

Gauge influence of application mapping algorithms on NoC embedded applications with respect to energy and latency

Shweta Ashtekar¹, Kushal Tuckley²

¹Department of Electronics and Telecommunication Engineering, Ramrao Adik Institute of Technology, Dr D Y Patil deemed to be University, Navi Mumbai, India

²Department of Electrical Engineering, Indian Institute of Technology, Mumbai, India

Article Info

Article history:

Received Nov 26, 2024

Revised Jul 3, 2026

Accepted Jul 18, 2026

Keywords:

Application mapping
Communication sensitive
Embedded application
Latency and energy
optimization
Memory sensitive
Network on chip
Prediction using machine
learning

ABSTRACT

To satisfy efficient requirements of real time embedded applications, high speed communication platforms are needed in multicore system on chip (SoC) architectures and network on chip (NoC) proves promising packet switched interconnection solution. These applications can be either compute, communication or memory sensitive. This paper analyzes application mapping techniques near optimal mapping (NMAP), branch and bound (BB), particle swarm optimization (PSO), gaussian fusion/flow/field (GFF), and random to identify the most appropriate mapping technique for specific application considering tradeoff between latency and energy. For this, machine learning (ML) model has been evaluated on dataset generated from the simulator with 12 different applications. For compute sensitive applications NMAP and PSO, for communication sensitive applications BB and for memory sensitive applications GFF are appropriate mapping algorithms. Also, by varying buffer size it has been noted that for example MWD application with BB mapping algorithm, when buffer size is increased from 2 flits to 4,8,16 flits, latency is reduced by 58.84%, 75%, and 99%, respectively. In contrast, a router with 4 flits/buffer uses nearly the same amount of energy as router with 2 flits/buffer, and routers with 8 flits/buffer and 16 flits/buffer use 11.5% and 66% more energy, respectively. This analysis may benefit researchers conducting further research on the embedded application.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Shweta Ashtekar

Department of Electronics and Telecommunication Engineering, Ramrao Adik Institute of Technology

Dr D Y Patil deemed to be university

Navi Mumbai, India

Email: Shweta.ashtekar@rait.ac.in

1. INTRODUCTION

The thriving demand as well as complexity of real time embedded applications is satisfied by scalable and flexible system on chip (SoC) architecture that integrates multiple cores on a single chip for increased performance. To overcome the shortcomings of traditional bus based approach of SoC systems, a new paradigm network on chip (NoC) has been opted as packet switched communication fabric that effectively makes communication between number of cores [1]. NoC is capable of supporting variable traffics requirements such as synthetic or embedded and gained significant contribution in various domains of multimedia applications, internet of thing (IoT), robotics, and autonomous vehicles. Basic architecture of NoC system as shown in Figure1 that consists of various basic building blocks as core or processing element (PE) that can be either

central processing unit (CPU), graphics processing unit (GPU), digital signal processing (DSP) or memory block. The second block which works as router implementing routing [2] or communication protocol for the flow of data between source to destination core. The third block is network interface (NI) which connects core with network and does packet data conversions. For core placements on a single chip, NoC supports different topologies like Mesh, Torus, Tree, or custom designs. There are various challenges related with NoC systems as routing, switching, application mapping, scheduling, and quality of service (QoS) [3].

As seen in Figure 2, one of the most important issues when mapping application tasks on the NoC platform is application mapping. Every application consists of several tasks and communication between them. Any NoC system's performance is enhanced and its resources are properly utilized when several jobs are mapped [4] for any given application. Application mapping methods can be roughly categorized as either dynamic, adaptive, or run time mapping, or static, nonadaptive, or design time mapping [5]. It is also possible to develop hybrid mapping that incorporates both. The mapping in the dynamic approach [6] relies on the traffic situation at the moment by allocating the tasks to available cores. This ensures that packets and flits are reliably delivered to their destination, but moving between different jobs may result in additional overhead. Static approaches are less appropriate in unexpected scenarios, although they often execute only once, resulting in lower overhead, comparatively lower latency, and energy usage. The two primary types of static or nonadaptive mapping techniques are exact mapping and search-based mapping [7].

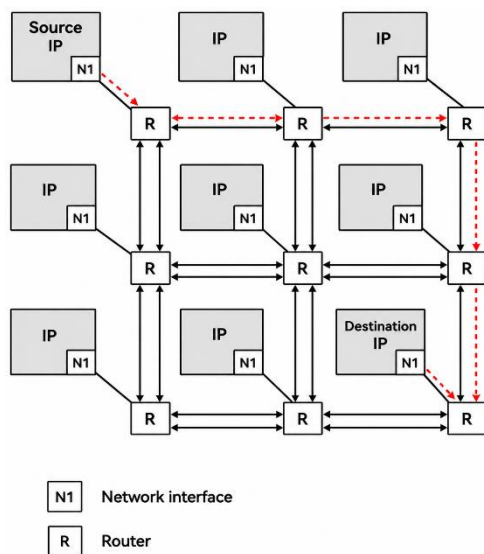


Figure 1. A typical 3x3 mesh NoC architecture [5]

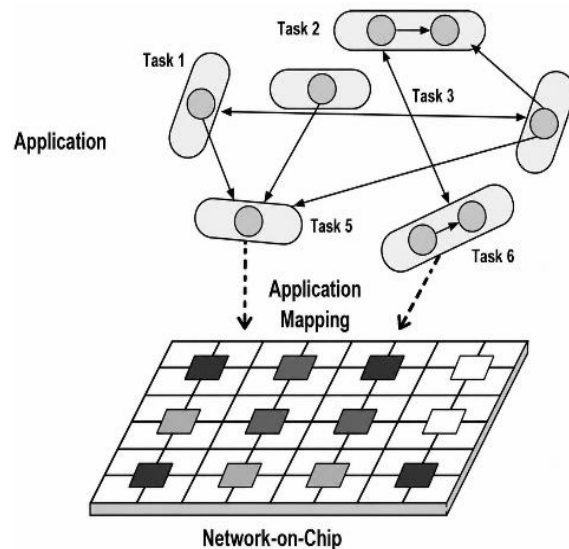


Figure 2. Application mapping process [3]

Different real time embedded applications can be classified into three categories as computation sensitive (multimedia applications), communication sensitive (networking applications), and memory sensitive (audio/video processing applications). Some applications are combination of above three categories also. Major factors affecting overall performance of these applications are large volume data, lots of computations involved because of compression/decompression, encryption/decryption, more memory requirements, and limited resources on implemented systems [8].

Some of these applications require high performance or minimum latency but result into increased energy [9] while some applications will compromise for increased latency but require to consume less energy saving resources. Usually compute sensitive applications utilize more core usage so their objective is to consume less energy due to more frequency of task switching between the cores will result into increased energy consumption. whereas communication sensitive applications utilize more routers so their objective must be to minimize the latency when data is being transferred between the number of tasks [10].

The main focus of this paper is to evaluate different application mapping algorithms on various embedded benchmarks and then as per the classification of these applications as either compute, communication, memory sensitive or hybrid approach and their corresponding requirements such as latency or energy minimisation, the best appropriate mapping algorithm will be suggested by running machine learning (ML) model on the dataset generated by the NoC simulator [11].

Various application mapping algorithms as near optimal mapping (NMAP) that works on constructive heuristic search mapping approach, branch and bound (BB) that works on systematic search approach, particle swarm optimization (PSO) that works on [12] transformative heuristic search approach, gaussian fusion/flow/field (GFF) that works on nature inspired metaheuristic approach [13] and random algorithm are considering for running on 12 different either multimedia, networking or signal processing embedded applications. This paper focuses on a trade off between latency and energy [9] to find the best possible mapping algorithm. Paper [14] has proposed an energy-efficient, high-performance NoC architecture with an optimized mapping solution for deep neural network (DNN) applications, aiming to reduce communication energy and latency while improving overall NoC performance.

This paper is articulated as follows: section 2 presents the literature review. Section 3 discusses the proposed methodology. Section 4 gives implementation details as well as carries result analysis. Section 5 concludes the paper.

2. LITERATURE REVIEW

Due to data transfer being their primary role, networking applications are typically communication sensitive that involves mainly two tasks as data transmission and packet handling. Data transmission over networks is a core architectural element of networking applications. Applications like load balancers, routers, and firewalls that oversee substantial amounts of network traffic are extremely sensitive to the effectiveness and speed of network communication. They may, nevertheless, also be memory- and compute-sensitive when handling buffering, caching, and several active connections. Examples of these tasks include encryption.

Multimedia applications are both memory and computation sensitive, with the dominant requirement varying by application and task. For example, processing large, high-resolution files may require greater memory capacity.

Heavy computing power is needed for tasks like real-time effects, 3D rendering, image processing, and video encoding. These jobs heavily rely on the performance of the CPU and, occasionally, the GPU. Software for editing videos, for instance, that uses transitions or filters, will require lot of processing power.

Applications for signal processing might be compute- or memory-sensitive, depending on the architecture and particular task at hand. The computational burden is high in applications where real-time performance is essential, such audio processing or telecommunications [6]. To process massive amounts of data in real time, these systems need to be able to process information quickly. When working with high-resolution signals or sizable datasets, algorithms such as fast fourier transforms (FFT), filtering, and convolution demand a significant amount of processing.

The two main categories of application mapping strategies are dynamic/adaptive/run time mapping and static/nonadaptive/design time mapping [5], [6]. It is also possible to develop hybrid mapping that combines the two. Task allocation and reordering have been done during the execution or running of the programs in a dynamic or adaptive mapping method. When there are fluctuating traffic loads, faults, congestion, or any temperature changes, these adaptive approaches are necessary. While this method may result in increased overhead when switching between several activities, it ensures that packets or flits reach their destination reliably by distributing the tasks on free cores based on the current traffic conditions [5]. This could lead to additional execution delays and higher energy consumption because there is rerunning of tasks is required. Dynamic algorithms are further classified as reliability aware, thermal aware and congestion aware.

The allocation of jobs for an application is determined during design time solely taking into account the initial underlying network architecture or available resources in a static or nonadaptive mapping approach [6]. These methods are less appropriate under unexpected circumstances, but they often only run once, which reduces overhead, latency, and energy usage.

Search-based mapping and exact mapping are two more categories for static mapping approaches. Exact mapping is better for NoC architectures with fewer activities because it requires extensive mathematical computations and programming to reach the precise optimal solution. In order to reduce intertile network conflict, the integer linear programming (ILP) formulation has been proposed in Yan *et al.* [15] that analyzes network contention. An accurate application mapping called ILP is suggested for NoC systems in order to obtain the best or most near-optimal solution within the constraints of computational time. Using the idea of hardware/software codesign for automated mapping on application-specific integrating systems that analyze images and signals within specified timeframes, a mixed integer linear programming (MTLP) model has been implemented in a [16] study. There are two types of search-based mapping: i) systematic and deterministic search and ii) heuristic search.

BB is one algorithm example of a systematic search strategy. It bounds unallowable solutions to systematically examine the complete search space of potential solutions before delivering an ideal answer. The hybrid branch and bound based exact mapping (BEMAP) algorithm, which combines BB map with precise systematic search to obtain a multi-objective solution in terms of throughput and cost, was published in the work

[17]. In general because they explore multiple potential solutions to arrive at the optimal solution compared to mathematical approximations and are based on population or swarm intelligence, heuristics [18] or metaheuristic algorithms are either transformative, like the genetic algorithm (GA) [19], particle swarm optimization (PSO) [12], or constructive, like NMAP. They are typically self-learning and inspired by nature, having been inspired by the natural intelligence of a variety of animals and birds. There are many heuristic/metaheuristic algorithms, as the whale optimization algorithm (WOA) [7], which mimicked the bubble-net foraging, encircling prey, and prey-seeking behaviors of humpback whales using three operators. This study was applied to 29 different mathematical benchmarks in order to analyse the algorithm's convergence. An optimization approach in paper [20] improves the standard firefly algorithm through enhanced movement and attraction strategies helping to reduce communication cost and network latency.

To optimize task placement on the NoC cores, a metaheuristic Cuckoo search via Levy flight is proposed in paper [21]. Ant colony optimization (ACO) is a population-based probabilistic approach that draws inspiration from the fact that ants naturally follow a single path when seeking food [22]. By adding ML algorithms to hybrid mapping techniques, performance can be improved [23].

3. SUGGESTED APPROACH FOR MAPPING OPTIMIZATION

This section describes about the evaluation parameters latency and energy, the basic working of mentioned mapping algorithms and how depending on requirement of specific application, particular mapping algorithm can be best suitable in order to have trade off between latency and energy.

3.1. Definitions

3.1.1. Definition 1: directed application task or communication graph

As every application comprises of multiple required tasks and communication between the tasks so an application having certain number of n tasks as $(t_1, t_2, t_3, \dots, t_n)$ can be represented using the directed application task graph as DAG (T, C) where T represents the number of vertices each representing single task whereas C represents weighted edge $(C_{i,j})$ or link between any two tasks i and j [9]. The weight is communication volume or bandwidth between set of tasks. For example, application multi window display (MWD) has $T=12$ and $C=13$.

3.1.2. Definition 2: directed core or architecture or topology graph

The NoC architecture is composed certain amount of processing cores on which the application tasks needs to efficiently mapped. The topological architecture of NoC is represented using directed core graph as DCG (P, E) where P signifies set of processing cores $(P_1, P_2, P_3, \dots, P_n)$ whereas E symbolizes set of edges $(E_{i,j})$ presenting communicating between these cores or tiles.

3.2. Evaluation parameters

Among the multiple evaluation parameters like latency, throughput, energy and power. In this paper two parameters latency and energy are considered.

3.2.1. Average latency

Let L_{ij} be the packet latency of packet j and N_i be the number of packets received by processor i (after the warmup time), then the average network latency is given by (1):

$$L_{avg} = \frac{1}{N} \sum_{i=1..N} \left(\frac{1}{N_i} \sum_{\forall j} L_{ij} \right) \quad (1)$$

where N is the number of processors in the platform.

3.2.2. Average energy

Average energy dissipated by each router after warming up is given by (2):

$$E_{avg} = P_{avg}(T_{sim} - T_{warm}) \quad (2)$$

where P_{avg} is average power and T_{sim} is simulation time, and T_{warm} is warm up time.

Different algorithms are used for mapping and optimization in NoC as NMAP, BB, and PSO, GFF. A brief review of these methods is provided here.

For optimization, all the application mapping algorithms work on communication cost minimization. The communication cost is calculated as (3):

$$Cost = \sum [B_{ti,tj} \times N_{i,j}] \quad (3)$$

where $B_{ti,tj}$ denotes communication bandwidth between any two tasks i and j . Also $N_{i,j}$ represents the Manhattan distance between pair of nodes represented with coordinates as (x_i, x_j) as well as (y_i, y_j) given as (4):

$$N_{i,j} = |x_i - x_j| \times |y_i - y_j| \quad (4)$$

3.3. An embedded application example

One of the video processing application used for evaluation is picture in picture (PIP). This application has 8 tasks and related task flow graph with edge weights shown on edges connecting the tasks as shown in Figure 3. They are important for calculating communication cost and network latency.

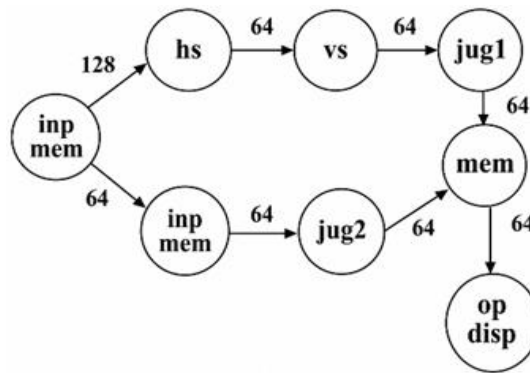


Figure 3. PIP application task flow graph

4. EVALUATION

This section gives the details of implantation for executing various application mapping algorithms on different embedded applications on the simulation platform. After evaluation best possible mapping algorithm suitable for that application has been suggested by considering trade of between latency and energy [19].

4.1. Simulation environment

To implement and evaluate the performance of different embedded applications, the NoCTweak Simulator [24] based on system C/C++ is used. The results are carried on (8×8) 2D mesh for implementation total 12 multimedia, signal processing and networking Embedded applications are used. The NoCTweak Environment for simulation is shown in Table 1.

Table 1. NoCTweak simulation environment

Platform parameters	Description
Network topology	2D mesh
Type of platform	Real time embedded
Length of packet	Fixed 10 flits
Offered load or traffic (flit injection rate)	0.1 to 1.0 (flits/cycle/node)
Router type	Pipelined, wormhole switching
Selected routing algorithm	XY ordered
Embedded task mapping algorithms	Random, NMAP, BB, GFF, and PSO
Size of buffer	2,4,8,16 (flits)
Length between routers	1000 (um)
Type of pipeline	5
Clock frequency (operating)	1000 MHz
Time taken for warmup	20000 cycles

4.2. Simulation results

This section elaborates the results obtained from simulator after running these mapping algorithms on various embedded application. Mainly two approaches are considered here:

4.2.1. Observations from varying traffic or load on applications

For every application the traffic load (flit injection rate) has been increased from 10 to 100% and latency and energy parameters are observed as indicated in Tables 2 and 3. Then the mean values for both parameters are calculated. On the generated data set, the support vector machine (SVM) model [21] has been trained with test size of 0.8 and then it has been evaluated to predict the best possible mapping algorithm [25] considering minimum latency or minimum energy. This approach will be considered for compute and communication sensitive applications.

Table 2. Mean value latency for embedded applications

Embedded application	Number of tasks	Mapping algorithms and mean value latency (ns)				
		Random	NMAP	GFF	BB	PSO
VOPD	16	2805.98	1202.16	1163.95	1170.51	1658.25
MPEG 4	12	9649.09	8808.99	11988.01	9358.82	9202.61
VCE	25	2460.82	1752.95	1865.46	2540.83	1362.08
MWD	12	4661.17	2055.13	2047.59	4510.16	2873.06
PIP	8	2713.55	2138.70	1695.19	1975.03	2085.56
CAVLC	16	1637.99	283.74	277.95	1874.95	256.14
MMS	25	2586.19	1868.09	2445.07	2432.89	288.94
Wi-Fi Rx	25	5341.96	2907.77	5388.60	2779.51	3735.63
E3S Consumer	12	3411.69	4048.97	2824.56	2838.75	12556.31
E3S Networking	13	1070.17	120.55	123.86	121.71	5.17
E3S AutoIndustrial	24	1167.93	732.90	1082.43	1829.82	1947.15
E3S Telecom	30	1070.17	74.19	1161.98	82.00	1038.28

Table 3. Mean value energy for embedded applications

Embedded application	Number of tasks	Mapping algorithms and mean value energy (pJ/packet/router)				
		Random	NMAP	GFF	BB	PSO
VOPD	16	3.16	2.08	2.40	2.61	3.37
MPEG 4	12	3.86	2.69	3.87	4.35	4.08
VCE	25	2.71	1.67	1.90	2.19	3.36
MWD	12	3.78	2.58	3.09	3.43	4.46
PIP	8	5.45	5.03	4.48	5.15	4.98
CAVLC	16	3.66	2.31	2.47	2.91	3.30
MMS	25	2.47	1.66	2.05	1.87	3.38
Wi-Fi Rx	25	2.91	1.59	1.82	1.96	1.47
E3S Consumer	12	4.20	3.21	3.01	3.50	2.77
E3S Networking	13	2.30	3.01	3.15	2.74	18.11
E3S AutoIndustrial	24	2.48	1.49	1.74	2.03	2.24
E3S Telecom	30	2.30	1.29	1.58	1.54	1.79

4.2.2. Observations from varying buffer size for an application

In this case by varying the buffer size of the router, the effect on performance of both latency and energy has been monitored, four buffer sizes are considered as bsize 2,4,8,16 flits per queue. The flit injection rate (Fir) is kept at 0.9. Four representative applications are considered here are moving picture experts' group 4 (MPEG 4), MWD, embedded system synthesis benchmark suite (E3S) networking and video object planar decoder (VOPD) as shown in Tables 4-7. It has been observed that for MWD application among different algorithms, using BB mapping algorithm by increasing buffer size from 2 flits to 4,8,16 flits reduces latency by 58.84% and 99% respectively whereas the router with 4 flits/buffer consumes almost same energy with 2 flits/buffer router and the routers with 8 flits/buffer and 16 flits/buffer consumes 11.5% and 66 % higher energy respectively.

Table 4. Latency and energy for MPEG 4 application

Bsize	NMAP		BB		GFF	
	Latency	Energy	Latency	Energy	Latency	Energy
2	31747	9.76	20603	9.42	31951	11.37
4	10888	9.336	8480	9.24	10905	11.50
8	5995	10.73	56.48	10.44	5933	13.29
16	5750	16.00	54.76	15.34	5816	19.00

Table 5. Latency and energy for MWD application

Bsize	NMAP		BB		GFF	
	Latency	Energy	Latency	Energy	Latency	Energy
2	36581	12.15	32692	16.03	35715	17.12
4	27850	10.76	29087	16.02	28603	15.73
8	20165	11.69	22208	18.59	21675	16.99
16	19828	16.97	21158	27.69	20663	25.19

Table 6. Latency and energy for E3S networking app

Bsize	NMAP		BB		GFF	
	Latency	Energy	Latency	Energy	Latency	Energy
2	20618	10.83	20603	9.42	20691	10.96
4	8452	10.50	8480	9.24	8628	10.91
8	58.54	11.70	56.48	10.44	61.16	12.32
16	58.54	17.28	56.48	15.34	61.16	18.23

Table 7. Latency and energy for VOPD application

Bsize	NMAP		BB		GFF	
	Latency	Energy	Latency	Energy	Latency	Energy
2	30350	18.32	27037	17.40	28512	15.92
4	13585	17.95	9813	17.68	12315	15.57
8	5914	21.50	5230	21.03	4692	17.91
16	5634	30.95	5230	31.17	4613	26.43

4.3. Analysis and discussion

This section describes each embedded application mentioning whether it computes, communication or memory sensitive and how to opt for particular mapping algorithm to have tradeoff between energy and latency as per the requirements of that application. In all 12 different embedded applications are considered for analysis.

4.3.1. Video object planar decoder

A VOPD is a system or model used in computer vision and ML applications VOPD involves various tasks such as object detection and tracking, perspective estimation and projection, planar object reconstruction. For performing these operations VOPD requires lots of processing power and computational cores as it involves real time processing as well as parallel processing such as GPU acceleration to carry out signal processing tasks. VOPD are typically more sensitive to computational capabilities, even though memory and compute resources are equally vital [26].

Being compute sensitive signal processing application, energy minimization will be the main requirement for the same. The GFF would prove appropriate mapping algorithm considering both latency and energy trade off.

4.3.2. Moving picture experts' group 4

Video and audio compression standard MPEG 4 for compressing digital audio and visual data into smaller file sizes for storage and transmission. This signal processing application can be both compute and memory sensitive depending on the specific use case and the hardware they are running on [27].

The tasks that are sensitive to computing include coding and deciphering. Real-time streaming and rendering whereas memory sensitive tasks involve buffering, frame storage and high-resolution/bitrate content. From the results of the simulator, it has been indicated that NMAP algorithm is best suitable for this MPEG 4 application.

4.3.3. Video conference encoder

This signal processing application is compute as well as communication sensitive. So, minimization of both energy and latency is the requirement of this signal processing application. Encoding video demands a large amount of processing power, particularly when encoding real-time, high-resolution video. Since the encoded video data must be sent to other participants in real-time, video conferences significantly rely on network connection. Because bandwidth, latency, and packet loss have a direct impact on the user experience and video quality, the program is sensitive to these variables. From the results of the simulator, it has been indicated that NMAP algorithm is best suitable for this video conference encoder (VCE) application.

4.3.4. Multi window display

MWD applications' sensitivity to memory, connectivity, and computation might vary according on their specific tasks and level of complexity. Applications with multi-window displays frequently demand a large amount of processing resources, particularly when generating intricate visuals, juggling several open apps at once, or carrying out activities like gaming and 3D modelling. These apps are compute-sensitive because they largely rely on the CPU and GPU to organize and render content across several windows smoothly. Applications that depend on the network for data (such as cloud-based programs and collaborative tools) are particularly vulnerable to communication issues. The GFF would prove appropriate mapping algorithm considering both latency and energy trade off.

4.3.5. Picture in picture

This multimedia application is sensitive to compute, memory as well as communication. Video decoding, rendering, and potentially real-time processing (such as video calls) are common in PIP applications. The application is communication-sensitive if the PIP content is transmitted live or via video conferencing on the internet. More RAM is needed for rendering and buffering high-resolution videos. This may result in crashes or other performance problems in low-memory situations. The GFF would prove appropriate mapping algorithm considering both latency and energy trade off.

4.3.6. Context-aware variable length coder

Context-aware variable length coder (CAVLC) applications for context-aware video coding are typically compute and communication-sensitive. A lot of processing goes into these apps, particularly when it comes to encoding, decoding, and real-time content or user context adaptation. Depending on the amount of network bandwidth available, they might change the bitrate or quality of the video; this calls for effective communication to maximize streaming and lower latency. So PSO will be best possible mapping algorithm.

4.3.7. Multimedia systems

In a real-time video conferencing or live streaming situation, the multimedia systems (MMS) incorporates video/audio processing components such as the H.263 video encoder/decoder and the MP3 audio encoder/decoder. It can be classified as both compute-sensitive and communication sensitive. When processing audio and video feeds in real time is needed, MMS is compute-sensitive. Due to the transmission of audio and video streams, MMS is extremely communication-sensitive, particularly in settings with restricted bandwidth. PSO will be best appropriate mapping algorithm for the same.

4.3.8. Wi-Fi Rx (Wi-Fi receiver)

Applications for Wi-Fi baseband receivers are usually computed and communication sensitive. Real-time data streams are handled by Wi-Fi baseband receivers, necessitating low latency in their signal processing. Because of this, communication sensitivity is essential since hiccups or delays might result in lost data or decreased functionality [15]. The application is compute-sensitive since these tasks require a lot of processing power. BB mapping algorithm would be best mapping for this application as from the simulator results.

4.3.9. Embedded system synthesis benchmark suite

There are 17 different CPUs and 45 distinct tasks in the E3S. Task graphs for the following five embedded application types are included in this benchmark suite: office automation, automotive/industrial, consumer, networking, and telecommunications [28].

4.3.10. E3S consumer benchmark

This application encompasses a variety of consumer-oriented multimedia and image processing tasks operations, which are frequently seen in multimedia processing, picture handling, and other computationally demanding tasks in embedded systems. From the results of the simulator, it has been indicated that GFF algorithm is best suitable for this E3S consumer application.

4.3.11. E3S networking benchmark

Routing algorithms, effective route lookups, and packet flow management are the main focuses of networking activities. These are essential in embedded systems for networking devices like switches, routers, firewalls, or network processors, where effective management of massive volumes of data is required. From the results of the simulator, it has been indicated that PSO algorithm is best suitable for this E3S networking application.

4.3.12. E3S auto industrial benchmark

Tasks related to automotive and industrial settings encompass a broad spectrum of processes, including bit manipulation, matrix arithmetic, control algorithms, and signal processing. The computing requirements of embedded systems in the automotive and industrial industries, where real-time control and data processing are essential, are reflected in these jobs. From the results of the simulator, it has been indicated that NMAP algorithm is best suitable for this E3S networking application.

4.3.13. E3S telecom benchmark

This application part of E3S, assesses embedded systems performance in telecommunications-related tasks, with particular attention on bit manipulation, encoding, and signal processing. The kinds of processes that are essential in contemporary communication systems, like wireless devices, modems, and cell phones, are reflected in these benchmarks. From the results of the simulator BB will be most suitable algorithm for this benchmark.

5. CONCLUSION

A promising on chip communication multicore architecture that enhances SoC performance and scalability is NoC. This paper tackles the trade off between energy and latency for various embedded applications by running a ML model on the dataset generated using the simulator for different mapping algorithms for numerous embedded applications. They can be either compute sensitive, memory sensitive or communication sensitive. The GFF algorithm is most suited for high end video processing applications like VOPD, PIP, and MWD applications. The BB algorithm is most suitable for the applications as Wi-Fi baseband receiver and telecom benchmark which is special case of streaming app that require same processing on each packet and intensive computations per packet. The PSO algorithm is best suited for applications which has more requirement of audio/video encoders and decoders such as MMS, VCE, and networking. The NMAP mapping is best suitable for MPEG 4 and auto industrial benchmark that involve huge computations like compression/decompression, decoding, FIR/IIR filtering and also because of their more memory requirements. This study may be helpful for NoC designers to select appropriate mapping technique for specific kind of embedded application.

FUNDING INFORMATION

The authors declare that no funding was received for the research described in this article.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Shweta Ashtekar	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			
Kushal Tuckley	✓	✓			✓	✓	✓	✓	✓			✓	✓	

C : **C**onceptualization

M : **M**ethodology

So : **S**oftware

Va : **V**alidation

Fo : **F**ormal analysis

I : **I**nvestigation

R : **R**esources

D : **D**ata Curation

O : Writing - **O**riginal Draft

E : Writing - Review & **E**ding

Vi : **V**isualization

Su : **S**upervision

P : **P**roject administration

Fu : **F**unding acquisition

CONFLICT OF INTEREST STATEMENT

The authors declare that there are no conflicts of interest regarding the publication of this paper.

ETHICAL APPROVAL

The research related to human use has been complied with all the relevant national regulations and institutional policies in accordance with the tenets of the Helsinki Declaration and has been approved by the authors' institutional review board or equivalent committee.

DATA AVAILABILITY

All data generated or analyzed during this study are included in this published article.

REFERENCES

- [1] N. Ramakrishnan, V. Sanju, and Farhanaaz, "Analysis of Network on Chip Topologies," in *International Conference on Applied Intelligence and Sustainable Computing, ICAISC 2023*, Jun. 2023, pp. 1–6, doi: 10.1109/ICAISC58445.2023.10200489.
- [2] M. A. Meghabber, A. Aroui, L. Loukil, A. El H. Benyamina, K. Benhaoua, and T. Djeradi, "A flexible network on-chip router for data-flow monitoring," in *2017 5th International Conference on Electrical Engineering - Boumerdes, ICEE-B 2017*, Oct. 2017, pp. 1–6, doi: 10.1109/ICEE-B.2017.8192158.
- [3] S. Ashtekar and K. Tuckley, "Comparison of Latency Minimisation Techniques and Performance Evaluation for Application Mapping in RTNoC (Real Time Network on Chip)," in *Advances in Computational Intelligence and Its Applications*, London: CRC Press, 2024, pp. 320–327, doi: 10.1201/9781003488682-41.
- [4] C. Benchehida, M. K. Benhaoua, H. E. Zahaf, and G. Lipari, "Memory-processor co-scheduling for real-time tasks on network-on-chip manycore architectures," *International Journal of High Performance Systems Architecture*, vol. 11, no. 1, pp. 1–11, 2022, doi: 10.1504/IJHPSA.2022.121877.
- [5] S. Saleem *et al.*, "A Survey on Dynamic Application Mapping Approaches for Real-Time Network-on-Chip-Based Platforms," *IEEE Access*, vol. 11, pp. 122694–122721, 2023, doi: 10.1109/ACCESS.2023.3329233.
- [6] P. K. Sahu and S. Chattopadhyay, "A survey on application mapping strategies for network-on-chip design," *Journal of Systems Architecture*, vol. 59, no. 1, pp. 60–76, Jan. 2013, doi: 10.1016/j.sysarc.2012.10.004.
- [7] S. Saleem, F. Hussain, and N. K. Baloch, "IWO-IGA—A Hybrid Whale Optimization Algorithm Featuring Improved Genetic Characteristics for Mapping Real-Time Applications onto 2D Network on Chip," *Algorithms*, vol. 17, no. 3, Mar. 2024, doi: 10.3390/a17030115.
- [8] W. Fornaciari, D. Sciuto, C. Silvano, and V. Zaccaria, "A sensitivity-based design space exploration methodology for embedded systems," *Design Automation for Embedded Systems*, vol. 7, no. 1–2, pp. 7–33, Sep. 2002, doi: 10.1023/A:1019791213967.
- [9] A. Cakin, S. Dilek, and S. Tosun, "Energy-aware application mapping methods for mesh-based hybrid wireless network-on-chips," *Journal of Supercomputing*, vol. 80, no. 11, pp. 15582–15612, Jul. 2024, doi: 10.1007/s11227-024-06062-4.
- [10] D. Bertozzi *et al.*, "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 2, pp. 113–129, Feb. 2005, doi: 10.1109/TPDS.2005.22.
- [11] K. Balamurugan, S. Umamaheswaran, T. Mamo, S. Nagarajan, and L. R. Namamula, "Roadmap for machine learning based network-on-chip (M/L NoC) technology and its analysis for researchers," *Journal of Physics Communications*, vol. 6, no. 2, Feb. 2022, doi: 10.1088/2399-6528/ac4dd5.
- [12] M. Obaidullah, "Application mapping and NoC configuration using hybrid particle swarm optimization," Toronto Metropolitan University, 2021, doi: 10.32920/ryerson.14656014.v1.
- [13] X.-S. Yang, *Nature-inspired metaheuristic algorithms*, 2nd Edition, Luniver Press, 2010.
- [14] M. F. Reza and P. Ampadu, "Energy-efficient and high-performance NoC architecture and mapping solution for deep neural networks," in *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip*, Oct. 2019, pp. 1–8, doi: 10.1145/3313231.3352377.
- [15] R. Yan, Y. Zhou, A. Cai, C. Li, Y. Yan, and M. Yin, "Contention-aware mapping and scheduling optimization for NoC-based MPSoCs," in *Proceedings International Conference on Automated Planning and Scheduling, ICAPS*, vol. 30, pp. 305–313, Jun. 2020, doi: 10.1609/icaps.v30i1.6675.
- [16] A. Bender, "MILP based task mapping for heterogeneous multiprocessor systems," in *European Design Automation Conference - Proceedings*, 1996, pp. 190–197, doi: 10.1109/eurdac.1996.558204.
- [17] S. Khan, S. Anjum, U. A. Gulzari, M. K. Afzal, T. Umer, and F. Ishmanov, "An Efficient Algorithm for Mapping Real Time Embedded Applications on NoC Architecture," *IEEE Access*, vol. 6, pp. 16324–16335, 2018, doi: 10.1109/ACCESS.2018.2811716.
- [18] M. F. Reza and Z. McCloud, "Heuristics-Enabled High-Performance Application Mapping in Network-on-Chip based Multicore Systems," in *2023 IEEE International Conference on Omni-Layer Intelligent Systems, COINS 2023*, Jul. 2023, pp. 1–6, doi: 10.1109/COINS57856.2023.10189228.
- [19] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications*, vol. 80, no. 5, pp. 8091–8126, Feb. 2021, doi: 10.1007/s11042-020-10139-6.
- [20] S. L. Tilahun and H. C. Ong, "Modified firefly algorithm," *Journal of Applied Mathematics*, no. 1, Jan. 2012, doi: 10.1155/2012/467631.
- [21] M. J. Mohiz, N. K. Baloch, F. Hussain, S. Saleem, Y. B. Zikria, and H. Yu, "Application mapping Using cuckoo search optimization with lévy flight for NoC-based system," *IEEE Access*, vol. 9, pp. 141778–141789, 2021, doi: 10.1109/ACCESS.2021.3120079.
- [22] M. Bougherara and R. Amara, "Application Mapping Onto Network on Chip Using Genetic Algorithm and Ant Colony Optimisation," in *2023 International Conference on Computer and Applications (ICCA)*, Nov. 2023, pp. 1–6, doi: 10.1109/ICCA59364.2023.10401486.
- [23] W. Amin, F. Hussain, S. Anjum, S. Saleem, W. Ahmad, and M. Hussain, "HyDra: Hybrid Task Mapping Application Framework for NOC-Based MPSoCs," *IEEE Access*, vol. 11, pp. 52309–52326, 2023, doi: 10.1109/ACCESS.2023.3279501.
- [24] A. Tran and B. Baas, "Noctweak: A highly parameterizable simulator for early exploration of performance and energy of networks on-chip," Technical Report, VLSI Computation Lab, ECE Department, UC Davis, 2012.
- [25] A. Kumar and B. Talawar, "Machine Learning Based Framework to Predict Performance Evaluation of On-Chip Networks," in *2018 11th International Conference on Contemporary Computing, IC3 2018*, Aug. 2018, pp. 1–6, doi: 10.1109/IC3.2018.8530505.
- [26] H.-N. Nguyen, V.-D. Ngo, and H.-W. Choi, "Realization of Video Object Plane Decoder on On-Chip Network Architecture," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3820 LNCS, 2005, pp. 256–264, doi: 10.1007/11599555_26.
- [27] E. B. van der Tol and E. G. Jaspers, "Mapping of MPEG-4 decoding on a flexible architecture platform," in *Media Processors 2002*, 2001, vol. 4674, p. 1, doi: 10.1117/12.451067.
- [28] R. Dick, "Embedded systems synthesis benchmark suites (e3s)," *National Institute of Informatics*, 2008, <https://ziyang.eecs.umich.edu/~dickrp/e3s/>, (Accessed Aug. 20, 2026).

BIOGRAPHIES OF AUTHORS

Dr. Shweta Ashtekar    has received her Masters in Electronics degree from Mumbai University, India and her Ph.D. in Electronics and Telecommunication from Mumbai university. Currently she is working as Assistant Professor at Ramrao Adik Institute of Technology, Mumbai. With over 23 years of teaching experience in engineering education, she has made significant contributions to academics, research, and curriculum development. She has published more than 45 research papers in reputed international journals and conferences. She holds four patents to her credit, including one granted patent and three published patents, and has secured more than 10 copyrights. She has also authored two books in her areas of expertise. Her research of interests are embedded systems, IoT and cloud computing, computer networks, and computer architecture. She can be contacted at email: shweta.ashtekar@rait.ac.in.



Dr. Kushal Tuckley    is currently holding the post of Chairman and Director R&D in agv Systems Pvt.Ltd. He had professional education in IIT Bombay, with B.Tech (EE, 1985), M.Tech. (Communications Engineering, 1989) and Ph.D. (2009) Earlier. He has worked as a scientist for more than 24 years in society for applied microwave electronics engineering and research (SAMEER), Mumbai, a premier research institute in the field of RF and microwave system engineering. He had a stint at Astra Microwave Products Ltd. Hyderabad, the largest Indian private company for microwave products. He has a large experience in DSP, RF, and microwave engineering. He has contributed in design and commissioning of many prestigious systems of India. He can be contacted at email: kushal_tuckley@rediffmail.com.