Journal of Universal Computer Science, vol. 31, no. 2 (2025), 136-158
submitted: 20/3/2024, accepted: 24/9/2024, appeared: 28/2/2025

Energy-aware Application Mapping onto 3D Mesh-Based Network-on-Chip using Heuristic Mapping Algorithms

Kiran K A
(Rajagiri School of Engineering & Technology (Autonomous), APJ Abdul Kalam Technical University, Kerala, India
https://orcid.org/0000-0001-5423-9648, kirank@rajagiritech.edu.in)

Jaison Jacob
(Rajagiri School of Engineering & Technology (Autonomous), APJ Abdul Kalam Technical University, Kerala, India
https://orcid.org/0000-0003-4836-3874, jaison_jacob@rajagiritech.edu.in)

Abstract: Network-on-chip (NoC) architectures have emerged as a potential solution for facilitating communication between processing elements (PEs) in modern multi-core systems. The design and optimization of NoC architectures are critical for achieving efficient communication, reduced energy consumption, and improved overall system performance. In this study, we investigate and compare the performance of two prominent optimization algorithms, like Genetic Algorithm (GA) and CastNet Algorithm, for 2D and 3D mesh NoC architectures. The study's objective is to estimate these algorithms' effectiveness in optimizing communication cost, communication energy, and CPU run time in both 2D and 3D mesh NoC architectures. Performance metrics such as communication cost, communication energy consumption, and CPU run time are measured and compared between the two algorithms and carried out on real and custom benchmark applications like MWD, VOPD and MPEG4.

Keywords: Network-on-chip, Genetic Algorithm, CastNet Algorithm, 2D & 3D NoC architecture
Categories: C.1.2, C.1.4, C.2.1, C.2.2, C.2.4
DOI: 10.3897/jucs.123539

1 Introduction

The NoC communication architecture has received substantial attention in recent times. It provides a propitious solution for addressing data transmission bottlenecks in sophisticated systems-on-chip (SoCs). The scalable and flexible interconnect mechanism provides efficient communication between on-chip network elements. NoC connects several processors or cores to enable them to communicate with each other. NoC includes a network of routers and communication links, forming a grid-like structure. It enables communication among the processing elements, memories, and other components of the SoC. Several key parameters, such as topology, routing algorithms, arbitration schemes, and flow control mechanisms were considered while designing the NoC architecture. NoC can significantly improve the performance and efficiency of SoCs while reducing design complexity and cost. As higher computational power and communication bandwidth requirement grows, NoC will surely play a crucial role in designing next-generation SoCs.

The design of NoC follows multiple metrics, including the number of processing nodes, the communication requirements, and the power and area constraints. The routing

algorithms cater to the task of selecting the optimal path for data transmission between the processing nodes, while considering parameters like congestion in the network, available bandwidth, and the latency. The commonly used routing algorithms are adaptive, deterministic, and hybrid. The network topology controls the physical connectivity between the processing elements. The most common topologies are 2D mesh, torus, ring, 3D mesh, and tree based. The criteria for selection of the topology varies based on several factors, including the number of processing nodes, the communication bandwidth requirements, and constraints on power and area. The buffer management is responsible for managing the data packets in the network. It should consider the available memory, the data rate, and the communication requirements. The most common buffer management techniques are virtual channel, wormhole, and store-and-forward. The power and area constraints are critical design considerations for NoC. The communication system should be energy-efficient and area-optimized. To minimize the power consumption of NoC[Gu et al. 2024], several techniques like voltage scaling, clock gating, and power gating are frequently utilized.

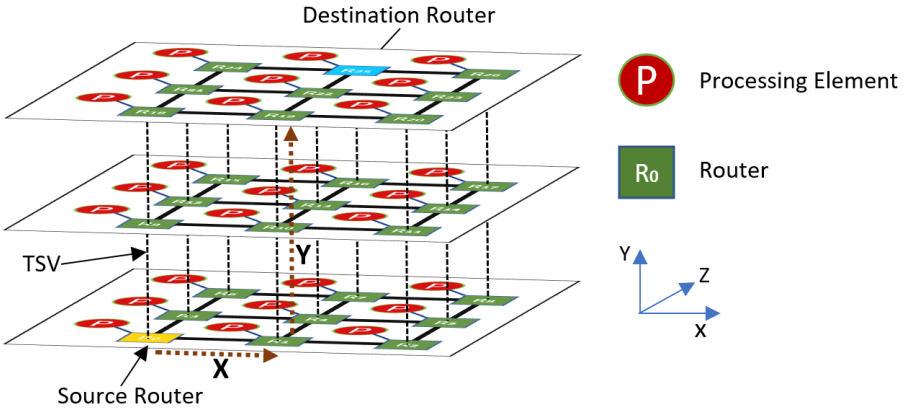

Figure 1: 3-layer 3D mesh NOC architecture for XYZ routing.

This paper presents a simulation-based experimental setup with result comparison using various benchmark applications (such as MPEG-4, VOPD, DVOPD, MWD etc.) and some custom benchmark applications on a 3-layer 3D mesh using XYZ routing scheme as shown in Figure 1. A 3D mesh network-on-chip (NoC) consists of a grid of nodes connected by interconnects, where each node represents a processing element or a memory block. The nodes arranged in 3D topology allow optimal routing of data packets through the network. Each node can have six links connecting adjacent routers: two along X, Y and Z directions. The XYZ routing scheme is a simple routing algorithm commonly used in 3D mesh NoC's. This algorithm routes data packets through the network based on their destination coordinates. The data packet or flits will be initially routed along the X direction, followed by routing in the Y direction, and ultimately directed towards the Z direction [Nalci et al. 2021].

2 Related Works

Several categories of Network-on-Chip (NoC) routing algorithms have been proposed over time to optimize various network performance parameters, as suggested by the literature. These algorithms are designed to achieve specific network performance goals for 2-D mesh-based NoC architecture [Pradip et al. 2013]. Branch and Bound (BB) and Integer Linear Programming (ILP) formulation [Nalci et al. 2021] produce high-quality optimal solutions at the expense of greater CPU time. Hence these techniques are used only for small-sized NoC designs. The pseudo heuristic mappings techniques like Simulated Annealing (SA) [Hanna et al. 2018], Genetic Algorithm (GA) [Zhen et al. 2013], and Particle Swarm Optimization (PSO) [Sahu et al. 2014] are population-based greedy algorithms that converge to near exact solution and typically require high computation time but much less than that of exact mapping techniques like BB, ILP etc. Heuristic application mapping techniques list CastNet algorithm [Tosun et al. 2011] speeds up the mapping process at the expense of higher communication energy than pseudo heuristic application mappings.

The selection of routing schemes significantly influences the flexibility of mapping algorithms. Several routing schemes like Deterministic Routing (XY routing), Adaptive Routing (like Odd-Even, turn model) and Hybrid Routing (Utilizing a mix of fixed and dynamic routes to balance performance and complexity) [Zhang et al. 2009] are commonly used. Deterministic NoC routing algorithms are most effective in achieving the shortest route path length as their objective. These algorithms use XY routing [Ankur et al. 2022] having fixed routing paths, to establish the shortest path between the sender and the receiver core.

2.1 Research Motivation

The routing techniques and architectures mentioned above have been primarily designed for planar chip designs that utilize 2D topologies such as mesh, torus, and tree. The main objective of this paper is to present an optimised solution for mapping applications onto 3D-NoC architectures with energy consumption and communication costs as the evaluation metrics. Here we propose a constructive heuristic algorithm, CastNet_3D and a pseudo heuristic algorithm, GA_3D, for mapping tasks onto the 3D-NoC architectures-based cores. Both mapping algorithms maps applications onto 3D mesh architecture considering the vertical links called through silicon via's (TSV's). Given an application with N cores, our task reduces to finding a mapping strategy for a 3-D mesh in which the total communication bandwidth and energy consumption are minimized.

3 System Model

In the preceding section, we formally define our work's communication energy and cost model.

3.1 Communication Energy Model

Here, we aim to minimize the communication cost and energy consumed across the processing elements of the architecture. Total energy consumed per traffic is defined as the total energy required to transmit one bit across the NoC. E is the total energy consumed

when one bit of data is transferred across internal interconnection wires, switches, links, and buffers along route [Mandelli et al. 2011][Jingcao et al. 2003]. Therefore, the energy model for 3-D mesh-based NoC architecture is given by Equation 1.

$$E = E_S + E_B + E_W + E_L \tag{1}$$

The energy consumed across switches, buffers and internal wires can be consolidated as router energy, E_R. Hence Equation 1 becomes,

$$E = E_R + E_L \tag{2}$$

Equation 2 gives the energy consumed per traffic as the sum of energy consumed when data passes through the switches and the physical links[Mohajer et al. 2022]. The average energy consumption per bit, E_{avg} across $tile_i$ and $tile_j$ is linearly proportional to the number of routers, r_{ij} between $tile_i$ and $tile_j$. As TSVs (Through Silicon Via) are shorter than horizontal links, we multiply a scaling factor, θ of 0.2 with their energy consumption [Nalci et al. 2021][Tosun et al. 2018].

$$E_{avgi,j} = (E_R * r_{ij}) + (E_L * r_{ij,h}) + (E_L * \theta * r_{ij,v}) \tag{3}$$

Here Manhattan distance determines the distance between two cores i and j.

$$r_{ij,h} = |x_i - x_j| + |y_i - y_j| \tag{4}$$

$$r_{ij,v} = |z_i - z_j| \tag{5}$$

The total energy required for data transmission through the network is a function of bandwidth e_{ij} between the two tasks v_i and v_j and the distance separating the routers[Gan et al. 2021]. Therefore, the total energy required for the data transmission is given by the Equation 6.

$$E_{Total} = \sum_{i,j} (E_{avgi,j} * e_{ij}) \tag{6}$$

If a bit traverses through r_{ij} routers, it is evident that it will pass through $r_{ij} - 1$ links connecting cores i and j which is commonly referred to as the number of hops, represented by the Equation 7.

$$l_{ij} = r_{ij} - 1 \tag{7}$$

3.2 Communication Cost Model

The function communication cost is calculated as the sum of the product of the number of hops and the required bandwidth across all the nodes in the core graph as shown in Equation 8.

$$T_{cost} = sum(l_{ij} * M_{ij}) \tag{8}$$

Here, l_{ij} defines the number of hops between $node_i$ and $node_j$ of the topology graph and M_{ij} represents the required bandwidth across the nodes.

4 Mapping Algorithms

This work adopts a 3D system model for NoC architecture as given in Figure 1, where TSVs (Through Silicon Via) form the vertical links. It is modelled as a directed graph G(V,E) and the routing protocol used is deterministic XYZ routing. Both constructive heuristic (CastNet algorithm) and transformative heuristic mapping algorithms (Genetic algorithm) were used for mapping various benchmark applications onto the 3D mesh NoC. The mapping function: V→C is defined as the mapping of the core graph G(V,E) onto the topology graph T(C), where V is the set of vertices representing the cores $v_i \in$V and E is the set of edges representing the required bandwidth between the cores $e_{ij} \in$E.

4.1 Benchmark applications

To test the effectiveness of our simulation experiment we took six classic benchmarks used for video processing applications: VOPD (Video Object Plane Decoder), MPEG-4 (Moving Pictures Expert Group 4), MWD (Multi Window Displayer), 263 decoder MP3 decoder, 263 encoder MP3 decoder and DVOPD (Double Video Object Plane Decoder) [Pradeep et al. 2019][Pradip et al. 2013]. Figure 2 shows some of the standard benchmark applications used in our experiments.

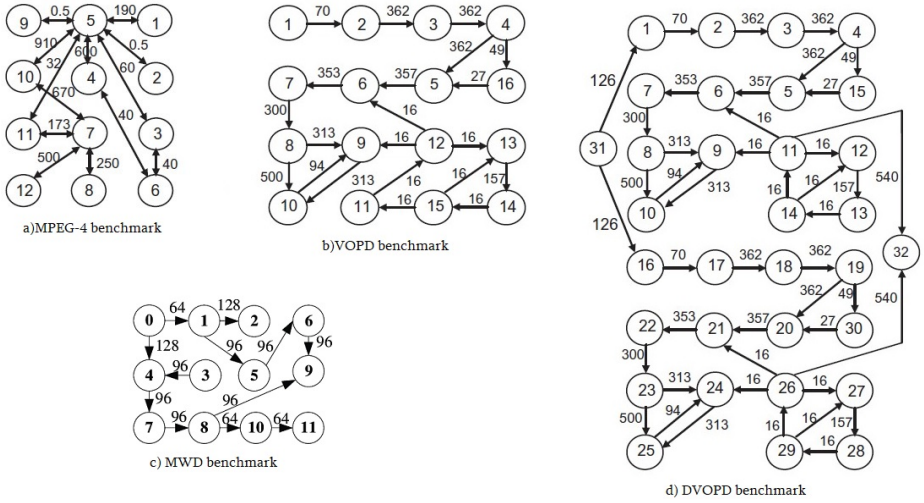

Figure 2: Core communication Graph of standard benchmark applications.

To ensure a comprehensive assessment, our experiment incorporates a collection of random task graphs for multiprocessor task scheduling, as referenced in [Boveiri, Hamid Reza 2018]. These custom benchmarks encompass five distinct node values, with our selection focusing on just three: 32, 64, and 128. We gauge the Communication-to-Computation Ratio (CCR) to ascertain the extent to which a graph leans towards being communication-intensive or computation-intensive. Node weights are randomly assigned from a uniform distribution, with an average set to 50 over a given time-instance.

Code	Application	Core count	No. of edges	Mesh structure
B1	VOPD	16	21	2x3x3
B2	MPEG-4	12	13	2x2x3
B3	MWD	12	12	2x2x3
B4	263dec mp3dec	14	15	2x3x3
B5	263enc mp3dec	12	12	2x2x3
B6	DVOPD	32	44	2x4x3

Table 1: Standard Benchmark applications.

Likewise, edge weights are chosen randomly from a uniform distribution, with the average computation cost multiplied by the CCR serving as the mean for these selections. The dataset comprises five distinct CCR values, specifically 0.1, 0.5, 1.0, 5.0, and 10.0. In this range, a CCR of 0.1 signifies task graphs that are predominantly computation-intensive, while a CCR of 10.0 indicates primarily communication-intensive task graphs. For our experiment, we selected 0.1 and 10.0 as CCR values. Table 1 and Table 2 list the number of nodes, edges, and mesh structures for the standard and synthetic benchmark applications used in our application.

Code	Application	Core count	No. of edges	Mesh structure
C1	32N_0.1	32	80	3x4x3
C2	32N_10	32	306	3x4x3
C3	64N_0.1	64	169	5x5x3
C4	64N_10	64	821	5x5x3
C5	128N_0.1	128	347	7x7x3
C6	128N_10	128	1971	7x7x3

Table 2: Synthetic Benchmark applications.

4.2 CastNet mapping algorithm

This section covers a detailed explanation of the CastNet algorithm for the mapping problem. The CastNet algorithm is a constructive algorithm. A constructive algorithm arrives at a solution step by step, by selecting the nodes and cores based on predefined criteria until the desired outcome is achieved. Once the solution is obtained, the positions of the tasks on the mesh remain fixed and unchanged.

The CastNet algorithm can generate multiple solutions based on the mesh symmetry. Each set of symmetric cores results in a unique solution. Therefore, the solution set is the same as the sets of symmetric cores in the mesh. The CastNet algorithm operates based on two decisions [Tosun et al. 2011]. (i) identifying the next node to be placed. (ii) the core onto which the node must be placed. The process of node selection consists of two distinct steps: the Initial node selection and the subsequent selections for the remaining nodes. Similarly, the core selection procedure can be broken down into two steps: (1) the selection of the initial core for the first node, and (2) the selection of the remaining nodes. The pseudocode for the CastNet algorithm is given in Algorithm 1.

Algorithm 1 Proposed CastNet mapping Algorithm for 3-D NoC

Core Graph G, Topology Graph T
Mapping of G on to T, Best mapping M, Total Energy TEC, Minimum cost TCC
$v_h = top\_priority(G)$
$C_d = get\_candidates(T)$
$TEC_{best} = \infty$
$TCC_{best} = \infty$
for $i = 1$ to $|Cd|$ **do**
  G_i = G; T_i = T
  $G'_i = \phi$; $T'_i = \phi$
  $f(v_h) = t_{c[i]}$
  $G_i = G_i - \{v_h\}$; $G'_i = G'_i + \{v_h\}$
  $T_i = T_i - \{t_{c[i]}\}$; $T'_i = T'_i + \{t_{c[i]}\}$
  while $G \neq \phi$ **do**
    $t_0 = select\_task(G_i, G'_i)$
    $c_0 = select\_core(t_0, t_i, t'_i)$
    $f(v_n) = t_k$
    $G_i = G_i - \{t_0\}$; $G'_i = G'_i + \{t_0\}$
    $T_i = T_i - \{c_0\}$; $T'_i = T'_i + \{c_0\}$
    $TEC_i, TCC_i = total\_cost()$
    if $(TEC_i < TEC_{best})$ **then**
      $TEC_{best} = TEC_i$
      $TCC_{best} = TCC_i$
      $best\_map() = curr\_map()$
      $returnbest\_map(), TEC_{best}, TCC_{best}$

The first step in the algorithm is to define a function top_priority to fix the initial task for mapping as indicated in line 2. Here the priority P_i (Equation 9) of task i is calculated based on the total communication with the surrounding cores. To select the initial task consider the task with the highest priority and is mapped onto the mesh. In the event of a tie, the task with the highest average cost is selected, H_i as given in Equation 10. Still, if tied, we select the initial task at random [Tosun et al. 2011]. $N(v_i)$ represents the number of neighbours of v_i.

The next step is to find a set of initial candidate cores, selected from different symmetry groups. Initially, any core can be considered as a candidate for the initial core selection. However, only a select few cores need to be selected as some cores are symmetrical to each other. Hence, from all symmetric groups select only a single candidate core.

$$P_i = \sum_{e_{i,j} \in E} w_{i,j} \tag{9}$$

$$H_i = \sum_{e_{i,j} \in E} w_{i,j} / N(v_i) \tag{10}$$

Figure 3 shows the symmetrical candidate cores for a 3×4×4 3D-mesh. The initial

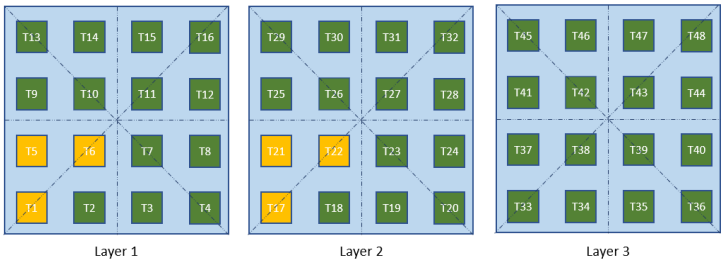

Figure 3: Candidates for initial core selection.

pool of candidate cores is the same as the number of symmetrical candidate groups. For a 3 x N x N 3D-mesh, the number of symmetrical candidate groups is given by the Equation 11.

$$|C| = (\lceil N/2 \rceil)(\lceil N/2 + 1 \rceil) \tag{11}$$

The function get_candidates is used for this step (line 4). For all the candidates as indicated in lines 7–19, we find the highest priority task and assign it to the first core as in line 10. Now we update the set of mapped, G_i' and unmapped, G_i tasks as in line 11. Similarly update the mapped, T_i' and unmapped, T_i cores as in line 12. The initial core selection for a 3 x 4 x 4 mesh is shown in Figure 3 and candidate cores are T1, T5, T6, T17, T21 and T22.

The remaining tasks are selected based on their communication weights with the help of the function select_task(G_i, G_i') (line 13). In this algorithm, the next task is selected by considering all the unmapped tasks, $t_0 \in T_i$ having edges with mapped tasks in T_i' and calculating the total communication of T_0 with mapped tasks in T_i'. Choose the highest communicating task. As for a tiebreaker, select the task with the greater priority.

Next, considering the communication cost of the mesh, we determine the cores for the chosen tasks, employing the function select_core$(v_n,\ T_i, T_i')$ as indicated in line 15. To identify the eligible core where the selected task is to be placed, the algorithm assesses all vacant cores and estimates the total cost between the selected task for each empty core with the mapped tasks. Subsequently, the algorithm maps the task to an available core that minimizes the cost. In situations where there are multiple candidate cores with the same minimum cost, the algorithm decides based on the number of interactions between edges (E_i) of the current task (t_0) and the unmapped tasks in T_i', along with neighbouring free 1-hop distance cores (H_k) of the current core (c_k) in T_i'. If still there are multiple candidates, select the first core in chronological order. This process continues until the list of unmapped tasks becomes empty (lines 13–19). For every solution, the total communication cost (TCC) and the total consumed energy (TEC) are calculated and compared with the previous results. Lastly, the best mapping solution with minimum TCC, TEC is obtained as our solution (lines 20–25).

4.3 Genetic mapping algorithm

The genetic algorithm represents a metaheuristic methodology utilized for the resolution of search and optimization problems. Its basis is rooted in the biological process of evolution and emulates functions such as selection, mutation, and crossover to identify

optimal or nearly optimal solutions [Zhen et al. 2013]. Much like the natural world, where individuals who are best adapted to their surroundings are chosen through successive generations, the genetic algorithm selects the most suitable solutions from each generation. Given its effectiveness in resolving complex optimization problems that contain multiple parameters, it is a compelling approach and has been implemented in the context of mapping problems.

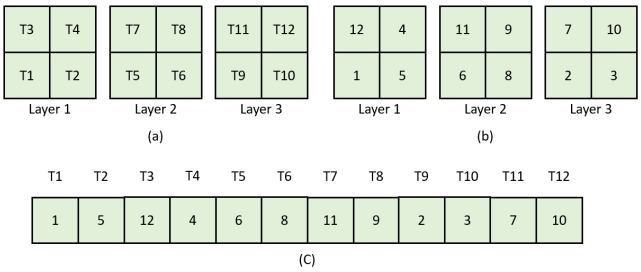

Figure 4: (a) 3×2×2 mesh, (b) mapping of MWD task graph onto a 3×2×2 3D-mesh, (c) mapping represented as chromosomes.

In our proposed Genetic Algorithm (GA) method, the first step involves creating a chromosome structure. This chromosome is formed by stringing together genes, where each gene represents a tile of the mesh. The size of the chromosome is limited by the tile count in our mesh. Our initial population is comprised of a set of chromosomes that have been randomly generated. Figure 4 illustrates the mapping of a weighted task graph and its corresponding chromosome representation. Figure 4(a) shows the tile sequence for a 3×2×2 mesh. The sequence starts from the lower-left corner tile of Layer 1 and ends at the upper-right corner tile of Layer 3 on the 3D mesh. Meanwhile, Figure 4(b) illustrates a random mapping of the MWD benchmark application onto a 3×2×2 3D-mesh architecture. Figure 4(c) illustrates the chromosome representation used for this mapping. An initial population of 100 individuals was generated using random mapping. After creating the initial population, each chromosome's fitness is calculated, which is the inverse of the communication cost of the mappings in the population.

The randomly generated initial population undergoes a series of operations, including selection, mutation, and crossover operators to create new individuals. Figure 5 illustrates the application of mutation and crossover operations on randomly selected pairs of chromosomes. Initially, the crossover operation is applied to two individuals within the population. In this process, a crossover point on the chromosome is randomly determined, and the second portions of both chromosomes are exchanged or swapped. After this exchange, two child individuals are created. In the newly generated chromosomes, some nodes may occur twice whereas some may be entirely missing. As we observe in Figure 5, nodes 6 and 8 of child 1 occur twice while nodes 9 and 11 are missing. Likewise, in Child 2, nodes 9 and 11 occur twice while nodes 6 and 8 are missing. For addressing these invalid chromosomes, the first step involves removing all nodes listed more than once in the chromosomes. Subsequently, any missing nodes are randomly assigned to the empty genes until the chromosome contains all the required nodes. After a crossover operation, the population size increases from n to 2n individuals.

The principal stages Involved in the execution of the Genetic algorithm are as follows:

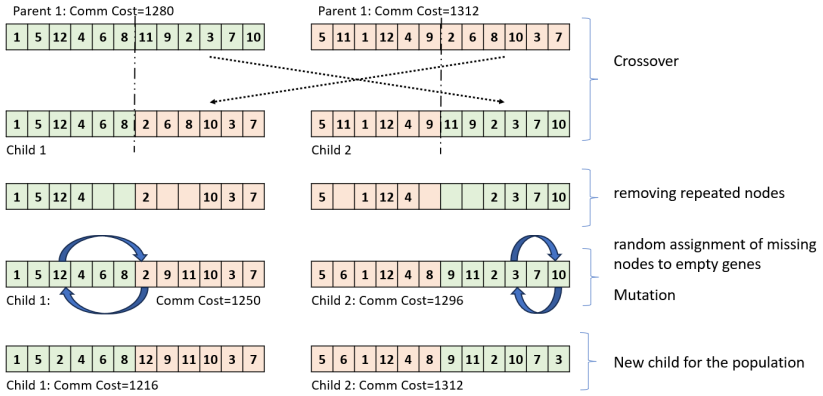

Figure 5: Mutation and crossover operations on two randomly selected chromosome pairs.

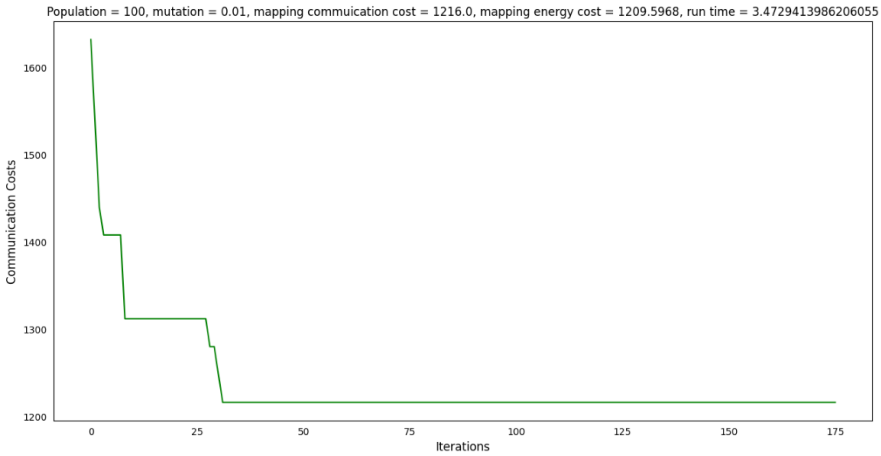

Figure 6: The convergence Curve for the Genetic algorithm applied to the Memory Write Delay (MWD) Benchmark.

1) the creation of the initial population, 2) the determination of the communication cost function, 3) the determination of the total communication energy function, and 4) the fitness function [2]. For creating the initial population, a random mapping is established by placing the cores onto the mesh in a random fashion. For illustrative purposes, the pseudo-code for the genetic algorithm is given in Algorithm 2.

A fitness function of the genetic algorithm for mapping is designed to estimate the quality of the solutions generated by the algorithm. It considers the objectives of minimizing energy consumption and communication costs while mapping the application onto the 3D mesh-based architecture. The fitness function calculates the cost of the mapping solution, which is derived from the hop count and the required bandwidth. The fitness of mapping is obtained by finding the inverse sum of the fitness of individual mapping in the population. The greater the fitness of a mapping, the greater the probability

Algorithm 2 Proposed Genetic Algorithm for Application Mapping on 3-D NoC

Core Graph G, Topology Graph T, Population size $Psize$
Mapping of G on to T, Best mapping M, Total Energy E, Minimum cost C
P = Initialize population $(G, Psize)$
Evaluate $Cbest, Mbest$
$C = Cbest$
$M = Mbest$
set external loop iteration count to $|n|^2$
For each member in P
Evaluate fitness. Inverse sum of $Cbest$
Select two fit parents randomly
Perform mutation and crossover
Add the child to the population
Evaluate $Cbest, Mbest$
Update $M, Comm$
Remove unfit members from the population
$E = Calculate\_Energy(M)$
$return(M, E, Comm)$

of becoming a parent. The genetic algorithm then seeks to optimize this cost function, finding solutions that minimize energy consumption and communication costs while still satisfying the constraints of the mapping problem. The convergence curve for the Genetic algorithm (GA) applied to the Memory Write Delay (MWD) Benchmark is shown in figure 6.

4.4 Computational complexity

This section covers a detailed explanation of the computational complexity associated with the proposed algorithms. The computational complexity of an algorithm is a measure of the time and memory required to execute the algorithm namely time complexity and space complexity. Since there are no constraints on the memory requirement, only time complexity is discussed in this work.

The time complexity of the CastNet algorithm depends on factors like time to obtain the highest priority node and the selection of the candidate for initial core placement. The time complexity of generating the priority list is $O(N(N+1)/2)$ represented as $O(U/2)$, where U is the number of nodes in each layer of a 3 x N x N mesh. The function get_candidate for initial core placement has a complexity $(1+2+3+....\sqrt{U}/2)$, which is given by $O(U)$. For each initial core mapping, we call select_task and select_core functions. The select_task function looks only at the edges of the mapped task set, with a time complexity of $O(E)$. The select_core function calculates the cost of mapping a task onto neighbouring cores of all the mapped cores. Hence, its time complexity is given by $O(UE)$. The overall time complexity of CastNet is approximated as $O(EU^2)$.

For the proposed Genetic algorithm factors like the population size, the number of generations, the crossover and mutation operators, and the fitness function evaluation affect the complexity of GAs. Here C is taken as the number of cores. The number of generations ie the stopping criteria for the algorithm is C^2 iterations. The time complexity

of calculating the total communication cost for a particular mapping is $O(C^2)$. The fitness function is calculated as the inverse sum of the total communication cost of each member of the population. The time complexity for evaluating the fitness of all individuals in a single generation is $O(P * C^2)$, where P is the number of individuals in the population. Considering all the generations, it becomes $O(P * C^2 * C^2)$. The time complexity for selecting the parents using the roulette wheel method is $O(P)$ per generation. Over C^2 generations, it becomes $O(P * C^2)$. The time complexity for performing crossover and mutation on the population is $O(P * C)$, where C is the chromosome length same as the core size. Over C^2 generations, it becomes $O(P * C * C^2)$. The overall time complexity of a genetic algorithm can be approximated as $O(P * C^4))$.

5 Experimental Setup

5.1 Selection of Simulation Platform and Hardware

The simulation for the experiment is run on a Windows 10 operating system powered by Intel i7 12th gen CPU running at 2.1GHz with 16GB LPDDR5 RAM and SSD m.2 nvme 512GB hard disk. The algorithm is written in Python and the IDE used is Microsoft Visual Studio code version 1.85.1.

5.2 Selection of mapping algorithm

The choice of algorithm depends on the specific problem characteristics and requirements. Owing to the simplicity and ease of implementation this work focuses on the evolutionary algorithm, Genetic algorithm (GA) [Masdari et al. 2023] and a meta-heuristic algorithm, CastNet algorithm [Tosun et al. 2011].

Genetic algorithms (GAs) are relatively simple to understand and implement compared to other evolutionary algorithms. They use straightforward operations like selection, crossover (recombination), and mutation, making them accessible for various applications. They are particularly useful for problems where the search space is large, complex, or poorly understood. They maintain a population of solutions, providing a good balance between exploring the search space and exploiting known best solutions which is crucial in NoC design, where the design space is large and complex.

CastNet algorithm is designed to generate optimized NoC topologies, which is crucial for minimizing latency and reducing power consumption. CastNet algorithm can handle large-scale designs, making them suitable for modern multi-core and many-core systems. They can optimise NoCs for heterogeneous environments, ensuring efficient communication between diverse components.

5.3 Topology Selection

To obtain the 3D Mesh structure, the 2D Mesh extended to form a 3D structure with a 3-layer of N×N mesh. Here ”Through Silicon Via” (TSV) technology is used to create the vertical links. This will significantly reduce the overall wire length required and improve the transmission efficiency. Hence, for our experiments, this study opts to utilize the 3D Mesh structure [Gan et al. 2021].

5.4 Routing Selection

Regarding the routing algorithm, XYZ routing algorithm is employed in this literature. This algorithm is chosen for its simplicity in implementation and since it is widely used for routing in 3D NoC architecture.

6 Result Analysis

6.1 Simulation results for CastNet Algorithm

In this experiment, the communication cost, energy consumption and CPU runtime of a given task graph for the two mapping algorithms, the CastNet and the genetic algorithm, are compared and analysed. We establish our bandwidth constraint for an application by selecting the maximum bandwidth which is the highest value among the edge weights in the Communication Task Graph as our bandwidth limitation.

Benchmark	Ref [2D]	2D-Mesh	3D-Mesh
VOPD	4135[Tosun et al. 2011]	4135	4119
MPEG-4	3852[Tosun et al. 2011]	3672	3773
MWD	1344[Tosun et al. 2011]	1312	1248
263 Dec	19.82[Tosun et al. 2011]	19.84	19.82
263 Enc	230.41[Tosun et al. 2011]	230.41	230.43
DVOPD	9784[Wang et al. 2020]	9765	9602
32N_0.1	–	876	722
32N_10	–	504151	404017
64N_0.1	–	2769	2064
64N_10	–	1771768	1340474
128N_0.1	–	6311	4622
128N_10	–	5909745	4079343

Table 3: Communication cost (hops X BW) Comparison of CastNet Algorithm.

For estimating the energy consumption of each mapping, we adopt the power/energy model outlined in reference [Ye et al. 2002]. This model quantifies energy consumption by considering the energy expended for each bit transmitted from the source to the destination core, considering factors such as switch energy, buffer energy, and interconnection wire energy. Notably, the calculation excludes the energy consumed by the cores, as our primary objective is to reduce the energy consumed by the network. The CPU run time constraint is evaluated by considering the time taken to obtain the best mapping for a particular benchmark application using both mapping algorithms. Even though Communication cost and energy consumption metrics are independent of the simulation platform and the hardware used for the experimental setup, they have a significant impact on the CPU runtime. Considering this all the simulations are run using the same experiment setup mentioned in section 5.1.

Benchmark	Ref [2D]	2D-Mesh	3D-Mesh
VOPD	4456.23[Tosun et al. 2011]	4082.71	4072.59
MPEG-4	3933.67[Tosun et al. 2011]	3685.67	3749.54
MWD	1253.79[Tosun et al. 2011]	1270.29	1229.83
263 Dec	20.43[Tosun et al. 2011]	20.27	20.26
263 Enc	236.24[Tosun et al. 2011]	236.27	236.29
DVOPD	–	9622.26	9519
32N_0.1	–	717.98	620.61
32N_10	–	382660.97	319346.25
64N_0.1	–	2123.48	1677.71
64N_10	–	1285305.88	1012598.69
128N_0.1	–	4682.22	3614.26
128N_10	–	4132402.85	2974739.67

Table 4: Energy Consumption (μJ) Comparison of CastNet Algorithm.

For energy consumption values, we rely on data from a 100-nm technology source, as documented in [Srinivasan et al. 2006]. Based on this reference, the energy estimate for the router input port is 328 nanojoules per megabit per second (nJ/Mb/s), while the output port's energy consumption is specified as 65.5 nanojoules per megabit per second (nJ/Mb/s). The physical link's consumed energy is calculated at 79.6 nanojoules per megabit per second (nJ/Mb/s) per link length (mm). In this work, a link length of 3 mm is taken when connecting two cores, as mentioned in [Marculescu et al. 2009].

Benchmark	2D-Mesh	3D-Mesh
VOPD	8	6
MPEG-4	11	16
MWD	16	34
263 Dec	12	38
263 Enc	12	166
DVOPD	219	362
32N_0.1	236	415
32N_10	321	623
64N_0.1	3423	6566
64N_10	4811	9524
128N_0.1	151547	207358
128N_10	198434	236708

Table 5: CPU run time (ms) comparison of CastNet Algorithm.

We present the outcomes of our experiments conducted with the CastNet algorithm in Tables 3 - 5. Table 3 compares the total bandwidth requirements (TCC) of CastNet-generated mappings for various benchmark applications, with columns 2 to 4 presenting these comparisons. In contrast, Table 4 is dedicated to comparison TEC (total energy consumptions) for the best mappings solution created by the CastNet algorithm across different benchmark applications. In column 2, we provide data sourced from the literature reviews. Columns 3 and 4 display the results achieved on 2D and 3D mesh architectures.

Table 5 presents a comparison of CPU runtime for the best mapping solution across various benchmark applications using the CastNet mapping algorithm. Since this metric is heavily dependent on the experimental setup, making comparisons with existing works may not yield meaningful results. Instead, this paper attempts to compare CPU runtime with 2D and 3D mesh NOC architecture.

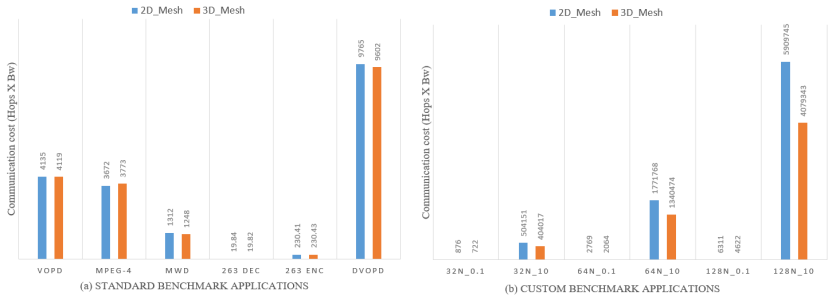

Figure 7: Comparison of Communication cost for CastNet Algorithm using different benchmarks on 2D-Mesh and 3D-Mesh architecture. (a) Standard benchmark applications. (b) Custom benchmark applications.

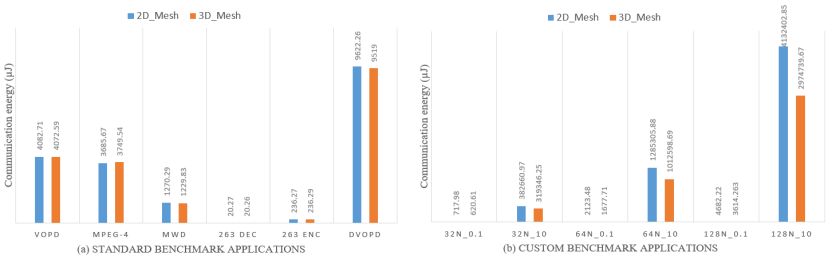

Figure 8: Comparison of Communication energy in µJ for CastNet Algorithm using different benchmarks on 2D-Mesh and 3D-Mesh architecture. (a) Standard benchmark applications. (b) Custom benchmark applications.

The simulation results for the implementation of the CastNet Algorithm on both 2D and 3D Mesh are depicted in Figures 7-9. In Figure 7, the communication costs are compared for 2D and 3D mesh architectures, while Figure 8 illustrates a comparison of communication energy between the two architectures. Finally, Figure 9 presents a comparison of CPU runtime for 2D and 3D mesh architectures. It's worth mentioning that when applied to 3D mesh architectures with larger benchmark sizes, CastNet performs comparably well.

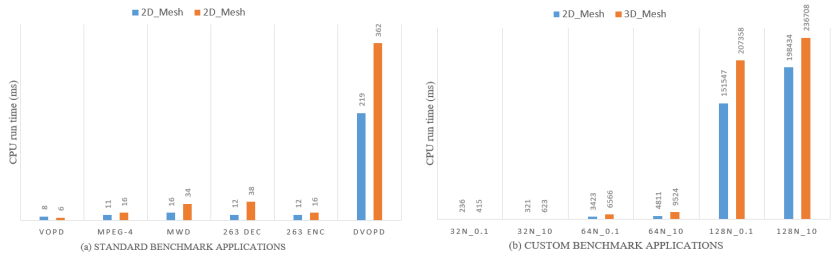

Figure 9: Comparison of CPU run time in ms for CastNet Algorithm using different benchmarks on 2D-Mesh and 3D-Mesh architecture. (a) Standard benchmark applications. (b) Custom benchmark applications.

For benchmark applications with a limited number of nodes, the implementation on both 2D and 3D mesh architectures yields similar results. The simulation results were also classified based on the nature of the benchmark used. Simulation results using standard benchmarks are represented in Figure 7-9 (a), whereas results obtained for synthetic benchmark applications are represented in Figure 7-9 (b).

6.2 Simulation results for Genetic Algorithm

In this section, we compare the outcomes of the experiments conducted with the Genetic algorithm. We focus on task graphs that consist of varying numbers of IP cores. For the Genetic algorithm a population size of 100 is selected with mutation probability set as 0.01. The algorithm is run on both standard and synthetic benchmarks to generate the best mapping solutions. Results for 10 iterations where obtained, after which the average value is computed. The best, average and worst solutions of each of the benchmark applications out of the 10 iterations is included in this work. The experimental findings, comparing bandwidth requirements, energy consumption and CPU runtime for the Genetic algorithm implemented on both 2D and 3D mesh architectures, are presented in Tables 6 - 8.

Benchmark	**Communication cost**			**Communication Energy**		
	Average	**Maximum**	**Minimum**	**Average**	**Maximum**	**Minimum**
VOPD	4028.8	4174	3971	3949.06	4040.87	3912.51
MPEG-4	3710.55	3881	3567.5	3710.05	3817.83	3619.6
MWD	1350.4	1472	1248	1294.58	1371.47	1229.83
263 DEC	19.46	19.64	19.27	19.83	19.95	19.72
263 ENC	82.89	82.94	82.87	84.81	85.03	82.94
DVOPD	10846.4	12155	9723	10061.27	10888.7	9350.94
32N_0.1	956.6	1015	922	768.95	805.87	747.07
32N_10	505024	523362	496848	383212.97	394808.09	378043.29
64N_0.1	2894.2	3315	2654	2202.65	2468.72	2050.77
64N_10	1759497	1788098	1730895	1277546.6	1295631.3	1259461.9

Table 6: Communication cost (hops X BW) and Energy Consumption (μJ) for mapping using Genetic Algorithm on the 2D mesh.

Benchmark	Communication cost			Communication Energy		
	Average	Maximum	Minimum	Average	Maximum	Minimum
VOPD	4113.8	4374	3891	4002.8	4167.33	3861.93
MPEG-4	3728.1	3792	3631	3721.15	3761.55	3659.75
MWD	1267.2	1312	1216	1241.97	1270.3	1209.6
263 DEC	19.44	20.02	19.25	19.82	20.19	19.7
263 ENC	82.88	82.88	82.87	85.01	85.02	85.01
DVOPD	10552.6	11648	9251	9875.5	10568.12	9052.5
32N_0.1	835.1	886	768	692.12	724.31	649.7
32N_10	418276	429434	408035	328362.2	335417.4	321886.8
64N_0.1	2376.8	2449	2335	1875.49	1921.15	1849.06
64N_10	1332251	1352295	1308846	1007399	1020073	992600.3

Table 7: Communication cost (hops X BW) and Energy Consumption (uJ) for mapping using Genetic Algorithm on 3D mesh.

Benchmark	2D-mesh	3D-mesh
VOPD	5.87	13.442
MPEG-4	1.93	1.12
MWD	1.84	3.473
263 DEC	2.179	7.19
263 ENC	1.77	0.994
DVOPD	75.37	175.718
32N_0.1	282.24	380.19
32N_10	247.54	493.99
64N_0.1	4564.64	6067.41
64N_10	8253.68	17642.13

Table 8: CPU run time (sec) comparison of Genetic Algorithm.

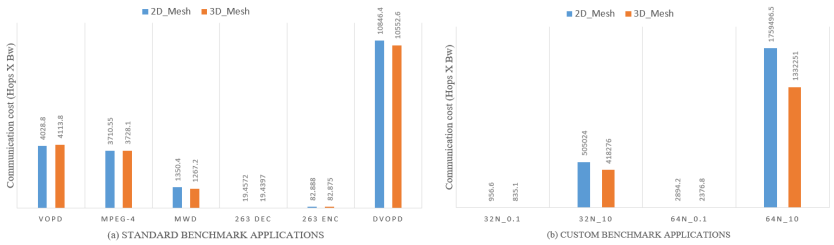

Figure 10: Comparison of Communication cost for Genetic Algorithm using different benchmarks on 2D-Mesh and 3D-Mesh architecture. (a) Standard benchmark applications. (b) Custom benchmark applications.

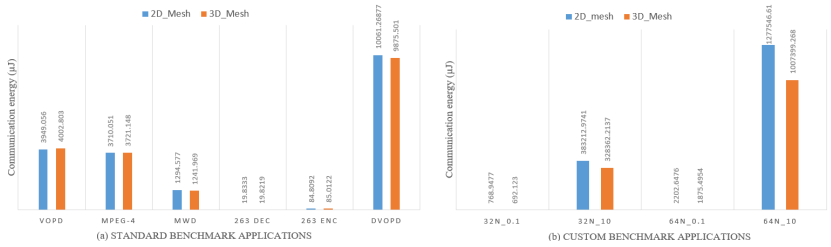

Figure 11: Comparison of Communication energy in µJ for Genetic Algorithm using different benchmarks on 2D-Mesh and 3D-Mesh architecture. (a) Standard benchmark applications. (b) Custom benchmark applications.

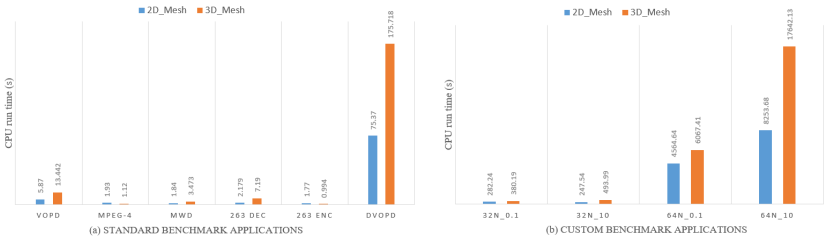

Figure 12: Comparison of CPU run time in ms for Genetic Algorithm using different benchmarks on 2D-Mesh and 3D-Mesh architecture. (a) Standard benchmark applications. (b) Custom benchmark applications.

The outcomes of applying the Genetic Algorithm to both 2D and 3D mesh configurations are presented in Figures 10 - 12. Figure 10 compares communication costs in 2D and 3D mesh architectures, while Figure 11 illustrates a comparison of communication energy between these two setups. Finally, Figure 12 presents a comparison of CPU runtime for 2D and 3D mesh architectures. Similar to the analysis observed for simulations using the CastNet Algorithm, the Genetic Algorithm also demonstrates commendable performance when employed in 3D mesh architectures with larger benchmark sizes.

For benchmark applications with a limited number of nodes, the implementation on both 2D and 3D mesh architectures yields comparable results. Additionally, same as in the case for CastNet Algorithm, the simulation results are categorized based on the nature of the benchmark utilized. Results obtained using standard benchmarks are shown in Figure 10 - 12 (a), while those from synthetic benchmark applications are presented in Figure 10 - 12 (b).

6.3 Comparison of Simulation results

In this section, a comparison of simulation results of GA (Genetic Algorithm) and CastNet Algorithm is done for both 2D and 3D mesh architecture. While comparing the performance metrics of GA and CastNet across different benchmarks and node sizes in both 2D and 3D mesh, several trends emerge. Generally, GA demonstrates lower

communication costs and energy consumption at the expense of greater CPU runtime compared to CastNet. In the 3D mesh architecture, while GA still outperforms CastNet in terms of communication costs and energy consumption for most of the benchmark applications, the margin of difference appears to be narrower compared to the 2D mesh architecture.

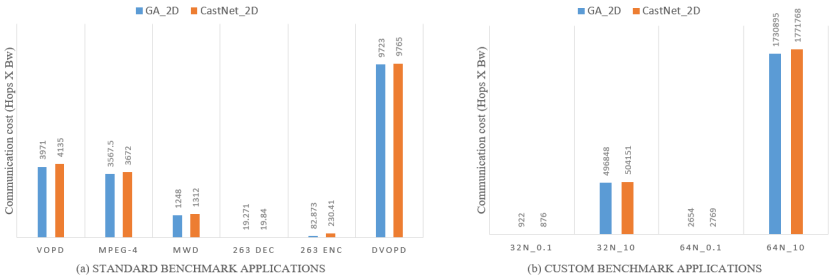

Figure 13: Comparison of communication cost (hops X BW) for running benchmark application using Genetic and CastNet Algorithm on 2D-Mesh. (a) Standard benchmark applications. (b) Custom benchmark applications.

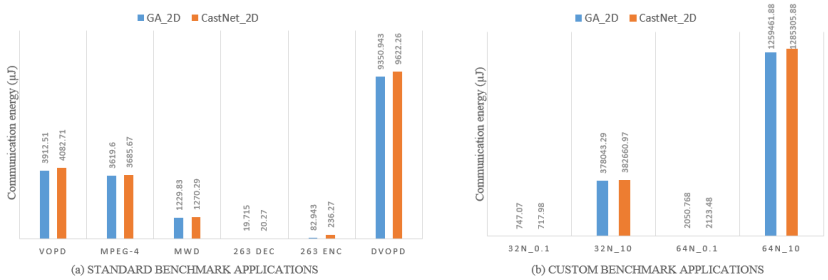

Figure 14: Comparison of communication energy (µJ) for running benchmark application using Genetic and CastNet Algorithm on 2D-Mesh. (a) Standard benchmark applications. (b) Custom benchmark applications.

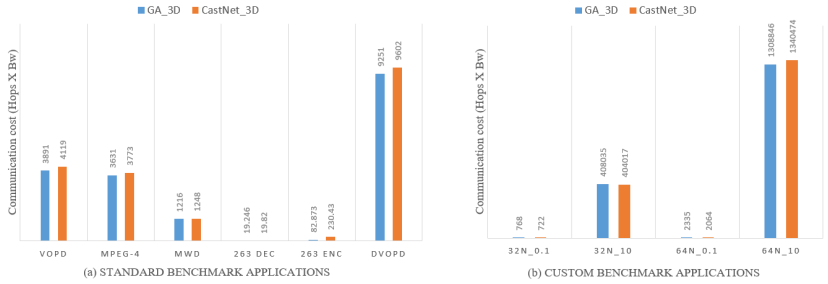

Figure 15: Comparison of communication cost (hops X BW) for running benchmark application using Genetic and CastNet Algorithm on 3D-Mesh. (a) Standard benchmark applications. (b) Custom benchmark applications.

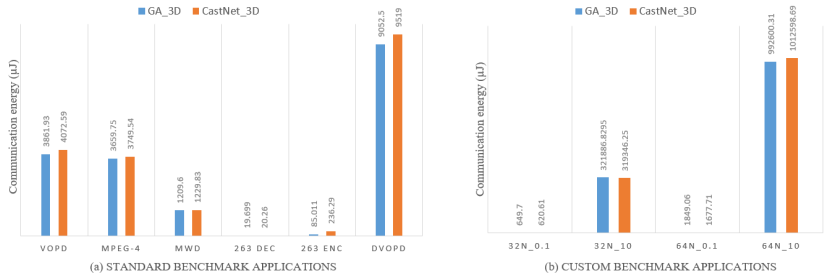

Figure 16: Comparison of communication energy (µJ) for running benchmark application using Genetic and CastNet Algorithm on 3D-Mesh. (a) Standard benchmark applications. (b) Custom benchmark applications.

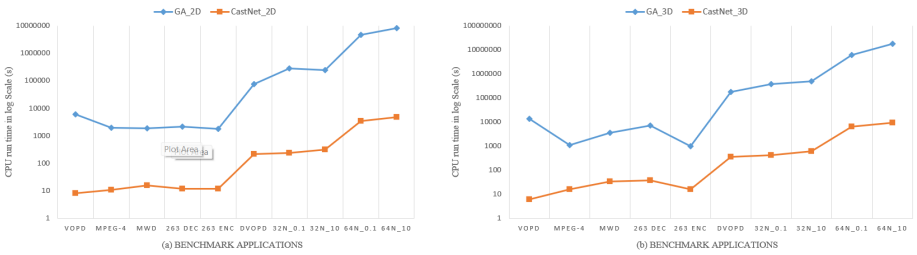

Figure 17: Comparison of CPU run time in log scale running benchmark application using Genetic and CastNet Algorithm on (a) 2D-Mesh and (b) 3D-Mesh.

7 Conclusion

The study addresses the critical issue of optimizing the mapping of applications onto a 3-layer 3D Network-on-Chip (NoC) architecture. Traditional 2D NoC architectures are limited in their ability to efficiently handle the increasing demands for high performance

and low power consumption in modern computing systems. As the number of cores in these systems increases, so do the challenges related to communication cost, energy consumption, and CPU run time. Therefore, this research aims to develop effective algorithms to enhance the efficiency of 3D NoC systems, which can potentially offer better performance than their 2D counterparts. The proposed solution to tackle these challenges introduces two mapping algorithms: a heuristic algorithm named CastNet_3D and a pseudo-heuristic algorithm named GA_3D. These algorithms are designed to map applications onto a 3-layer 3D NoC architecture with the main objectives of reducing CPU run time, energy consumption, and communication cost.

The performance of the proposed algorithms was evaluated through extensive simulations on both standard and synthetic benchmarks. These benchmarks were applied to 2D-mesh and 3D-mesh NoC architectures to compare the effectiveness of the proposed 3D mapping algorithms against traditional 2D mapping approaches. The simulation results indicate significant improvements in key performance metrics for the proposed algorithms: Communication Cost: On average, the proposed 3D mapping algorithms can reduce the total communication cost by up to 10% compared to existing 2D mapping algorithms. Energy Consumption: Both CastNet_3D and GA_3D demonstrated substantial improvements in reducing energy consumption. The 3D architecture's ability to shorten communication paths through vertical connections (enabled by TSV technology) contributes to these energy savings. CPU Run Time: While both algorithms achieve significant improvements in communication cost and energy consumption, the GA_3D algorithm requires more time to arrive at the best possible solution due to its pseudo-heuristic nature.

The potential applications of these algorithms are significant. They can optimize task mapping and resource allocation in high-performance computing (HPC) systems that utilize Network-on-Chip (NoC) architectures. Furthermore, these algorithms can enhance the efficiency of routing and resource allocation in telecommunication hardware that employs NoC technologies. Despite these potential advantages and applications, several limitations and challenges exist. Implementing these algorithms in Network-on-Chip (NoC) systems is computationally complex, potentially leading to increased development time and resource allocation. Scaling NoC systems with genetic algorithms (GAs) to larger networks presents challenges due to heightened computational requirements and the potential for increased latency. Additionally, genetic algorithms may necessitate a considerable amount of time to converge to optimal or near-optimal solutions. Ensuring that the CastNet algorithm quickly converges to optimal solutions is particularly challenging in dynamic environments. Moreover, managing heterogeneous cores and varied task requirements within NoC systems adds a layer of complexity to the process.

In this work, there is some scope for improvement. Firstly, bandwidth limitation on the links is not considered in this work. Also, we could use other deadlock-free routing protocols [Gabis B. et al. 2016] instead of XYZ routing. Similarly, our experiment focuses on homogeneous cores where any task can be mapped onto any cores. This work can also be extended to heterogeneous cores [Yang et al. 2024] where a task can only be mapped onto a specific set of cores. As these considerations significantly increase the computational complexity, we omitted them in this work.

Acknowledgements

The authors express sincere gratitude to all those who contributed to the completion of this work. We deeply acknowledge the Department of Electronics and Communication

Engineering, Rajagiri School of Engineering & Technology, APJ Abdul Kalam Technical University, Kerala, India, for their valuable support in conducting the research. We also wish to thank our colleagues at Rajagiri School of Engineering & Technology, whose collaboration and input have enriched our research process. Their contributions, whether through discussions, critiques, or technical assistance, have been invaluable to this project.

References

[Ankur et al. 2022] Ankur Gogoi, Bibhas Ghoshal, Akash Sachan, Rakesh Kumar, Kanchan Manna, Application driven routing for mesh-based Network-on-Chip architectures, Integration, Volume 84, 2022, Pages 26-36, ISSN 0167-9260.

[Boveiri, Hamid Reza 2018] Boveiri, Hamid Reza , "125 random task-graphs for multiprocessor task scheduling", Mendeley Data, V2, 2018. http://doi.org/10.17632/4fycv9td56.2

[Gabis B. et al. 2016] Benmessaoud Gabis, A., Koudil, M., NoC routing protocols – objective-based classification, J. Syst. Archit. 66–67 (2016) 14–32. https://doi.org/10.1016/j.sysarc.2016.04.011.

[Gan et al. 2021] Gan, Y., Guo, H., Zhou, Z. 3D NoC Low-Power Mapping Optimization Based on Improved Genetic Algorithm. Micromachines. 2021; 12(10):1217. https://doi.org/10.3390/mi12101217.

[Gu et al. 2024] Gu, L., Mohajer, A. Joint throughput maximization, interference cancellation, and power efficiency for multi-IRS-empowered UAV communications. SIViP 18, 4029–4043 (2024). https://doi.org/10.1007/s11760-024-03015-5.

[Hanna et al. 2018] He Hanna, Fang Fang, and Wei Wang. "Improved simulated annealing genetic algorithm based low power mapping for 3D NoC." MATEC web of conferences. Vol. 232. EDP Sciences, 2018.

[Jingcao et al. 2003] Jingcao Hu and Marculescu, R., "Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures," 2003 Design, Automation and Test in Europe Conference and Exhibition, Munich, Germany, 2003, pp. 688-693.https://doi.org/10.1109/DATE.2003.1253687.

[Mandelli et al. 2011] Mandelli, M., Ost, L., Carara, E., Guindani, G., Gouvea, T., Medeiros, G., Moraes, F. G., Energy-aware dynamic task mapping for NoC-based MPSoCs, in: 2011 IEEE International Symposium of Circuits and Systems (ISCAS), IEEE, 2011, pp. 1676–1679.

[Marculescu et al. 2009] Marculescu, R., Ogras, U. Y., Peh, L. S., Jerger, N. E. and Hoskote. Y., "Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 28, no. 1, pp. 3-21, Jan. 2009. http://doi.org/10.1109/TCAD.2008.2010691.

[Masdari et al. 2023] Mohammad Masdari, Sultan Noman Qasem, Hao-Ting Pai. Optimizing Network-on-Chip using metaheuristic algorithms: A comprehensive survey, Microprocessors and Microsystems, Volume 103, 2023, 104970, ISSN 0141-933.https://doi.org/10.1016/j.micpro.2023.104970.

[Mohajer et al. 2022] A. Mohajer, F. Sorouri, A. Mirzaei, A. Ziaeddini, K. J. Rad and M. Bavaghar, "Energy-Aware Hierarchical Resource Management and Backhaul Traffic Optimization in Heterogeneous Cellular Networks," in IEEE Systems Journal, vol. 16, no. 4, pp. 5188-5199, Dec. 2022. https://10.1109/JSYST.2022.3154162.

[Nalci et al. 2021] Nalci, Y., Kullu, P., Tosun, S. et al. ILP formulation and heuristic method for energy-aware application mapping on 3D-NoCs. J Supercomput 77, 2667–2680 (2021). https://doi.org/10.1007/s11227-020-03365-0.

[Pradeep et al. 2019] Pradeep Kumar Sharma, Santosh Biswas, Pinaki Mitra, Energy efficient heuristic application mapping for 2-D mesh-based network-on-chip, Microprocessors and Microsystems, Volume 64, 2019, Pages 88-100, ISSN 0141-9331.https://doi.org/10.1016/j.micpro.2018.10.008.

[Pradip et al. 2013] Pradip Kumar Sahu, Santanu Chattopadhyay, A survey on application mapping strategies for Network-on-Chip design, Journal of Systems Architecture, Volume 59, Issue 1, 2013, Pages 60-76, ISSN 1383-7621. https://doi.org/10.1016/j.sysarc.2012.10.004.

[Sahu et al. 2014] Sahu, P. K., Shah, T., Manna, K. and Chattopadhyay, S., "Application Mapping Onto Mesh-Based Network-on-Chip Using Discrete Particle Swarm Optimization," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 22, no. 2, pp. 300-312, Feb. 2014. https://doi.org/10.1109/TVLSI.2013.2240708.

[Srinivasan et al. 2006] Srinivasan, K., Chatha, K. S. and Konjevod, G., "Linear-programming-based techniques for synthesis of network-on-chip architectures," in IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, vol. 14, no. 4, 2006, pp. 407-420.http://doi.org/10.1109/TVLSI.2006.871762.

[Tosun et al. 2011] Tosun, S., "New heuristic algorithms for energy aware application mapping and routing on mesh based NoCs", Journal of Systems Architecture, Volume 57, Issue 1, 2011, Pages 69-78, ISSN 1383-7621. https://doi.org/10.1016/j.sysarc.2010.10.001.

[Tosun et al. 2018] Tosun, S., Ajabshir, VB. Energy-aware partitioning of fault-tolerant irregular topologies for 3D network-on-chips. J Supercomput 74(9): 2018, pp. 4842–4863

[Wang et al. 2020] Wang, W., Choi, T. M., Yue, X., Zhang, M. and Du, W., "An Effective Optimization Algorithm for Application Mapping in Network-on-Chip Designs," in IEEE Transactions on Industrial Electronics, vol. 67, no. 7, pp. 5798-5809, July 2020, http://dx.doi.org/10.1109/TIE.2019.2926043.

[Yang et al. 2024] Yang, T., Sun, J. Mohajer, A. Queue stability and dynamic throughput maximization in multi-agent heterogeneous wireless networks. Wireless Netw 30, 3229–3255 (2024). https://doi.org/10.1007/s11276-024-03730-4.

[Ye et al. 2002] Ye, T. T., Benini, L. and De Micheli, G., "Analysis of power consumption on switch fabrics in network routers," Proceedings 2002 Design Automation Conference (IEEE Cat. No.02CH37324), New Orleans, LA, USA, 2002, pp. 524-529.http://doi.org/10.1109/DAC.2002.1012681.

[Zhang et al. 2009] Zhang, W., Hou, L., Wang, J., Geng, S., Wu, W., Comparison research between XY and odd-even routing algorithm of a 2-dimension 3X3 mesh topology networkon-chip, in: 2009 WRI Global Congress on Intelligent Systems, Vol. 3, 2009, pp. 329–333.http://dx.doi.org/10.1109/GCIS.2009.110.

[Zhen et al. 2013] Zhen, Y., Marsono, M. N., Shaikh Husin, Shaikh-Husin, N., Yuan, W. H. (2013). Network partitioning and GA heuristic crossover for NoC application mapping. Proceedings - IEEE International Symposium on Circuits and Systems. 1228-1231. https://doi.org/10.1109/ISCAS.2013.6572074.