Enhancing Reliability and Energy Efficiency in Many-Core Processors Through Fault-Tolerant Network-on-Chip

B. Naresh Kumar Reddy, *Senior Member, IEEE*, Md. Zia Ur Rahman, *Senior Member, IEEE*, and Aime Lay-Ekuakille, *Senior Member, IEEE*

***Abstract*—This article presents a proposal for fault-tolerant task mapping on many-core processors to enhance system performance and reduce communication energy. The proposed algorithm maps tasks onto a 2-D mesh network-on-chip (NoC) and a modified NoC (MNoC) platform. The focus of this article is primarily on addressing permanent faults. In the scenario of a permanent fault within the mapped core, the algorithm also proposes a spare core placement strategy. This involves allocating the spare core based on considerations related to communication energy. The proposed task mapping algorithm underwent evaluation using various benchmarks, including multimedia and synthetic benchmarks. The results were then compared to those obtained from a 2-D mesh NoC and three related algorithms, all under the same task graph and NoC size. The simulation results showed that the proposed mapping algorithm on the modified NoC platform leads to improved performance and communication energy reductions when compared to the 2-D mesh NoC and the other three algorithms. To validate the proposed fault-tolerant task mapping algorithm on the modified NoC platform, A Field Programmable Gate Array (FPGA) was used to measure performance metrics such as application runtime, area, and on-chip power consumption in both faulty and non-faulty conditions. The hardware results indicated significant improvements when comparing the proposed FTTM on MNoC and 2-D NoC with existing approaches.**

***Index Terms*—Network-on-chip, core, task mapping, communication energy, performance, FPGA board.**

I. Introduction

SYSTEM-ON-CHIP is a design and implementation model for circuits that can support multiple devices on a single chip. As VLSI technology advances, there has been an increase in the use of network performance and connectivity, leading to the creation of multiprocessor system-on-chips (MPSoCs) and Network on Chips (NoCs) [1]. The core mechanisms used to interact are (i) a traditional bus-based system used in SoCs, (ii) point-to-point connections used in MPSoCs, and (iii) communication through routers implemented in NoCs [2].

NoCs address the architecture's efficiency and flexibility requirements [3]. Information is passed across the network fabric using packets, which are composed mainly of network interfaces and interconnections between processing elements and routers [4]. The configuration and architecture of NoCs are comprised of three major components: 1) partitioning of an application, 2) scheduling tasks, where application partitioning and task scheduling are closely linked to task deadlines and processing time, and 3) application mapping. Application mapping is the key element for providing an efficient network [5].

Initially, tasks for a given system are assigned to each appropriate core, and then, a mapping technique is constructed based on criteria to coordinate the alignment of cores into a network. A well-categorized mapping methodology must be implemented in NoC, taking into account the primary gaps in previous networks [6]. Application mapping is an NP-hard challenge because the search space increases as the network size grows. It is mainly divided into two types: 1) Static mapping, where tasks are assigned before the application is executed, and 2) Dynamic mapping, which allocates tasks to the application during runtime [7]. Providing a fault-tolerant mapping mechanism is a great challenge nowadays. Tasks of an application should continue even after a core fails, with the help of a spare core replacement. This technique creates a fault-tolerant mapping [8].

Motivated by these facts, the authors propose a task mapping algorithm with consideration of faults and a modified NoC platform. The main contributions of this article are as follows:

1. The authors propose a fault-tolerant task mapping algorithm (FTTM), which can map and schedule both offline and runtime tasks on a 2-D mesh NoC and modified NoC platform, initially optimizing the NoC mapping and scheduling region.
2. The authors also propose a spare core technology where if a fault occurs at a core in a 2-D mesh NoC and modified NoC (MNoC) platform, the spare core can handle the communications associated with the failed core.

The objective of the proposed FTTM algorithm is to enhance performance and minimize communication energy. Given the fact that mapping and spare core positions are

Manuscript received 5 October 2023; revised 6 February 2024 and 28 March 2024; accepted 25 April 2024. Date of publication 30 April 2024; date of current version 16 October 2024. The associate editor coordinating the review of this article and approving it for publication was M. Tornatore. *(Corresponding author: B. Naresh Kumar Reddy.)*

B. Naresh Kumar Reddy is with the Department of Electronics and Communication Engineering, National Institute of Technology Tiruchirappalli, Tiruchirappalli 620015, India (e-mail: naresh.nitg@gmail.com).

Md. Zia Ur Rahman is with the Department of Electronics and Communication Engineering, Koneru Lakshmaiah Education Foundation, Guntur 522502, India (e-mail: mdzr55@gmail.com).

Aime Lay-Ekuakille is with the Department of Innovation Engineering, University of Salento, 73100 Lecce, Italy.

Digital Object Identifier 10.1109/TNSM.2024.3394886

NP-complete, a modified NoC platform is a good choice for this problem. The proposed FTTM algorithm on the modified NoC platform is tested using various benchmarks, including multimedia and synthetic benchmarks, and compared with a 2-D mesh NoC and other related algorithms. The proposed fault-tolerant task mapping algorithm has been simulated using the Vivado tool and verified using the Zynq UltraScale MPSoC ZCU104 Evaluation Kit. The simulation results showed that the FTTM algorithm on the modified NoC platform demonstrated improvement compared to the 2-D mesh NoC and other related algorithms in the literature.

The article is organized as follows. In Section II, the authors briefly review related work on task mapping and fault tolerance issues in NoC. The motivation of the paper is clarified in Section III. Problem definition and proposed strategy are explained in Sections IV and V, respectively. In Section VI, the authors present the experimental results. Hardware implementation is discussed in Section VII. Finally, in Section VIII, the authors conclude the paper.

II. Related Work

Recently, much attention has been paid to fault-tolerant mapping techniques for NoC. Many of the published works in the literature consider performance metrics and communication energy when it comes to fault tolerance core mapping methodologies. Zixu Wu et al. [9] proposed a Fast Two-Step Topology Reconfiguration (FTTR) to address the challenges faced with reconfigurable mapping techniques in fault-tolerant NoC architectures. This provides an efficient solution by balancing the interface mostly to the upper application, with a unified virtual topology to the physical topology. Bizot et al. [10] proposed a methodology for performing parallel applications for upcoming many-core processors that are inefficient. This implementation dynamically maps the tasks of an application either while creating the tasks or even after a fault occurs. Wang et al. [11] introduced a new fault-tolerant graph that addresses the reliability issues of the network through the ideology of k-node. This employs the divide-and-conquer technique to implement a fault-tolerant ad-hoc network. The number of edges is reduced with the help of an exhaustive search verification algorithm. Bonney et al. [12] developed a methodology for mapping fault-tolerant applications to many-core architectures. It uses a multi-objective evolutionary algorithm to develop various feasible mappings for the tasks in the application. The fault-tolerant characteristics are prioritized to facilitate task mappings that allow for fast and low-cost recovery, while performance metrics provide mappings with less traffic.

Zhang et al. [13] proposed a two-stage task mapping technique called Variation. In the first stage, various mapping solutions are generated through static mapping techniques, and the most effective mapping route is selected during runtime execution. Chatterjee et al. [14] introduced a dynamic technique for allocating tasks in a fault-tolerant manner, taking into account deadlines, timing characteristics, and performance metrics. Khalili and Zarandi [15] developed an algorithm for mapping applications, consisting of two steps. First, the application is mapped to free and non-faulty cores using heuristic mapping, and then a spare core is replaced whenever there are failed cores in the application. Khalili and Zarandi [16] proposed a dynamic allocation technique for spare cores in a network, where a spare core is allocated near the nearest critical core in the event of a failure. Bhanu et al. [17] utilized the Discrete Particle Swarm Optimization (DPSO) technique to provide an efficient placement for spare cores in a torus topology network. Chatterjee et al. [18] proposed a fault-tolerant dynamic mapping and scheduling method for single and multiple core failures, which uses a unified mapping and scheduling algorithm for real-time applications. A predictive model was used to control the failure-prone cores in the processor, and a fault-tolerant distribution with task redundancy was performed. Bhanu et al. [19] utilized Integer Linear Programming (ILP) and Particle Swarm Optimization (PSO) techniques to provide the best placement for spare cores in a network, considering permanent faults. This research was carried out on both mesh and torus topologies and with various application benchmarks. Mohiz et al. [20] introduce an efficient approach for NoC application mapping, combining a greedy algorithm for initial task placement with the metaheuristic Cuckoo Search via Levy flight for further optimization. This work introduces a novel Cube-Tree-Hybrid (CTH) topology for runtime application mapping in NoC-based multiprocessor System-on-Chips (MPSoCs), emphasizing traffic load balancing. The contributions include a scalable CTH design, a runtime mapping strategy optimizing traffic load distribution, and comprehensive experimentation across various benchmarks and network topologies [21].

All these works focus on mapping algorithms and fault-tolerance techniques, but in this paper, we propose a fault-tolerant task mapping and a modified NoC platform that not only optimizes the mapping region but also improves performance and reduces communication energy.

III. Motivation

When mapping a given application onto a NoC architecture, we must choose the mapping region and schedule the application tasks on the available cores in that region. The selection criteria are typically based on communication energy, performance, and fault tolerance.

The Mesh NoC architecture (m x n) consists of a router, core, and network interface, which is shown in Fig. 1(a). Each router has five input and output (I/O) ports, where four I/O ports are connected to the neighboring routers and one I/O port is connected to the local core. Every core in the NoC is connected to a router through a network interface. The cores can be homogeneous, such as CPUs, or heterogeneous, such as audio-video cores, wireless transmitters, and receivers. The network interface serves as a communication medium between the core and router and is used for packetizing and depacketizing data. In this study, a modified mesh NoC architecture (m x n) is proposed, where every core is connected to two routers, or each router has six I/O ports (four connected to the neighboring routers and two connected to the

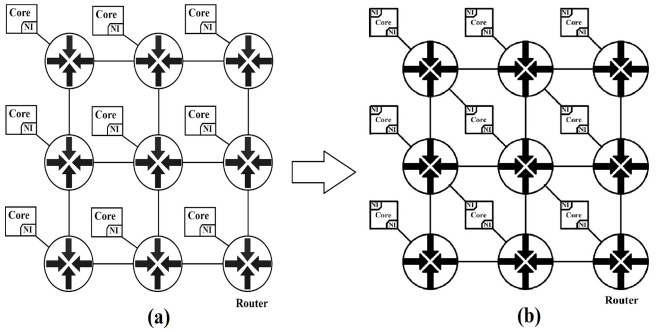

Fig. 1. (a) 2-D mesh NoC architecture, (b) Modified 2-D mesh NoC (MNoC) architecture.

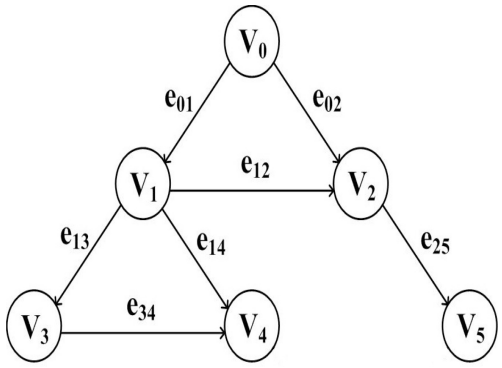

Fig. 2. An example of Task Model.

local core), as shown in Fig. 1(b). The modified mesh NoC architecture improves the efficiency of data transmission and point-to-point communication through the routers, reducing communication energy and enhancing performance. When mapping and scheduling application cores, the modified mesh NoC architecture is preferred. In a traditional NoC, the focus is on task running time and communication energy, rather than the number of links. In the modified NoC, both run-time and communication energy are significantly lower compared to a 2-D NoC, and the number of links is slightly higher compared to the 2-D NoC. This architecture is designed to provide a high level of redundancy by connecting each system in the network to every other system on the network.

This paper proposes a fault-tolerant task mapping algorithm (FTTM) designed for a modified mesh Network-on-Chip (NoC) architecture. The FTTM efficiently handles task mapping and scheduling for both offline and run-time scenarios. Initially, the mapping region undergoes optimization using optimal distance (OD), followed by task mapping and scheduling on both the traditional 2-D mesh NoC and the modified mesh NoC architecture using FTTM. In the event of faults occurring at a core, error correction codes (specifically Hamming code) are employed to verify and correct the code. If the fault persists even after applying the Hamming code, it is considered a permanent fault. In cases of permanent core failure on the 2-D NoC or MNoC, an alternative core (spare core) is chosen for the tasks, resulting in only a minor degradation in energy consumption and performance. All available cores in the 2-D NoC and MNoC are designated as spare cores in this study, serving as backups in the event of core failure during task processing. The primary objective is to achieve superior performance and minimize communication energy overhead compared to previous related work.

IV. Preliminaries and Problem Formulation

In this section, offline and runtime tasks are mapped and scheduled on a 2-D Network-on-Chip (NoC) using a modified mesh architecture. A communication energy model is also presented.

A. Task Model

Application tasks running on mesh NoC architecture are addressed through a task graph. A Task Graph TG = (V, E), where 'V' represents Vertex, which contains tasks ($t_i \in$ T). Each directed edge 'e_{ij}' $\in$ E represents the communication rate between two vertices (V_i and V_j) [22]. The pictorial representation of the task graph is outlined in Fig. 2.

B. Mapping and Scheduling Model

The task graph TG = (V, E) and the modified mesh NoC platform MN = (C, D) are inputs for the mapping and scheduling model. C represents the cores and D represents the distances between two cores. Tasks from each vertex are allocated to available free cores in the modified mesh NoC architecture based on the proposed algorithm, taking into consideration communication energy. In NoC task mapping, tasks can be executed sequentially on a single core or simultaneously on multiple cores. The NoC platform includes various types of cores: Free cores, which are accessible and responsible for executing tasks in the application; Busy cores, which are currently running tasks in the application; Failed cores, which have failed due to transient or permanent faults; Spare cores, which recover from failures in free or busy cores and temporarily work until the end of execution; and Manager cores, which monitor the status of all NoC cores and perform task migration in the event of a fault. It is crucial to ensure that the availability of free cores in the modified NoC architecture is always greater than the total number of vertices in the selected task graph [23].

$$V \leq C_f \tag{1}$$

where V is represented as the number of vertices in the task graph and C_f is represented as the number of free cores in the modified NoC architecture.

C. Communication Energy Model

Communication energy between two mapped vertices depends on the distance between them as well as the communication rate [24].

$$CE(V_i, V_j) = e_{ij}|I_{V_i V_j}| \tag{2}$$

e_{ij} is communication rate between two vertices (V_i, V_j).

$I_{V_i V_j}$ is communication distance between two vertices (V_i, V_j).

V_i parameters (x_1, y_1) and V_j parameters (x_2, y_2).

$$|I_{V_i V_j}| = |x_1 - x_2| + |y_1 - y_2| \tag{3}$$

Problem 1: Given Task Graph (V, E) and mesh NoC platform MN= (C, D); Find a mapping function $\Omega : V \rightarrow C$ with $\forall V_i \in V$ and calculate the total communication energy.

such that:

$\forall V_i \in V, \forall C_i \in C.$

$\Omega(V_i) \in C,$

$V_i \neq V_j \implies \Omega(V_i) \neq \Omega(V_j)$

$\forall C_{ij} \in D.$

Let $\forall V_i \in V$ be mapped to some $t_{xy} \in C.$

Then, $t_{xy} = \Omega(V_i) \in C.$

Communication energy is calculated using Eq. (2).

$CE(V_i, V_j) = (e_{ij}) \times C_{ij}$

Total communication energy

$$T_{CE} = \sum_{\forall(i,j)} (e_{ij}) \times C_{ij} \tag{4}$$

D. Optimal Distance (OD)

The Optimal Distance (OD) represents the shortest path between any two cores in the network, as determined by the Node Average Distance [25].

Calculating the shortest path in an N-dimensional mesh involves measuring the distance between two cores. In a linear array, this distance encompasses the positions of cores on the left and right sides relative to all other cores.

$$\begin{aligned}
\Delta_{N-path} &= \frac{1}{N^2} \sum_{0 \le i \le N-1} \left[\sum_{0 \le j \le i} (i-j) + \sum_{i \le j \le N-1} (j-i) \right] \\
&= \frac{1}{N^2} \sum_{0 \le i \le N-1} [i(i+1) - i(i+1)/2 \\
&+ (N-i)(N-1+i)/2 - i(N-i)] \\
&= \frac{1}{N^2} \sum_{0 \le i \le N-1} \left[i^2 - (N-1)i + N(N-1)/2 \right] \\
&= \frac{1}{N^2} \left[N(N-1)(2N-1)/6 - N(N-1)^2/2 + N^2(N-1)/2 \right] \\
&= \frac{1}{3}[N - 1/N]
\end{aligned}$$

The average core distance in a p-dimensional mesh is as follows:

$$\Delta_{p-mesh} = \frac{1}{3} \left[\sum_{0 \le i \le p} (n_i - 1/n_i) \right]$$

The preferred cores average distance on a NoC platform, with size X x Y. The Optimal Distance (OD) is computed as given in (4).

$$\begin{aligned}
\text{OD} &= \frac{1}{3}[(X - 1/X) + (Y - 1/Y)] \\
\text{OD} &= \frac{X+Y}{3} \left(1 - \frac{1}{XY} \right)
\end{aligned} \tag{5}$$

E. Fault Model

A typical Network-on-Chip (NoC) architecture comprises a core, router, and network interface. While faults can arise at any component and at any time, this research specifically targets faults occurring at the core. The identified faults fall into three categories: Transient faults, Intermittent faults, and Permanent faults. Detecting and correcting Transient and Intermittent faults are relatively straightforward using simulation code, particularly employing Hamming code [26]. However, Permanent faults at the core lead to complete task execution failure. To detect faults, data and control NoCs are utilized, with a higher likelihood of occurrence at heavily communicating vertices. This research places particular emphasis on addressing the challenges posed by Permanent faults at the core, recognizing their critical impact on task execution.

F. Problem Definition

The goal of the fault-tolerant task mapping algorithm is threefold:

1. To optimize the mapping region for mapping and scheduling tasks on a 2-D mesh NoC and modified NoC (MNoC) architecture.
2. To determine the mapping methodology in the event of permanent faults at the core.
3. To minimize energy consumption and enhance performance.

To achieve these goals, the first step is to determine the number of vertices in the task graph. Then, the mapping region is optimized using the Optimal Distance (OD) and vertices are mapped. In the presence of core faults, Hamming codes are utilized for error detection and correction. If the fault persists despite Hamming code application, signifying permanence, the algorithm turns to spare core technology for resolution.

V. Fault Tolerant Task Mapping Algorithm (FTTM)

To maintain consistency, the processor must have a mechanism to bypass faulty elements. The proposed fault-tolerant task mapping (FTTM) technique includes:

A. Mapping tasks onto free and non-faulty cores.

B. Spare core placement.

A. Task Mapping

In the proposed task mapping algorithm shown in Algorithm 1, the inputs are the task graph model and the modified NoC platform. The algorithm starts by initializing the task graph as mapped and unmapped vertices. The optimal mapping region is selected based on the unmapped vertices using the Optimal Distance (OD). The initial vertex with the maximum communication with other vertices is chosen (if more than one vertex has the same number of communications, the vertex with the highest communication rate is selected). The core 'C_{xy}' centered in the optimal mapping region (which communicates with the maximum number of free cores) is selected and the status of the vertices is updated. The next vertex with the maximum communication with other vertices in descending order is then selected, along with a core that has a direct connection to the center mapping core. This process is repeated until all vertices are mapped to their respective processing cores, with minimum communication energy. In case of a permanent fault at any core, the tasks of the faulty

Algorithm 1 Task Mapping Onto NoC Algorithm

Input : MN(C, D) - Modified NoC platform status, TG(V, E) - Task Graph Model
Output: TM - Task mapping table, MN(C, D) - Modified NoC platform status

MapTasksOntoNoC(MN(C, D), TG(V, E)): Initialize a task mapping table TM[V] with default values indicating no mapping;
foreach *Optimal mapping region* $\in$ R **do**
  compute $OD = \frac{X+Y}{3}(1 - \frac{1}{XY})$;
end
Select mapping region corresponds to minimum OD;
for *each task v in TG* **do**
  Find the set of candidate cores for mapping the task based on its communication requirements and available resources;
  Select the best core from the candidate set based on some criteria (e.g., minimize communication distance, balance load, etc.) using a cost function;
  if *a suitable core is found* **then**
    Map the task v onto the selected core;
    Update the task mapping table TM[v] with the selected core;
  end
end
return TM and MN(C, D) mapping platform status;

FindCandidatecores(task, MN(C, D)) Initialize an empty set Candidatecores;
for *each core t in MN(C, D)* **do**
  if *core t satisfies the resource requirements of the task and has sufficient bandwidth for communication* **then**
    Add core t to Candidatecores;
  end
end
return Candidatecores;

SelectBestcore(Candidatecores) Implement a strategy to select the best core from the set of Candidatecores based on a cost function;
return *Find the mapped vertices.*

Algorithm 2 Spare Core Placement Algorithm

Input: MN(C, D) - Modified NoC mapping platform status
Input: Task Graph Model TG(V, E)
Output: MN(C, D) - Modified NoC updated platform status
Step 1: Calculate Spare Core Communication Energy;
Let P_{spare} denote the spare core communication energy;
Initialize energy variable to 0;
Step 2: Calculate Communication Energy for Each Communication Link;
while $P_{\text{spare}} \neq 0$ **do**
  Calculate communication energy $CE(V_i, V_j)$ for each communication link (V_i, V_j) in the task graph;
  $\text{CE}(V_i, V_j) = e_{ij} * |I_{V(i)V(j)}|$;
  $T_{CE} = \Sigma_{\forall(i,j)} (e_{ij} * |I_{V(i)V(j)}|)$;
  energy = energy + T_{CE} ;
  Calculate total communication energy T_{CE} by summing up $CE(V_i, V_j)$ over all communication links;
  Add T_{CE} to the total energy;
end
Step 3: Check Minimum Energy Condition;
Let *minenergy* denote the minimum allowable energy;
if *total energy* $\leq$ *minenergy* **then**
  Update *minenergy* to the current total energy;
end
Step 4: Select Spare Core with Minimum Computational Energy;
Select an available core C_{xy} from the NoC platform that minimizes the computational energy;
Step 5: Update Platform Status;
Update the status of core C_{xy} in MN(C, D) to indicate it as a spare core;
Step 6: Output Updated Platform Status;
return *Updated MN(C, D) mapping platform status.*

core are migrated to an alternative free core (spare core). After the task mapping is completed, every free core acts as a spare core, which is explained in Algorithm 2.

B. Spare Core Placement

Spare core placement is explained in Algorithm 2, which involves selecting free cores that are located near the faulty cores, based on the minimum communication energy. Initially, the communication energy is set to zero and the spare core is placed near the faulty core. The subsequent energy is then added to the initial energy. Finally, the position of the spare core is updated with the free core that has the lowest energy among all the available free cores. This process is repeated for all the faulty cores presented [27].

C. Motivational Example

Fig. 3, shows the example of a task mapping strategy, a simple task model, and a modified NoC platform 5 x 5 shown in Fig. 3(a) and Fig. 3(b). The optimal mapping region has selected 3 x 3 regions according to the task model vertices rather than the 5 x 5 region (highlighted in red color) shown in Fig. 3(c). The initial vertex is selected as V_1, which has maximum communication with other vertices, it is mapped to C_{12} has centered in the optimal mapping region, which communicates a maximum number of free cores in NoC shown in Fig. 3(d). Select other vertices V_2, which has the next maximum communication with the other vertices in the task model, and choose the core (C_{02}), which is a direct connection with C_{12} shown in Fig. 3(e). Other vertices mapping order $(V_4, V_3, V_0, V_5$ and $V_6)$ based on next maximum communication with other vertices and cores scheduling order $(C_{22}, C_{11}, C_{01}, C_{13}$ and $C_{23})$ are shown in Fig. 3(f)–(j).

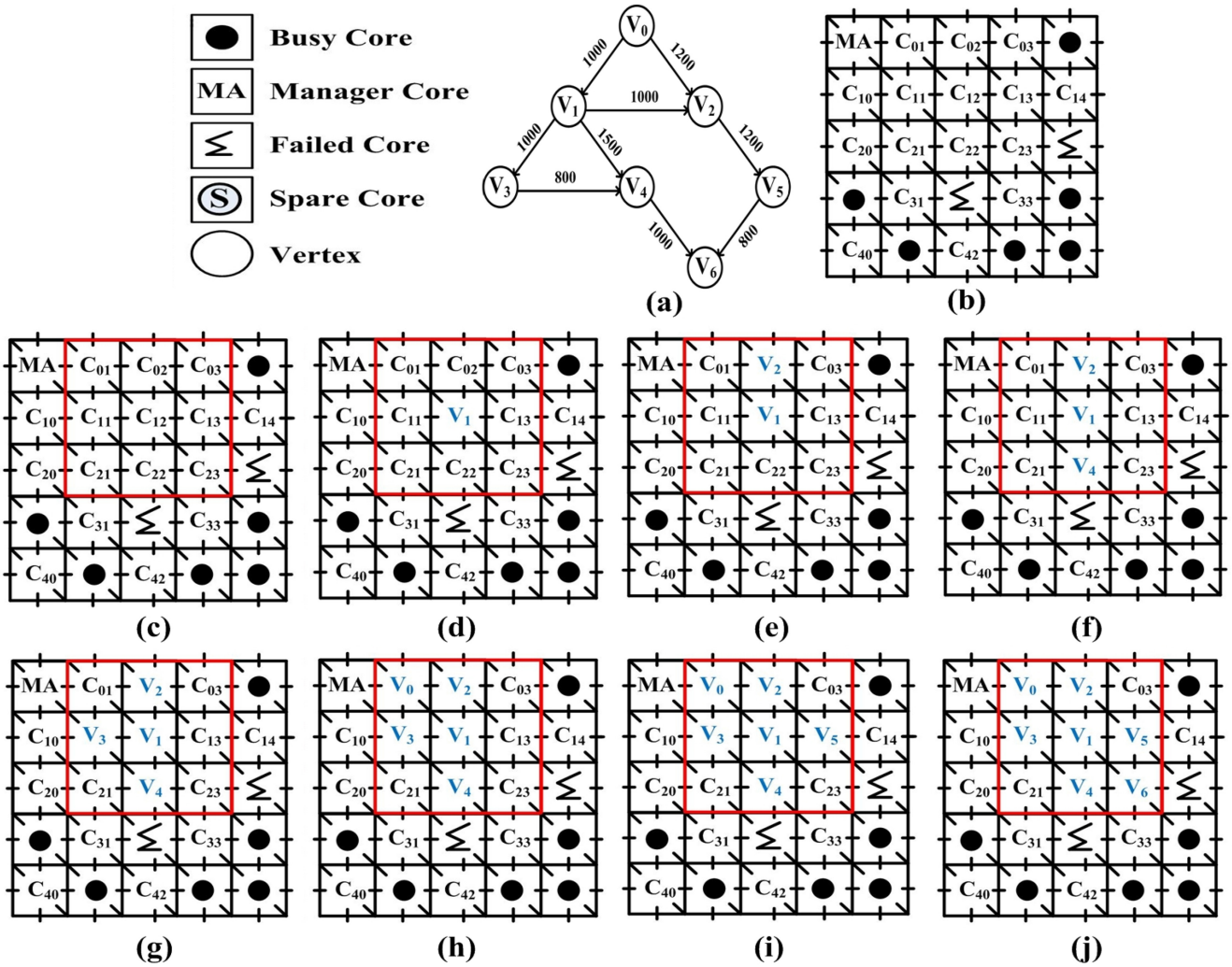

Fig. 3. Mapping: (a) An example of Task Model, (b) 5 x 5 modified mesh NoC, (c) Optimal Mapping Region, (d-j) Proposed FTTM algorithm process.

VI. Experimental Results

The efficiency of the FTTM algorithm is being evaluated using a cycle-accurate SystemC-based Noxim simulator [28]. The packet length is set to 16 flits, the simulation time is 20,000 cycles with the first 10,000 cycles being warm-up time. The average latency under various packet injection rates is used as the performance index. The proposed algorithm and comparison algorithms use modified XY routing, which has lower path diversity due to its deterministic properties. The network performance is assessed using a transpose distribution in a 5 by 5 mesh network. The average latency under various packet injection rates is utilized as the performance index of the simulations.

In this research mainly concentrated on the core, the simulator inputs are task graph, mesh NoC platform, routing algorithm, strategies for mapping and scheduling, and fault tolerance strategy. The task graph model has to map according to the algorithm. Next, we permit the user to choose the vertices on the NoC platform. Faults are injected erratically in the NoC platform and depend on the maximum number of faults that the system can withstand without failure. In Fig. 4, faults are considered as a priority of maximum communicating vertices, The Noxim simulator is activated according to the event of an occasion. An occasion is recognized as the appearance of any application or departure of a finished application from the objective platform or failure of a functional core. At whatever point one such occasion happens, the allocation algorithm is executed. The manager core controls the task mapping and spare core allocation. In our case, the essential goal comprises task mapping concerning minimizing the communication energy consumption of all running task graph applications while imparting fault tolerance to the multicore platform. Various task models with many vertices ranging from 5 to 25, were generated by TGFF [29]. The FTTM algorithm was evaluated by simulating task models on modified NoC platforms with sizes of 5 × 5, 6 × 6, 8 × 8, and 10 × 10. Over 1,000 test cases were examined in these simulations, which comprised randomly generated combinations of applications from different test categories with varying levels of scheduling difficulty. In each scenario, random core faults were introduced and distributed across the NoC platform. The test cases were analyzed, taking into account both single and multiple core failures as well as the order in which they occurred.

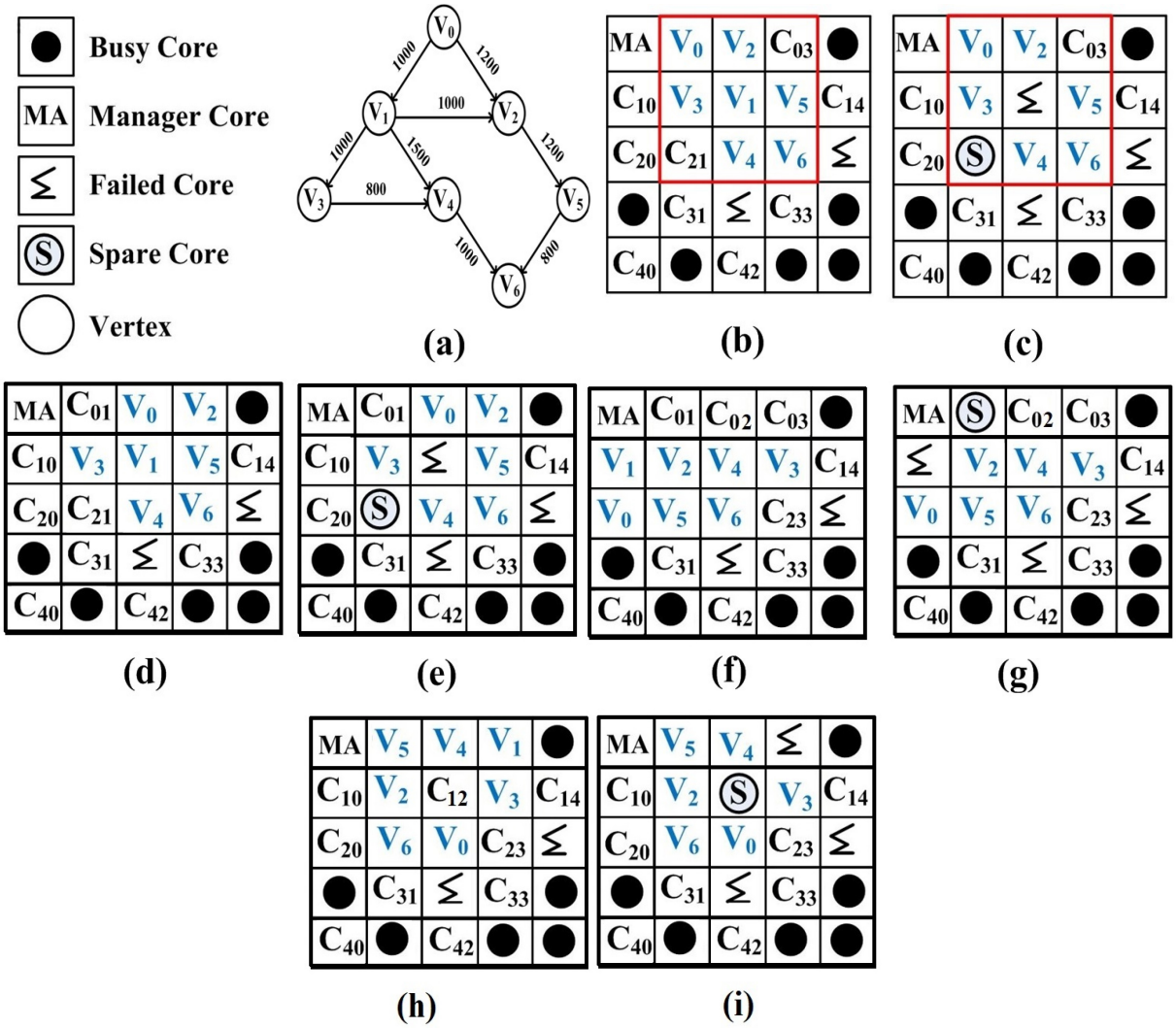

Fig. 4. Comparison of mapping and spare core placement: (a) An example of Task Model, (b) FTTM mapping, (c) FTTM spare core placement, (d) FASA mapping [16], (e) FASA spare core placement [16], (f) FTDTMS mapping [18], (g) FTDTMS spare core placement [18], (h) FTNoC mapping [19], (i) FTNoC spare core placement [19].

FASA [16], FTDTMS [18], and FTNoC [19] methods are commonly used for spare core allocation, we tested all three techniques using the same task graph and the same NoC platform to examine the merits and demerits of FTTM, FTTM mapping algorithm shown in Fig. 4(b). In this example, the high probability of failure cores was given in the order as V_1, V_2, V_4, V_3, V_0, V_5 and V_6. The spare core position is considered near the faulty cores based on the minimum communication energy, here we considered a single fault to occur on the NoC platform, FTTM spare core placement is shown in Fig. 4(c). In the FASA method, the vertices with the most extreme communication energy with neighbors and the cores are free, and communicating with free neighboring cores was chosen, which can be shown in Fig 4(d). Here C_{21} position is the most favorable as a spare core shown in Fig. 4(e). In the FTDTMS technique, vertices are mapped in a particular sequence and allocated cores in the free position shown in Fig. 4(f), the failed core is migrating to the next most communicating parent as shown in Fig. 4(g). The FTNoC method concentrated on only core failures and spare core placement based on Integer Linear Programming (ILP). In this example, FTNoC core mapping and spare core placement are shown in Fig. 4(h) and 4(i). To estimate the performance and communication energy of the FASA [16], FTDTMS [18], FTNoC [19], and proposed algorithm mappings on 2-D NoC and modified NoC, used the Noxim simulator. Mapping time and latency of FASA [16], FTDTMS [18], FTNoC [19], and proposed algorithm mappings on 2-D NoC and modified NoC are tabulated in Table I. Communication energy of FASA [16], FTDTMS [18], FTNoC [19], and proposed algorithm mappings on 2-D NoC and modified NoC without fault and 1 faulty condition is tabulated based on eq (3) in Table II.

The increase in the number of faulty cores results in higher communication energy, as illustrated in Fig. 5. Comparative analysis from Tables I and II reveals that the FTTM algorithm (MNoC) exhibits shorter mapping time and lower communication energy compared to FASA, FTDTMS, FTNoC, and FTTM algorithm (2-D NoC). The findings in Fig. 5 affirm the

TABLE I
MAPPING TIME AND LATENCY OF FASA [16], FTDTMS [18], FTNOC [19] AND PROPOSED FTTM ALGORITHM ON 2-D NOC & MNOC

	FASA	FTDTMS	FTNoC	FTTM (2-D NoC)	FTTM (MNoC)
Total latency (clock cycles)	92467	91926	89984	86641	84592
Longest path latency (clock cycles)	30864	29986	28768	27192	26853
Mapping time (ms)	124578	112624	108675	100262	99184

TABLE II
COMMUNICATION ENERGY (μJ) OF FASA [16], FTDTMS [18], FTNOC [19] AND PROPOSED FTTM ALGORITHM ON 2-D NOC & MNOC

Communication Energy (μJ)									
FASA [16]		FTDTMS [18]		FTNoC [19]		Proposed FTTM (2-D NoC)		Proposed FTTM (MNoC)	
Zero Fault	One Fault	Zero Fault	One Fault	Zero Fault	One Fault	Zero Fault	One Fault	Zero Fault	One Fault
11300	15300	14200	16200	18300	14300	12500	14500	9500	12500

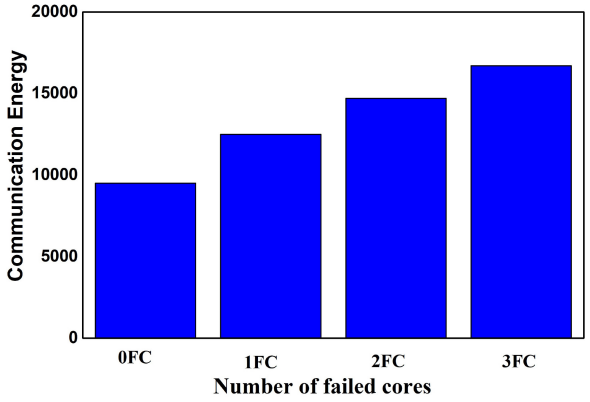

Fig. 5. Impact of core failures on the communication energy (μJ) for FTTM algorithm on MNoC.

TABLE III
BENCHMARK INFORMATION

Application model		# of vertices	# of edges
Multimedia benchmarks	MPEG-4	12	13
	VOPD	16	20
	MWD	12	13
	263 dec	14	15
	263 enc	12	12
	mp3 enc	13	13
Synthetic benchmarks	G1	24	30
	G2	32	36
	G3	64	82
	G4	128	156

consistency and practicality of the proposed FTTM algorithm on MNoC.

To evaluate the contention impact on the task graph on the 6 $\times$ 6 NoC platform, real multimedia benchmarks such as MPEG4, VOPD, MWD, 263dec, 263enc, and Mp3dec [30], along with synthetic benchmarks listed in Table III, were considered [31], [32]. Benchmarks application task graphs generated with the TGFF tool, simulated on 6 x 6, 8 x 8, and 10 x 10 NoC platforms applied FASA, FTDTMS, FTNoC, and proposed FTTM algorithm mapping on 2-D NoC and MNoC. We have calculated communication energy conservation and performance improvement of the FTTM algorithm over FASA and FTDTMS, over FTNoC and FTTM algorithm on 2-D NoC tabulated in Table IV and Table V. The average communication energy conservation and performance for simulated applications are given in the last row of Table IV and Table V. If any core fails in an application mapping, the communication energy conservation and performance will decrease automatically. The failed core task migrates to the spare core. The proposed FTTM algorithm on MNoC enhanced 16.275%, 14.18%, 11.65% and 9.27% on the communication energy conservation, 14.215%, 12.03%, 10.34% and 8.26% performance correspondingly compared with FASA, FTDTMS, FTNoC, and FTTM algorithm on 2-D NoC with respect to multimedia benchmarks, Similarly, the proposed FTTM algorithm on MNoC enhanced 18.1325%, 13.23%, 12.067% and 9.027% on the communication energy conservation, 16.545%, 12.7625%, 11.4075% and 8.29% performance correspondingly compared with FASA, FTDTMS, FTNoC, and FTTM algorithm on 2-D NoC with respect to synthetic benchmarks.

Benchmarks application core graphs are simulated on a 6 x 6 NoC platform applied FASA, FTDTMS, FTNoC, and proposed FTTM algorithm mapping on 2-D NoC and MNoC. We have calculated communication energy concerning 0FC 1FC, and 2FC which denotes the zero-faulty core, one-faulty core, and two-faulty cores respectively, tabulated in Table VI. The simulated results clearly show that the proposed FTTM algorithm on MNoC, communication energy reduction efficiency by an average of 10.4%, 12%, 18% and 6.8% when compared to FASA, FTDTMS, FTNoC, and proposed FTTM algorithm mapping on 2-D NoC concerning 0FC, 14%, 16.2%, 12% and 8.4% communication energy reduction efficiency when compared to FASA, FTDTMS, FTNoC, and proposed FTTM algorithm mapping on 2-D NoC concerning 1FC, and 18%, 25%, 16.2%, and 11% communication energy reduction efficiency when compared to FASA, FTDTMS, FTNoC, and proposed FTTM algorithm mapping on 2-D NoC concerning 2FC as portrayed in Fig. 6. The discrepancy in communication energy between FTTM and FTNoC for 1FC on 2-D NoC is indeed due to the dual nature of FTTM, accommodating both faulty and non-faulty conditions. Specifically, in non-faulty conditions (0FC), FTTM exhibits lower communication energy, as it caters to a broader spectrum of scenarios. Furthermore, the comparable communication energy of FTTM on MNoC with respect to 1FC compared to FTNoC highlights the

TABLE IV
EVALUATION OF COMMUNICATION ENERGY CONSERVATION AND PERFORMANCE IMPROVEMENT OF PROPOSED FTTM ALGORITHM ON MNOC AGAINST OF FASA [16] AND FTDTMS [18]

		NoC Size											
		Proposed FTTM algorithm on MNoC against FASA [16]						Proposed FTTM algorithm on MNoC against FTDTMS [18]					
		Communication Conservation Energy (%)			Performance Improvement (%)			Communication Conservation Energy (%)			Performance Improvement (%)		
	Graph	6 x 6	8 x 8	10 x 10	6 x 6	8 x 8	10 x 10	6 x 6	8 x 8	10 x 10	6 x 6	8 x 8	10 x 10
Multimedia benchmarks	MPEG4	16.24	18.3	20.16	13.6	15.8	17.2	14.28	16.5	17.8	11.2	12.4	14.4
	VOPD	17.12	18.6	20.64	14.8	17.4	19.3	15.4	16.4	18.6	12.8	13.5	15.72
	MWD	13.2	15.8	17.92	10.8	12.9	14.14	11.8	12.9	13.9	10.8	11.2	13.1
	263Dec	14.6	16.4	18.54	12.8	14.89	16.25	12.7	14.1	14.9	11.6	12.5	13.6
	263Enc	12.24	13.95	15.98	10.2	11.4	12.92	10.9	12.12	13.1	9.4	10.3	11.4
	Mp3dec	13.2	14.6	16.33	11.4	12.9	13.8	12.4	13.1	13.9	11.2	12.3	13.42
	Average	**14.43**	**16.275**	**18.26**	**12.26**	**14.215**	**15.6**	**12.91**	**14.18**	**15.37**	**11.17**	**12.03**	**13.6**
Synthetic Bench-marks	G1	18.44	19.22	19.45	16.1	17.6	18.5	13.21	13.98	14.56	13.45	13.8	14.6
	G2	18.72	19.26	19.48	16.36	17.8	18.64	13.36	14.2	14.9	13.68	13.95	14.79
	G3	19.14	19.85	19.9	16.48	18.18	18.82	13.82	14.54	14.96	13.85	14.2	15.1
	G4	13.19	14.2	14.42	11.63	12.6	13.51	9.4	10.2	10.8	8.6	9.1	9.84
	Average	**17.3725**	**18.1325**	**18.3125**	**15.1425**	**16.545**	**17.3675**	**12.4475**	**13.23**	**13.805**	**12.395**	**12.7625**	**13.5825**

TABLE V
EVALUATION OF COMMUNICATION ENERGY CONSERVATION AND PERFORMANCE IMPROVEMENT OF PROPOSED FTTM ALGORITHM ON MNOC AGAINST FTNOC [19] AND PROPOSED FTTM ALGORITHM ON 2-D NOC

		NoC Size											
		Proposed FTTM algorithm on MNoC against FTNoC [19]						Proposed FTTM algorithm on MNoC against 2-D NoC					
		Communication Conservation Energy (%)			Performance Improvement (%)			Communication Conservation Energy (%)			Performance Improvement (%)		
	Graph	6 x 6	8 x 8	10 x 10	6 x 6	8 x 8	10 x 10	6 x 6	8 x 8	10 x 10	6 x 6	8 x 8	10 x 10
Multimedia benchmarks	MPEG4	10.8	12.6	14.92	9.6	10.28	11.41	7.4	9.6	11.4	7.1	8.4	9.1
	VOPD	11.4	13.2	14.8	10.2	11.5	12.72	9.1	10.3	12.4	8.1	9.9	10.4
	MWD	9.6	10.42	11.19	8.8	9.92	10.8	7.8	8.2	9.6	6.5	7.9	8.4
	263Dec	10.52	11.98	12.72	9.4	10.32	11.26	8.4	9.8	10.4	7.6	7.2	8.4
	263Enc	9.12	10.24	11.57	8.94	9.56	10.22	7.6	8.5	9.6	7.2	7.4	8.1
	Mp3dec	10.6	11.5	12.2	9.68	10.46	11.65	8.8	9.2	10.4	7.9	8.8	9.8
	Average	**10.34**	**11.65**	**12.9**	**9.43**	**10.34**	**11.34**	**8.1**	**9.27**	**10.6**	**7.4**	**8.26**	**9.03**
Synthetic Bench-marks	G1	12.4	13.1	13.8	11.4	11.9	12.6	8.6	9.1	9.65	7.9	8.41	8.68
	G2	12.86	13.43	14.1	11.72	12.4	12.9	8.9	9.31	9.8	8.4	8.52	8.71
	G3	13.1	13.6	14.24	12.44	12.92	13.21	9.12	9.9	10.48	8.56	8.8	9.02
	G4	8.02	8.14	8.32	8.16	8.41	8.72	7.64	7.8	8.24	7.2	7.43	7.61
	Average	**11.595**	**12.067**	**12.615**	**10.93**	**11.4075**	**11.857**	**8.565**	**9.027**	**9.5425**	**8.015**	**8.29**	**8.505**

TABLE VI
BENCHMARKS COMMUNICATION ENERGY (μJ) USING VARIOUS ALGORITHMS CONCERNING ZERO-FAULTY CORE (0FC), ONE-FAULTY CORE (1FC), AND TWO-FAULTY CORES (2FC)

Benchmarks	FASA [16]			FTDTMS[18]			FTNoC [19]			FTTM on 2-D NoC			FTTM on MNoC		
	0FC	1FC	2FC	0FC	1FC	2FC	0FC	1FC	2FC	0FC	1FC	2FC	0FC	1FC	2FC
MPEG4	6124	5876	6704	6864	6026	7228	6136	5216	6546	5044	5896	6200	4986	5266	5586
VOPD	5068	4986	5896	5462	5224	6024	5284	4784	5846	4928	4886	5582	4682	4896	5248
MWD	1679	1532	2452	1826	1628	2664	1828	1482	2698	1468	1524	2184	1396	1496	1946
263Dec	32.5	30.5	42.5	34	39	46	34	29.5	40.5	28	30.5	34	26.5	29.5	31.5
263Enc	398	369	464	412	446	476	426	366	446	368	378	444	324	352	426
Mp3dec	26.2	25.4	34.6	28	27.6	36.6	29.4	24.8	33.9	24.4	25.6	34	22.2	24.8	32

TABLE VII
BENCHMARKS EXECUTION TIME (S) USING VARIOUS ALGORITHMS CONCERNING ZERO-FAULTY CORE (0FC), ONE-FAULTY CORE (1FC), AND TWO-FAULTY CORES (2FC)

Benchmarks	FASA [16]			FTDTMS [18]			FTNoC [19]			FTTM on 2-D NoC			FTTM on MNoC		
	0FC	1FC	2FC	0FC	1FC	2FC	0FC	1FC	2FC	0FC	1FC	2FC	0FC	1FC	2FC
MPEG4	1.42	12.8	66	1.5	13.6	72	1.622	10.8	60.42	1.2	10.4	52.4	1.02	8.32	48.26
VOPD	1.26	10.4	44.2	1.7	12.6	48.4	1.8	8.525	38.5	1.1	7.24	36.4	0.9	6.06	34.8
MWD	1.34	7.46	126.6	1.42	8.2	132.4	1.64	5.964	120.9	1.2	6.04	118	1	5.02	106.4
263Dec	1.42	38.8	968.2	1.68	41.4	974	1.56	36.89	942.6	1.06	38	926.4	1.02	32.6	869.2
263Enc	1.16	3.82	38.4	1.28	4.4	42.6	1.58	3.26	36.2	0.84	3.2	38	0.68	2.92	29.4
Mp3dec	1.18	7.96	124.14	1.26	8.24	126.8	1.696	6.25	114.8	0.9	6.6	116	0.712	5.42	102.8

adaptability and efficiency of FTTM across various network architectures.

Performance is assessed as the entire processor's execution time, which contains task execution time, waiting time, and migration time (when faults occur). Execution time of FASA, FTDTMS, FTNoC, and proposed FTTM algorithm mapping on 2-D NoC and MNoC concerning 0FC 1FC, and 2FC that denotes the zero-faulty core, one-faulty core, and two-faulty cores respectively, tabulated in Table VII. The proposed FTTM algorithm on MNoC shows the reduction of execution

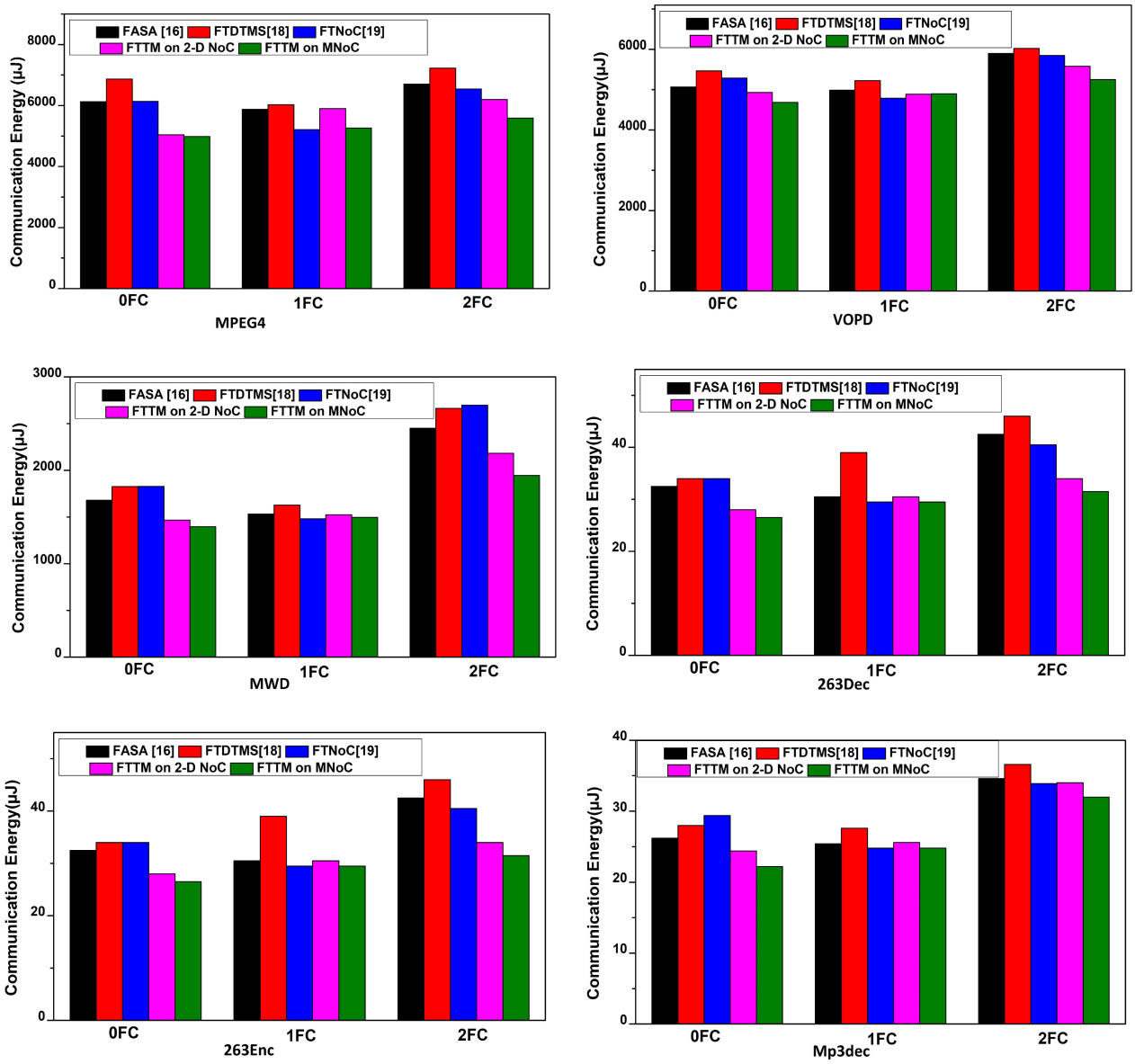

Fig. 6. Comparison of Benchmarks Communication Energy (μJ) using Various Algorithms concerning Zero-Faulty Core (0FC), One-Faulty Core (1FC), and Two-Faulty Cores (2FC).

time by an average of 18%, 22%, 24% and 14% when compared to FASA, FTDTMS, FTNoC, and proposed FTTM algorithm mapping on 2-D NoC concerning 0FC, 12%, 19%, 8% and 8.6% reduction of execution time when compared to FASA, FTDTMS, FTNoC, and proposed FTTM algorithm mapping on 2-D NoC concerning 1FC, and 22%, 28%, 16% and 14.2% reduction of execution time when compared to FASA, FTDTMS, FTNoC, and proposed FTTM algorithm mapping on 2-D NoC concerning 2FC as portrayed in Fig. 7. As for the extended execution time observed for 263Enc and Mp3dec with 2FC of FTTM on 2-D NoC, this trade-off is a result of prioritizing fault tolerance. FTTM, designed to handle both faulty and non-faulty scenarios, incurs a slightly longer execution time for specific tasks. However, the benefits in terms of enhanced fault tolerance capabilities justify this trade-off. Finally, the proposed FTTM algorithm on MNoC outperforms FASA, FTDTMS, FTNoC, and proposed FTTM mapping on 2-D NoC algorithms in terms of communication energy reduction efficiency and performance improvement.

Furthermore, we conducted a comparative analysis of the application mapping utilizing our proposed method alongside other recent mapping algorithms on an MPEG-4 benchmark across various m x n platforms, including 5 x 6, 6 x 8, and 8 x 10. Fig. 8, illustrates the comparison specifically on a 5 x 6 NoC platform. The application model of MPEG-4 with communication weights is outlined in Fig. 8(a). The application of the FASA [16] mapping algorithm on MPEG-4, utilizing bandwidth mapping on the 5 x 6 NoC platform, is visualized in Fig. 8(b). Additionally, FTDTMS [18] applied on MPEG-4 is portrayed in Fig. 8(c), while FTNoC [19] applied on MPEG-4 is depicted in Fig. 8(d). Our proposed mapping algorithm's application on MPEG-4 is showcased in Fig. 8(e). The communication energy of FASA [16], FTDTMS [18],

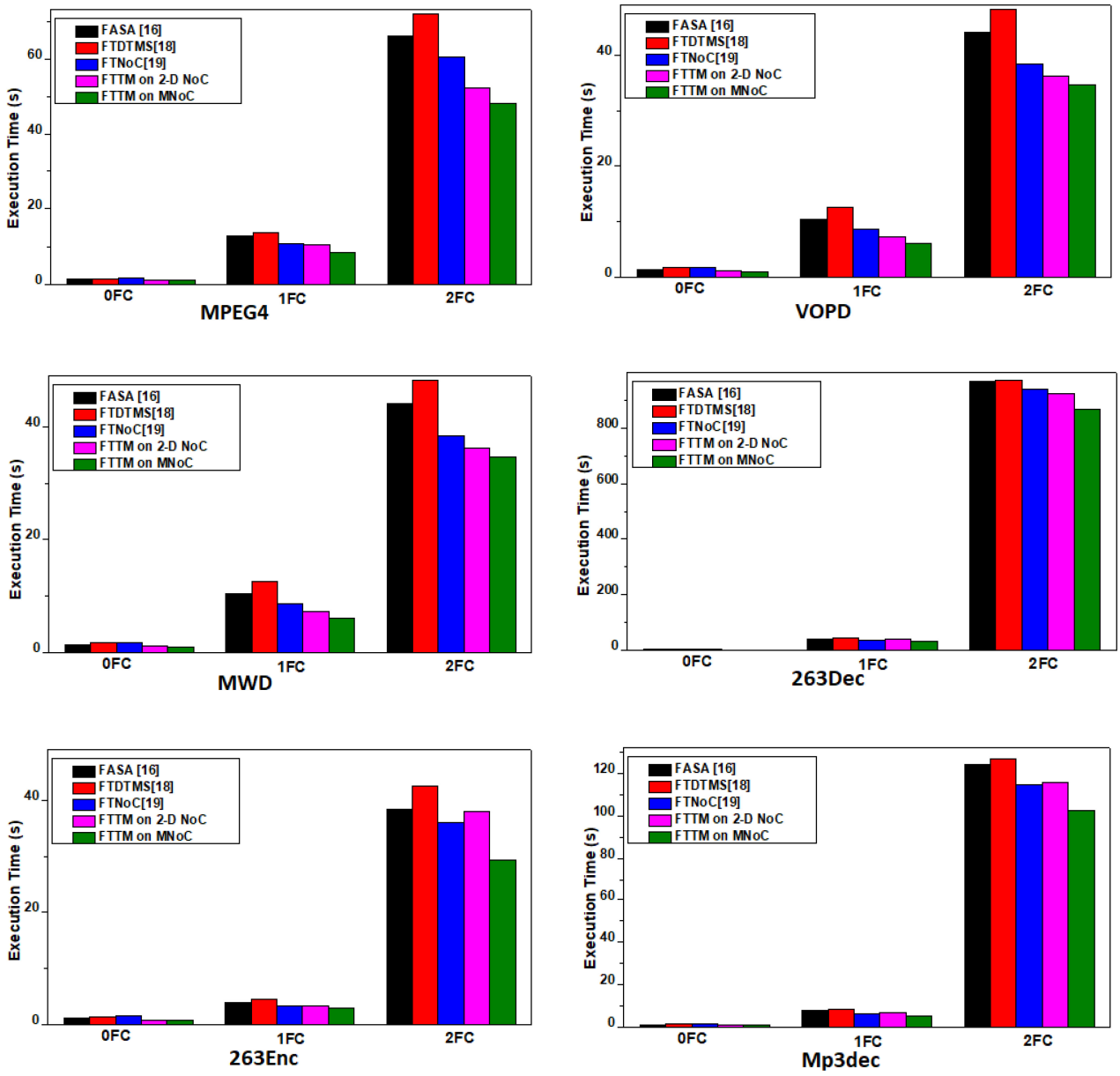

Fig. 7. Comparison of Benchmarks Execution Time (s) using Various Algorithms concerning Zero-Faulty Core (0FC), One-Faulty Core (1FC), and Two-Faulty Cores (2FC).

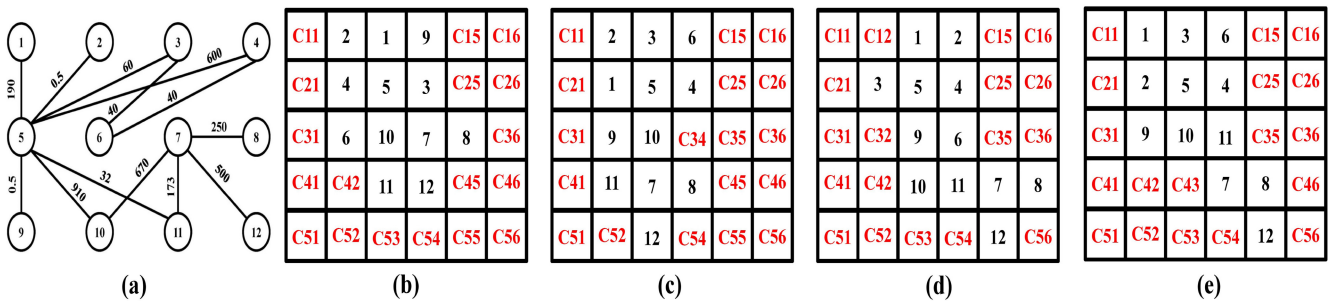

Fig. 8. MPEG-4 Mapping on 5 x 6 NoC platform: (a) Application model, (b) FASA [16], (c) FTDTMS [18], (d) FTNOC [19], and (e) Proposed FTTM algorithm.

FTNoC [19], and our proposed algorithm mappings on the modified NoC is tabulated according to eq. (4) in Table VIII. This comparison highlights the efficacy of our proposed algorithm in task mapping, demonstrating its adaptability not only on square architectures of size m x m but also on rectangular architectures with dimensions m x n.

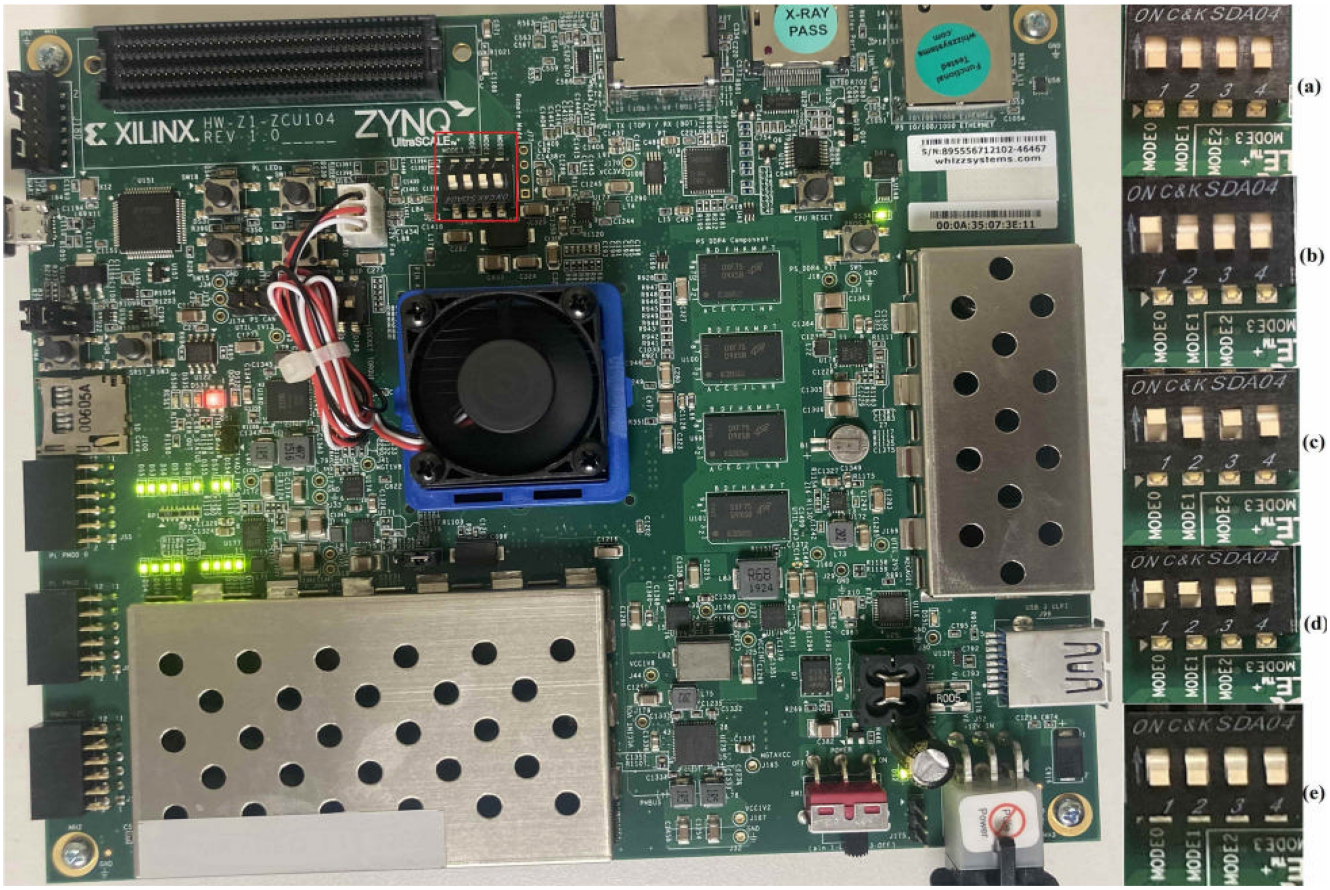

Fig. 9. DIP Switch SW6 configuration on Zynq UltraScale- MPSoC ZCU104 Evaluation Kit, (a) Zero failure, (b) Fault injected at one core, (c) Fault injected at one core and the spare core is activated, (d) Fault injected at two cores and (e) Fault injected at two cores and the spare cores are activated.

TABLE VIII
COMMUNICATION ENERGY (μJ) FOR MPEG-4 MAPPING ON A 5 X 6 NOC PLATFORM USING FASA [16], FTDTMS [18], FTNOC [19], AND THE PROPOSED FTTM ALGORITHM ON MNOC

Communication Energy (μJ)			
FASA [16]	FTDTMS [18]	FTNoC [19]	Proposed FTTM algorithm on MNoC
3752	3531	5190	3466.5

VII. HARDWARE IMPLEMENTATION

One of the novel contributions of this research is the implementation of fault-tolerant task mapping (FTTM) on Field-Programmable Gate Arrays (FPGAs). The implementation was carried out on various Network-on-Chip (NoC) approaches, including MNoC, 2-D NoC, FASA [16], FTDTMS [18], and FTNoC [19], using the Zynq UltraScale MPSoC ZCU104 Evaluation Kit [33]. The main motivation behind the hardware implementation is to assess the overall performance and run-time behaviour of the processor. The NoC task mapping was performed at a clock frequency of 250 MHz with a flit size of 64 bits, 64 bits per packet, and a modified XY routing algorithm. Fault tolerance was introduced to the system using SystemVerilog. The NoC architecture was embedded with the FTTM discussed in Section V. The performance of the task mapping and spare core placement was evaluated using multimedia benchmarks that could be implemented on an FPGA. The results from the Noxim simulation showed significant improvement when compared to the other three related algorithms [16], [18], [19].

TABLE IX
FPGA FAULT INJECTION SWITCH OPERATION

Fault Information (SW6)		Spare Core Activation (SW6)		swith operation
Mode0	Mode1	Mode2	Mode3	
0	0	0	0	Zero Failure
1	0	0	0	One core is failed and spare core is not activated
1	0	1	0	One core is failed and one spare core is activated
1	1	0	0	Two cores are failed and spare cores are not activated.
1	1	1	1	Two cores are failed and two spare cores are activated
Note: 0 - Low, 1- High				

In general, there are multiple ways to inject faults into the Zynq Evaluation Kit to test the FTTM methodology. In this research, permanent faults (stuck-at faults) were injected into the core with the highest communication activity. The faults were injected using switches on the FPGA board, demonstrating that fault tolerance can be achieved through the use of a spare core. The Zynq Evaluation Kit has a 4-bit DIP switch (SW6) with four modes (mode 0, mode 1, mode 2, and mode 3). Modes 0 and 1 are used to configure the faults, while modes 2 and 3 activate the spare core. The configuration of the 4-bit DIP switch (SW6) is summarized in Table IX.

TABLE X
COMPARISON OF TASK EXECUTION TIME ON AN FPGA AMONG PROPOSED FTTM ON MNOC, FASA [16], FTDTMS [18], FTNOC [19], AND 2-D NOC CONCERNING ZERO-FAULTY CORE (0FC), ONE-FAULTY CORE (1FC), AND TWO-FAULTY CORES (2FC)

Benchmarks	FASA [16]			FTDTMS [18]			FTNoC [19]			FTTM on 2-D NoC			FTTM on MNoC		
	0FC	1FC	2FC	0FC	1FC	2FC	0FC	1FC	2FC	0FC	1FC	2FC	0FC	1FC	2FC
MPEG4	28.34	54.1	99.6	25.6	52.91	97.16	26.8	53.44	99.13	25.5	51.4	97.11	24.32	48.65	89.28
VOPD	19.2	36.4	69.7	19	35.9	67.84	18.9	35.26	67.12	17.9	33.56	62.4	17.25	31.3	58.43
MWD	12.34	22.46	42.18	11.9	21.8	41.9	11.46	21.1	40.82	10.9	19.68	37.4	10.4	18.97	35.1
263Dec	4.04	7.9	13.6	4	7.5	13.2	3.8	7.1	12.4	3.4	6.7	11.5	3.2	6.1	10.2
263Enc	2.86	5.1	9.4	2.71	4.9	8.57	2.64	4.78	8.16	2.46	4.52	7.9	2.1	3.95	7.48
Mp3dec	1.72	3.21	6.18	1.6	3.1	5.9	1.46	2.9	5.7	1.39	2.72	5.48	1.35	2.46	4.86

TABLE XI
EVALUATION OF AREA AND ON-CHIP POWER OF FTTM ON MNOC, FASA [16], FTDTMS [18], FTNOC [19] AND 2-D NOC CONCERNING ZERO-FAULTY CORE (0FC), ONE-FAULTY CORE (1FC), AND TWO-FAULTY CORES (2FC)

Algorithm	Zero Fault			One fault is injected and spare core is activated			Two faults are injected and two spare cores are activated		
	Area		On-chip power (in mW)	Area		On-chip power (in mW)	Area		On-chip power (in mW)
	LUT's	FF's		LUT's	FF's		LUT's	FF's	
FASA [16]	24572	26486	0.315	27524	32646	0.38	29856	34568	0.405
FTDTMS[18]	24964	27868	0.326	28690	34524	0.41	31964	33654	0.43
FTNoC[19]	25368	28986	0.334	26925	31648	0.37	28564	32984	0.39
FTTM on 2-D NoC	19846	20988	0.285	23268	26454	0.306	25486	28642	0.32
FTTM on MNoC	16548	18542	0.272	19848	24624	0.294	21468	23546	0.305

The different switch modes for the 4-bit DIP switch (SW6) are shown in Fig. 9. The red region highlights the four switch modes. Each of the switch modes is described below:

1. Zero-failure: When all four modes of the Zynq Evaluation Kit's 4-bit DIP switch (SW6) are configured to low (0), it is considered as zero-failure.

2. Fault injected at one core: When mode 0 of the Zynq Evaluation Kit's 4-bit DIP switch (SW6) is configured to high (1) and the others are low (0), it is considered as a single failed core.

3. Fault injected at one core and the spare core is activated: When mode 0 and mode 2 of the Zynq Evaluation Kit's 4-bit DIP switch (SW6) are configured to high (1) and the others are low (0), it is considered as a single failed core with an activated spare core.

4. Fault injected at two cores: When mode 0 and mode 1 of the Zynq Evaluation Kit's 4-bit DIP switch (SW6) are configured to high (1) and the others are low (0), it is considered as two failed cores.

5. Fault injected at two cores and the spare cores are activated: When all four modes of the Zynq Evaluation Kit's 4-bit DIP switch (SW6) are configured to high (1), it is considered as two failed cores with two activated spare cores.

The spare cores will transmit or receive the data flits to or from the cores that are connected with the failed core. It is important to understand the execution time to comprehend the fault-tolerant task mapping. The execution time on an FPGA is defined as the time taken to complete the entire task mapping. To assess the performance of the proposed FTTM, multimedia benchmarks were implemented on the FPGA board. Task mapping time for Proposed FTTM on MNoC was compared with FASA [16], FTDTMS [18], FTNoC [19], and 2-D NoC across Zero-Faulty Core (0FC), One-Faulty Core (1FC), and Two-Faulty Cores (2FC). Results in Table X reveal that the proposed FTTM on MNoC and 2-D NoC demonstrated shorter task execution times compared to the FASA [16], FTDTMS [18], and FTNoC [19] approaches.

The proposed FTTM algorithm, as depicted in the above ACG (Fig. 4), was implemented on the Zynq Evaluation Kit. Table XI shows a comparison of the evaluated on-chip power and area of the proposed FTTM on 2-D NoC and MNoC, against the FASA [16], FTDTMS [18], and FTNoC [19] approaches, in the case of zero-faulty core (0FC), one-faulty core (1FC), and two-faulty cores (2FC). Area is a critical parameter when it comes to the realization of a NoC on an FPGA. The proposed FTTM algorithm on 2-D NoC and MNoC exhibits superior area efficiency compared to the FASA [16], FTDTMS [18], and FTNoC [19] approaches, as the area is calculated using Look-Up Tables (LUTs) and Flip Flops. For the on-chip power analysis, the proposed fault-tolerant task mapping uses a modified XY routing algorithm and distributes packets along minimum hop paths. The results obtained from both the FPGA and Noxim simulations are consistent and reliable.

VIII. CONCLUSION

In this article, we propose a fault-tolerant task mapping algorithm (FTTM) and a modified NoC platform to map and schedule both offline and runtime tasks on the modified NoC. Our approach optimizes the mapping area and leverages spare core technology in the presence of faults on the NoC platform. The experimental results show that the proposed FTTM algorithm on MNoC and 2-D NoC outperforms other related algorithms in terms of performance and communication energy reductions across various benchmarks, including multimedia and synthetic benchmarks. The FPGA implementation demonstrates the efficiency of the proposed algorithm in terms of area utilization and on-chip power consumption for zero-faulty core (0FC), one-faulty core (1FC), and two-faulty cores (2FC) scenarios. Our analysis highlights the effectiveness of the fault-tolerant task mapping technology. In future work, we plan to extend our approach to 3-D NoCs.

ACKNOWLEDGMENT

The authors extend their sincere gratitude to The School of Electronic Systems and Automation (SoE), Digital University (formerly IIITMK), for their support and provision of hardware boards. Their invaluable contribution played a crucial role in realizing our work, and they deeply appreciate their generosity. The authors also thank the anonymous reviewers for their constructive comments.

REFERENCES

[1] S. Raghav, M. Ruggiero, A. Marongiu, C. Pinto, D. Atienza, and L. Benini, "GPU acceleration for simulating massively parallel many-core platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 5, pp. 1336–1349, May 2015.

[2] D. DiTomaso, R. Morris, A. K. Kodi, A. Sarathy, and A. Louri, "Extending the energy efficiency and performance with channel buffers, crossbars, and topology analysis for network-on-chips," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 21, no. 11, pp. 2141–2154, Nov. 2013.

[3] J. Kim, C. Nicopoulos, D. Park, V. Narayanan, M. S. Yousif, and C. R. Das, "A gracefully degrading and energy-efficient modular router architecture for on-chip networks," in *Proc. 33rd Int. Symp. Comput. Archit.*, 2006, pp. 4–15.

[4] J.-J. Han, M. Lin, D. Zhu, and L. T. Yang, "Contention-aware energy management scheme for NoC-based multicore real-time systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 3, pp. 691–701, Mar. 2015.

[5] N. Michael, Y. Wang, G. Edward Suh, and A. Tang, "Quadrisection-based task mapping on many-core processors for energy-efficient on-chip communication," in *Proc. 7th IEEE/ACM Int. Symp. Netw. Chip (NoCS)*, 2013, pp. 1–2.

[6] Y. Wang, L. Zhang, Y.-H. Han, H.-W. Li, and X. Li, "Data remapping for static NUCA in degradable chip multiprocessors," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 5, pp. 879–892, May 2015.

[7] X. Wang, J. Xi, Y. Wang, P. Bogdan, and S. Nazarian, "An efficient task mapping for manycore systems," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2020, pp. 1–4.

[8] A. S. Kumar and T. V. K. H. Rao, "Efficient core mapping on customization of NoC platforms," in *Proc. IEEE Int. Symp. Smart Electron. Syst.*, 2019, pp. 57–62.

[9] Z. Wu, J. Zhang, F. Fu, and J. Wang, "A fast two-step topology reconfiguration algorithm for core-level fault tolerance in NoCs," in *Proc. 5th Int. Symp. Parallel Archit., Algorithm. Program.*, Taipei, Taiwan, 2012, pp. 86–92.

[10] G. Bizot, F. Chaix, N. Zergainoh, and M. Nicolaidis, "Variability-aware and fault-tolerant self-adaptive applications for many-core chips," in *Proc. 18th IEEE Eur. Test Symp. (ETS)*, Avignon, France, 2013, pp. 1–1.

[11] Z. Wang, A. Littarru, E. I. Ugwu, S. Kanwal, and A. Chattopadhyay, "Reliable many-core system-on-chip design using k-node fault tolerant graphs," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Pittsburgh, PA, USA, 2016, pp. 619–624.

[12] C. Bonney, P. Campos, N. Dahir, and G. Tempesti, "Fault tolerant task mapping on many-core arrays," in *Proc. IEEE Symp. Series Comput. Intell. (SSCI)*, Athens, Greece, 2016, pp. 1–8.

[13] L. Zhang, J. Yang, C. Xue, Y. Ma, and S. Cao, "A two-stage variation-aware task mapping scheme for fault-tolerant multi-core Network-on-Chips," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2017, pp. 1–4.

[14] N. Chatterjee, S. Paul, and S. Chattopadhyay, "Task mapping and scheduling for network-on-chip based multi-core platform with transient faults," *J. Syst. Archit.*, vol. 83, pp. 34–56, Feb. 2018.

[15] F. Khalili and H. R. Zarandi, "A fault-tolerant low-energy multi-application mapping onto NoC-based multiprocessors," in *Proc. IEEE 15th Int. Conf. Comput. Sci. Eng.*, 2012, pp. 421–428.

[16] F. Khalili and H. R. Zarandi, "A fault-tolerant core mapping technique in Networks-on-Chip," *Inst. Eng. Technol. Comput. Digit. Techn.*, vol. 7, no. 6, pp. 238–245, 2013.

[17] P. V. Bhanu, P. Kulkarni, J. Soumya, L. R. Cenkarmaddi, and H. Idsoe, "Torus topology based fault-tolerant Network-on-Chip design with flexible spare core placement," in *Proc. 14th Conf. Ph.D. Res. Microelectron. Electron. (PRIME)*, 2018, pp. 97–100.

[18] N. Chatterjee, S. Paul, and S. Chattopadhyay, "Fault-tolerant dynamic task mapping and scheduling for Network-on-Chip-based multicore platform," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 4, pp. 1–24, 2017.

[19] P. V. Bhanu, P. V. Kulkarni, and J. Soumya, "Fault-tolerant Network-on-Chip design with flexible spare core placement," *ACM J. Emerg. Technol. Comput. Syst. (JETC)*, vol. 15, no. 1, p. 23, 2019.

[20] 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.

[21] A. Bose and P. Ghosal, "The CTH network: An NoC platform for scalable and energy efficient application mapping solution," *IEEE Trans. Nanotechnol.*, vol. 22, pp. 58–69, Jan. 2023, doi: 10.1109/TNANO.2023.323712.

[22] L. Yang, W. Liu, W. Jiang, M. Li, P. Chen, and E. H.-M. Sha, "FoToNoC: A folded torus-like Network-on-Chip based many-core systems-on-chip in the dark silicon era," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 7, pp. 1905–1918, Jul. 2017.

[23] B. Li, X. Wang, A. K. Singh, and T. Mak, "On runtime communication and thermal-aware application mapping and defragmentation in 3D NoC systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 12, pp. 2775–2789, Dec. 2019.

[24] K. Cheshmi, S. Mohammadi, D. Versick, D. Tavangarian, and J. Trajkovic, "A clustered GALS NoC architecture with communication-aware mapping," in *Proc. 23rd Euromicro. Int. Conf. Parallel, Distrib., Netw. Based Process.*, 2015, pp. 425–429.

[25] B. N. K. Reddy and S. Kar, "Energy efficient and high performance modified mesh based 2-D NoC architecture," in *Proc. 22nd Int. Conf. High Perform. Switch. Rout. (HPSR)*, 2021, pp. 1–5.

[26] W. Wu, D. Somasekhar, and S.-L. Lu, "Direct compare of information coded with error-correcting codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 11, pp. 2147–2151, Nov. 2012.

[27] A. Ramdas and O. Sinanoglu, "Testing chips with spare identical cores," *IEEE Trans. Comput., Aided Design Integr. Circuits Syst.*, vol. 32, no. 7, pp. 1124–1135, Jul. 2013.

[28] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Noxim: An open, extensible and cycle-accurate network on chip simulator," in *Proc. IEEE 26th Int. Conf. Appl. Specific Syst., Archit. Process. (ASAP)*, 2015, pp. 162–163.

[29] R. P. Dick, D. L. Rhodes, and W. Wolf, "TGFF: Task graphs for free," in *Proc. Int. Workshop Hardw./Softw. Codesign (CODES)*, 1998, pp. 97–101.

[30] C.-H. Cheng and W.-M. Chen1, "Application mapping onto mesh-based network-on-chip using constructive heuristic algorithms," *J. Supercomput.*, vol. 72, pp. 4365–4378, Nov. 2016.

[31] P. K. Sharma, S. Biswas, and P. Mitra, "Energy efficient heuristic application mapping for 2-D mesh-based network-on-chip," *Microprocess. Microsyst.*, vol. 64, pp. 88–100, Feb. 2019.

[32] S. Mandal, N. Gupta, A. Mandal, J. Malave, J. Lee, and R. Mahapatra, "NoCBench: A benchmarking platform for network on chip," in *Proc. Workshop Unique Chips Syst. (UCAS)*, 2009, pp. 1–8.

[33] (xilinx, San Jose, CA, USA). *Zynq UltraScale+ MPSoC ZCU104 Evaluation Kit*. Accessed: Sep. 2022. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/zcu104.html