–6.
20. Radu, C.; Vințan, L. Domain-knowledge optimized simulated annealing for Network-on-Chip application mapping. In *Advances in Intelligent Control Systems and Computer Science*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 473–487.
21. Ascia, G.; Catania, V.; Palesi, M. Multi-objective mapping for mesh-based NoC architectures. In Proceedings of the 2nd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, Stockholm, Sweden, 8–10 September 2004; pp. 182–187.
22. Jena, R.; Sharma, G. Application mapping of mesh based-NoC using multi-objective genetic algorithm. *Int. J. Comput. Appl.* **2008**, *30*, 17–22.
23. Sepúlveda, J.; Strum, M.; Chau, W.J.; Gogniat, G. A multi-objective approach for multi-application NoC mapping. In Proceedings of the 2011 IEEE Second Latin American Symposium on Circuits and Systems (LASCAS), Bogota, Colombia, 23–25 February 2011; pp. 1–4.
24. Harmanani, H.M.; Farah, R. A method for efficient mapping and reliable routing for NoC architectures with minimum bandwidth and area. In Proceedings of the 2008 Joint 6th International IEEE Northeast Workshop on Circuits and Systems and TAISA Conference, Montreal, QC, Canada, 22–25 June 2008; pp. 29–32.
25. Hu, J.; Marculescu, R. Energy-aware mapping for tile-based NoC architectures under performance constraints. In Proceedings of the 2003 Asia and South Pacific Design Automation Conference, Kitakyushu, Japan, 24 January 2003; pp. 233–239.
26. Ye, T.T.; Micheli, G.D.; Benini, L. Analysis of power consumption on switch fabrics in network routers. In Proceedings of the 39th Annual Design Automation Conference, New Orleans, LA, USA, 10–14 June 2002; pp. 524–529.
27. Bhat, S. *Energy Models for Network-on-Chip Components*; Master of Science, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven: Eindhoven, The Netherlands, 2005.
28. Kahng, A.B.; Lin, B.; Samadi, K. Improved on-chip router analytical power and area modeling. In Proceedings of the 2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC), Taipei, Taiwan, 18–21 January 2010; pp. 241–246.
29. Ost, L.; Guindani, G.; Moraes, F.; Indrusiak, L.; Määttä, S. Exploring NoC-based MPSoC design space with power estimation models. *IEEE Des. Test Comput.* **2010**, *28*, 16–29. [[CrossRef](#)]

30. Feero, B.S.; Pande, P.P. Networks-on-chip in a three-dimensional environment: A performance evaluation. *IEEE Trans. Comput.* **2008**, *58*, 32–45. [[CrossRef](#)]
31. Sahu, P.K.; Chattopadhyay, S. A survey on application mapping strategies for network-on-chip design. *J. Syst. Archit.* **2013**, *59*, 60–76. [[CrossRef](#)]
32. Marcon, C.; Borin, A.; Susin, A.; Carro, L.; Wagner, F. Time and energy efficient mapping of embedded applications onto NoCs. In Proceedings of the 2005 Asia and South Pacific Design Automation Conference, Shanghai, China, 21–21 January 2005; pp. 33–38.
33. Ferrandi, F.; Lanzi, P.L.; Pilato, C.; Sciuto, D.; Tumeo, A. Ant colony heuristic for mapping and scheduling tasks and communications on heterogeneous embedded systems. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2010**, *29*, 911–924. [[CrossRef](#)]
34. Sahu, P.K.; Shah, T.; Manna, K.; Chattopadhyay, S. Application mapping onto mesh-based network-on-chip using discrete particle swarm optimization. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2013**, *22*, 300–312. [[CrossRef](#)]
35. Alagarsamy, A.; Gopalakrishnan, L. SAT: A new application mapping method for power optimization in 2D—NoC. In Proceedings of the 2016 20th International Symposium on VLSI Design and Test (VDAT), Guwahati, India, 24–27 May 2016; pp. 1–6.
36. Tavanpour, M.; Khademzadeh, A.; Janidarmian, M. Chain-Mapping for mesh based Network-on-Chip architecture. *IEICE Electron. Express* **2009**, *6*, 1535–1541. [[CrossRef](#)]
37. Tosun, S. New heuristic algorithms for energy aware application mapping and routing on mesh-based NoCs. *J. Syst. Archit.* **2011**, *57*, 69–78. [[CrossRef](#)]
38. Patooghy, A.; Tabkhi, H.; Miremadi, S.G. RMAP: A reliability-aware application mapping for network-on-chips. In Proceedings of the 2010 third International conference on dependability, Venice, Italy, 18–25 July 2010; pp. 112–117.
39. Niknam, T.; Amiri, B. An efficient hybrid approach based on PSO, ACO and k-means for cluster analysis. *Appl. Soft Comput.* **2010**, *10*, 183–197. [[CrossRef](#)]
40. Junior, L.S.; Nedjah, N.; de Macedo Mourelle, L. Routing for applications in NoC using ACO-based algorithms. *Appl. Soft Comput.* **2013**, *13*, 2224–2231. [[CrossRef](#)]
41. Xie, B.; Chen, T.; Hu, W.; Tang, X.; Wang, D. An energy-aware online task mapping algorithm in NoC-based system. *J. Supercomput.* **2013**, *64*, 1021–1037. [[CrossRef](#)]
42. Herbert-Read, J.E.; Romanczuk, P.; Krause, S.; Strömbom, D.; Couillaud, P.; Domenici, P.; Kurvers, R.H.; Marras, S.; Steffensen, J.F.; Wilson, A.D.; et al. Proto-cooperation: Group hunting sailfish improve hunting success by alternating attacks on grouping prey. *Proc. R. Soc. Biol. Sci.* **2016**, *283*, 20161671. [[CrossRef](#)] [[PubMed](#)]
43. Tran, A.T.; Baas, B. *NoCTweak: A Highly Parameterizable Simulator for Early Exploration of Performance and Energy of Networks on-Chip*; Tech. Rep. ECE-VCL-2012-2; VLSI Computation Lab, ECE Department, University of California: Davis, CA, USA, 2012.
44. Alagarsamy, A.; Gopalakrishnan, L.; Mahilmaran, S.; Ko, S.B. A self-adaptive mapping approach for network on chip with low power consumption. *IEEE Access* **2019**, *7*, 84066–84081. [[CrossRef](#)]
45. Li, J.; Song, G.; Ma, Y.; Wang, C.; Zhu, B.; Chai, Y.; Rong, J. Bat algorithm based low power mapping methods for 3D network-on-chips. In Proceedings of the National Conference of Theoretical Computer Science, Nanning, China, 13–15 November 2020; pp. 277–295.

## Short Biography of Authors

**Saleha Sikandar** completed her BS in Electronic Engineering from International Islamic University Islamabad (IIUI), Pakistan in 2017. She is currently doing her MSc in Computer Engineering from University of Engineering & Technology (UET) Taxila, Pakistan. She has vast experience in research and development in various embedded systems companies. She worked on the development of many embedded and chip design systems. Her research interests include embedded systems design, Network on Chip (NoC), and reconfigurable systems designs. She is currently working on the low-cost application mapping on NoC.

**Naveed Khan Baloch** received his BSc degree in Computer Engineering from the University of Engineering and Technology, Taxila, Pakistan in 2007. He has worked in multinational companies as an embedded system designer from 2007 to 2010. He joined the university as a lecturer after completing his MS degree from UET Taxila. He recently completed his Ph.D. in Computer Engineering from the same university. He has published many research papers in his field and has experience in embedded system design, fault tolerant systems, reconfigurable computing, and he is currently working on self-healing digital systems. Nowadays, he is working as an Assistant Professor in Computer Engineering Department UET Taxila. During his tenure in academia, he did many collaborations with industry and foreign universities in the field of on-chip networks, embedded vision, and reconfigurable computing.

**Fawad Hussain** received his B.Sc. in Computer Engineering, M.Sc. in Electrical Engineering, and Ph.D. in Computer Engineering at the University of Engineering and Technology (UET), Taxila, Pakistan in 2005, 2009, and 2015, respectively. He is currently working as an Assistant Professor in the Computer Engineering Department, the University of Engineering and Technology (UET), Taxila, Pakistan. His research interest includes speech and audio processing, computer vision, human activity recognition, and emotion recognition. He is currently leading research for his MS and Ph.D. students in the mentioned areas of interest.

**Waqar Amin** received his B.Sc. degree in computer engineering from the University of Engineering and Technology at Taxila (UET Taxila), Pakistan, in 2007, and MS degree also from the UET Taxila in 2014. Currently, he is pursuing his Ph.D. degree. He has

vast experience in research and development in various embedded systems companies. He worked on the development of many GSM and 3G systems. His research interests include fault-tolerant systems, Network on Chip (NoC), self-healing, and reconfigurable systems designs. Currently, he is working on the low-cost application mapping on NoC.

**Yousaf Bin Zikria** is currently working as an Assistant Professor in the Department of Information and Communication Engineering, College of Engineering, Yeungnam University, Gyeongsan-Si, South Korea. He received a Ph.D. degree from the Department of Information and Communication Engineering, Yeungnam University, Korea, in 2016. He has more than ten years of experience in research, academia, and industry in the field of Information and Communication Engineering and Computer Science. He authored more than 90 scientific peer-reviewed journals, conferences, patents, and book chapters. GoogleScholar: <https://scholar.google.com/citations?user=K90qMyMAAAAJhl=en> (accessed on 20 July 2021), Website: <https://sites.google.com/view/ybzikria> (accessed on 20 July 2021), Researchgate: <https://www.researchgate.net/profile/YousafZikria> (accessed on 20 July 2021).

**Heejung Yu** (Senior Member, IEEE) received a B.S. degree in Radio Science and Engineering from Korea University, Seoul, South Korea, in 1999 and M.S. and Ph.D. degrees in Electrical Engineering from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2001 and 2011, respectively. From 2001 to 2012, he was with the Electronics and Telecommunications Research Institute, Daejeon, South Korea, and, from 2012 to 2019, he was with the Yeungnam University, Gyeongsan, South Korea. Currently, he is an associate professor with the Department of Electronics and Information Engineering, Korea University, Sejong, South Korea. His areas of interest include statistical signal processing and communication theory.