IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 4, 2021, accepted March 6, 2021, date of publication March 17, 2021, date of current version March 29, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3066537

Flexible Spare Core Placement in Torus Topology Based NoCs and Its Validation on an FPGA

P. VEDA BHANU[1], RAHUL GOVINDAN[1], PLAVA KATTAMURI[1], J. SOUMYA[1], AND LINGA REDDY CENKERAMADDI[2], (Senior Member, IEEE)

[1]Department of Electrical and Electronics Engineering, Birla Institute of Technology and Science, Pilani-Hyderabad Campus, Hyderabad 500078, India
[2]Department of Information and Communication Technology, University of Agder (UiA), 4879 Grimstad, Norway

Corresponding author: Linga Reddy Cenkeramaddi (linga.cenkeramaddi@uia.no)

This work was supported in part by the Science Engineering Research Board (SERB), Government of India, Research Project, under Grant ECR/2016/001389, Dt. 06/03/2017, and in part by the Indo-Norwegian Collaboration in Autonomous Cyber-Physical Systems (INCAPS) of the INTPART Program from the Research Council of Norway, under Project 287918.

ABSTRACT In the nano-scale era, Network-on-Chip (NoC) interconnection paradigm has gained importance to abide by the communication challenges in Chip Multi-Processors (CMPs). With increased integration density on CMPs, NoC components namely cores, routers, and links are susceptible to failures. Therefore, to improve system reliability, there is a need for efficient fault-tolerant techniques that mitigate permanent faults in NoC based CMPs. There exists several fault-tolerant techniques that address the permanent faults in application cores while placing the spare cores onto NoC topologies. However, these techniques are limited to Mesh topology based NoCs. There are few approaches that have realized the fault-tolerant solutions on an FPGA, but the study on architectural aspects of NoC is limited. This paper presents the flexible placement of spare core onto Torus topology-based NoC design by considering core faults and validating it on an FPGA. In the first phase, a mathematical formulation based on Integer Linear Programming (ILP) and meta-heuristic based Particle Swarm Optimization (PSO) have been proposed for the placement of spare core. In the second phase, we have implemented NoC router addressing scheme, routing algorithm, run-time fault injection model, and fault-tolerant placement of spare core onto Torus topology using an FPGA. Experiments have been done by taking different multimedia and synthetic application benchmarks. This has been done in both static and dynamic simulation environments followed by hardware implementation. In the static simulation environment, the experimentations are carried out by scaling the network size and router faults in the network. The results obtained from our approach outperform the methods such as Fault-tolerant Spare Core Mapping (FSCM), Simulated Annealing (SA), and Genetic Algorithm (GA) proposed in the literature. For the experiments carried out by scaling the network size, our proposed methodology shows an average improvement of 18.83%, 4.55%, 12.12% in communication cost over the approaches FSCM, SA, and GA, respectively. For the experiments carried out by scaling the router faults in the network, our approach shows an improvement of 34.27%, 26.26%, and 30.41% over the approaches FSCM, SA, and GA, respectively. For the dynamic simulations, our approach shows an average improvement of 5.67%, 0.44%, and 3.69%, over the approaches FSCM, SA, and GA, respectively. In the hardware implementation, our approach shows an average improvement of 5.38%, 7.45%, 27.10% in terms of application runtime over the approaches SA, GA, and FSCM, respectively. This shows the superiority of the proposed approach over the approaches presented in the literature.

INDEX TERMS Network-on-chip, application mapping, torus topology, fault-tolerance, spare core, communication cost, FPGA.

I. INTRODUCTION

In the multi-processor era, processing elements are interconnected onto a single chip commonly known as System-on-Chip (SoC). The underlying communication platform used for SoCs design is bus-based architecture. With the increased integration density of processing elements on SoC, the bus-based architectures do not scale well [1]. Hence, there is a need for a suitable communication platform to meet the current application challenges. Network-on-Chip (NoC)

The associate editor coordinating the review of this manuscript and approving it for publication was Jagdish Chand Bansal.

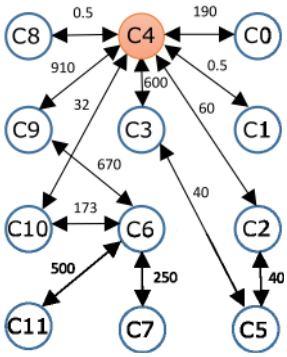


FIGURE 1. MPEG-4 application core graph.

interconnection paradigm has been proposed as a promising solution to address the current application challenges in the field of High Performance Computing (HPC) [2]. NoCs play a major role in the transmission of data from the source node to the destination node in the multi-processor SoCs (MPSoCs). The communication between the cores in NoC is achieved using packet-switching techniques through routers or switches, and interconnection links [3].

A. BACKGROUND

As per Moore's law, the number of transistors integrated onto MPSoCs double every two years [4]. The advent scaling down of the technology, and the validity of Moore's law to the MPSoCs design has led to a focus on reliability. This necessitates the need for fault-tolerant strategies that can improve system reliability. NoC components are susceptible to faults that may occur during the run-time of an application. Transient, Intermittent, and Permanent are three different types of faults that may occur in the components of NoC [5]. Among the three different types of faults, permanent faults are inevitable and vitiates the system performance. Hence, it is highly important to address the permanent faults that may occur in cores of an application.

Application mapping and core placement are traditional problems that have been addressed by several researchers [6]. However, most of the application mapping and core placement techniques that have considered core faults which are limited to Mesh topology only. Torus topology is an improved version of Mesh and has advantages over it in terms of hop count, communication latency, and throughput [7]. Core placement in any topology is depended on the application mapping techniques. Therefore, it is essential to address the problem of fault-tolerant application mapping onto Torus topology by considering the permanent faults in the cores. This can be illustrated with an example of multimedia benchmark Moving Pictures Expert Group (MPEG) - 4, which is one of the most frequently used applications in hand-held devices.

Fig. 1 shows the MPEG-4 application core graph that has twelve cores communicating with each other in terms of Mega bits per second (Mbps). We assume that each task in MPEG-4 is assigned to one core and the size of each core is uniform. As mentioned earlier, the NoC communication platform for any benchmark applications can be designed using any one of the regular topologies namely Mesh and Torus. From the literature [6], [8], it has been noticed that for an MPEG-4 application, Mesh topology based NoC communication platform has been provided as a primary solution. This is due to ease of implementation of Mesh topology based NoCs. However, a little importance is given to Torus topology while designing a NoC communication platform for MPEG-4 application. The wraparound links in Torus topology help to minimize the communication cost, network latency, and average network power consumption of an application. For multimedia benchmarks, the Torus topology adds an advantage in terms of performance metrics namely communication cost, network latency, and power consumption. The MPEG-4 application is most widely used in the domain of video processing units. The permanent faults in any one of the cores of an MPEG-4 application might lead to undesired response, which in turn halt or suspend the system. To mitigate this issue, there is a need for communication efficient and reliable fault-tolerant technique that improves the reliability. Therefore, the reliable Torus topology based NoC communication platform can be considered while designing a system. Further, detailed comparison between the Mesh and Torus topologies in terms of number of hops and communication cost is discussed in Motivation section.

To date, there are several software remedies that have been proposed in the literature [8] to address the problem of fault-tolerant application mapping onto NoCs. These techniques provide an abstract or static view of system performance and reliability. However, there is limited focus on the realization of the fault-tolerant solutions on hardware. The authors in [9] have realized the fault-tolerant application mapping on an FPGA by considering the core faults. However, they have not disclosed the architectural aspects and estimation of the FPGA resources required for the implementation of the proposed model [9]. In addition to it, there is also a need to know about the time taken by the proposed model in [9] to deliver the data packets. Moreover, a detailed study of system dynamics on an FPGA is helpful for understanding its behaviour in real world scenario. Therefore, it is highly important to investigate these factors that may be helpful for the realization of fault-tolerant application-mapping onto an FPGA.

B. KEY CONTRIBUTIONS AND ORGANIZATION OF THE PAPER

In this paper, we present a fault-tolerant application mapping onto Torus topology based NoC design. To address the problem of fault-tolerant application mapping, a mathematical formulation based solution namely Integer Linear Programming (ILP) and meta-heuristic algorithm Particle Swarm Optimization (PSO) based solution have been proposed. For the FPGA implementation of the proposed fault-tolerant application mapping solutions, a Virtual Channel (VC) based

NoC router architecture for Torus topology has been implemented. The key contributions of this paper are as follows.

- ILP and PSO based solutions have been proposed for the fault-tolerant application (multimedia and synthetic) mapping onto Torus topology.
- Router addressing scheme has been proposed for the implementation of Torus topology on an FPGA.
- Routing algorithm and run time fault-injection model have been proposed for the FPGA implementation of Torus topology.
- FPGA implementation of fault-tolerant application mapping onto Torus topology.

For the experimentation, we have taken multimedia [6], synthetic application benchmarks [11] that have been evaluated using static and dynamic simulations based environment. These benchmarks are most widely used in the literature to evaluate the effectiveness of the application mapping methodologies. The fault-tolerant solution obtained using static simulations has been implemented on an FPGA. For prototyping fault-tolerant application mapping solutions onto FPGA, dedicated router addressing scheme, run time fault-injection model, and routing algorithm for Torus topology have been implemented. The rest of the paper is organised as follows. Section II briefs the literature survey. Section III describes the problem formulation. Section IV details proposed methodology. Section V presents experimental results. Section VI presents the limitations of this work and Section VII concludes the paper.

C. MOTIVATION

From the literature [6], [8], it has been noticed that the exact method ILP and meta-heuristic algorithm PSO are successful in finding the fault-tolerant mapping solutions for different application benchmarks. This has motivated us to consider the ILP and PSO methodologies for fault-tolerant mapping of applications onto Torus topology. The fault-tolerant application-mapping onto Mesh-based NoC has been addressed in the literature [10], [12]–[15]. Among these one of our previous works [15] has shown significant improvements in the performance parameters in terms of communication cost. However, the formulations (ILP and PSO) proposed in one of our previous works [15] is limited to Mesh topology only. Due to the structural changes in the Torus topology, the formulations proposed for Mesh will not be applicable to it. The wraparound links in Torus topology connects the corner routers with a single hop such that the average hop count is less compared to Mesh topology. Therefore, there is a significant difference between the Mesh and Torus topologies. The major difference is in the formulation of objective function and fitness function using our previously proposed methodologies ILP and PSO [15]. The experimentation has been carried out to see the difference between the mapping information obtained for Mesh and Torus topologies using our approaches (ILP and PSO).

As mentioned above, with the change in the interconnection pattern between the routers in the topology, the performance of an application is varied from one topology to other. The primary goal of this work is to analyse the efficiency of the fault-tolerant application mapping onto Torus topology using our approaches (ILP, PSO) [15] and the approaches [10], [16]–[18]. The fault-tolerant spare core mapping (FSCM) techniques proposed by the same group of authors in [10], [16]–[18] considered the node average distance (NAD) region and map the spare core in the center of the NAD region. This technique has shown improvements over the traditional approaches presented in [12]–[14], [19]. Therefore, the FSCM technique is considered for the experimental evaluation. In addition to the FSCM technique, we have also considered the Simulated Annealing (SA) algorithm [20] and the Genetic Algorithm (GA) [21] for the comparison of results. Though, we modify the SA algorithm [20] and the GA [21] that can be suitable to Torus topology, we limit the comparison between the Mesh and Torus topologies to FSCM [10] technique only. However, a detailed analysis of the SA algorithm and the GA in the context of fault-tolerant application mapping can be seen in experimental section. We have modified the formulations (ILP/PSO) of our previously published work [15] by considering the wraparound links in the Torus topology. This is an attempt to analyse the applicability of our approaches (ILP/PSO) [15] to Torus topology by providing the flexibility to map the spare cores along with the application cores onto the routers. The cost function i.e., *communication cost* is the figure-of-merit for the application mapping problem. It is defined as the product of bandwidth and number of hops required for the cores to get communicated in the topology. Since the performance of an application is relied on the efficiency of the application mapping technique, therefore, it is highly necessary to analyse the fault-tolerant application mapping solutions in NoC.

$$Communication\ cost\ =\ \sum_{\forall edges} (bandwidth\ *\ hops) \quad (1)$$

Table 1 shows the comparison of communication cost (calculated using equation (1)) between our approaches (ILP/PSO) [15] and FSCM [10] for an MPEG-4 application mapped onto 4 × 4 Mesh and Torus topologies, respectively. For a fair comparison between our approaches [15] and FSCM [10], the most communicating core (C4) in the MPEG-4 application (shown in Fig. 1) is assumed as failure in the application. In the event of failure, the spare core (CS) is used to provide fault-tolerance to an application. From Table 1 it can be observed that the total communication cost obtained using our approach for Mesh and Torus topology is 3652 and 3587, respectively. Similarly, the communication cost obtained using FSCM [10] for Mesh and Torus topologies is 5290 and 7283, respectively.

The communication cost obtained using FSCM [10] is high because they have used the NAD region policy to map the application cores along with the spare core. Since the selected NAD region has restrictions in terms of the hop count between the cores, this has resulted in high number

TABLE 1. Communication cost comparison between our approaches (ILP/PSO) and FSCM [10] for Mesh and Torus topologies, respectively.

Edges	Bandwidth (Mbps)	Mesh				Torus			
		Our approach (ILP/PSO)		FSCM [10]		Our approach (ILP/PSO)		FSCM [10]	
		Hop count	Communication Cost	Hop count	Communication Cost	Hop count	Communication Cost	Hop count	Communication Cost
C0-**CS**	190	1	190	1	190	1	190	2	380
C1-**CS**	0.5	4	2	3	1.5	2	1	1	0.5
C2-**CS**	60	3	180	2	120	3	180	2	120
C2-C5	40	1	40	1	40	1	40	1	40
C3-**CS**	600	1	600	2	1200	1	600	2	1200
C3-C5	40	1	40	3	120	1	40	1	40
CS-C8	0.5	2	1	1	0.5	2	1	1	0.5
CS-C9	910	1	910	2	1820	1	910	4	3640
CS-C10	32	3	96	1	32	1	32	3	96
C6-C7	250	1	250	1	250	1	250	1	250
C6-C9	670	1	670	1	670	1	670	1	670
C6-C10	173	1	173	2	346	1	173	2	346
C6-C11	500	1	500	1	500	1	500	1	500
Total			**3652**		**5290**		**3587**		**7283**

Note: **CS** refers to the spare core which has similar communication requirements of failed core **C4**

of hops between the routers in the topology. On contrary, our approaches (ILP/PSO) have provided flexibility to select the routers to map the application cores along with spare one. This has led to achieve less number of hops between the cores resulting in less communication cost. The percentage improvements in communication cost using our approaches (ILP/PSO) for Mesh [15] and Torus topologies over FSCM [10] is 30.96% and 50.74%, respectively. The average hop count for Mesh and Torus topologies using our approaches (ILP/PSO) are 1.61 and 1.30, respectively. Similarly, for FSCM [10] the average hop count for Mesh and Torus topologies are 1.61 and 1.69, respectively. With the flexible spare core placement in the Torus topology, there is an improvement of 19.25% and 1.78% over the Mesh topology in terms of the average hop count and communication cost, respectively. Compared to Mesh topology, with the use of modified ILP and PSO formulations for Torus topology, the percentage of improvement in communication cost over FSCM [10] is high. This is due to the advantage of the wraparound links available in the Torus topology and the policy of the placement of spare cores as per the communication requirement of an application. Since there is a significant improvement in terms of average hop count in Torus topology, this has motivated us to apply the modified ILP and PSO formulations to the benchmark applications and synthetic applications generated using the TGFF tool [11].

II. RELATED WORKS

Fault-tolerant application mapping is becoming an important problem that needs to be addressed during the design time of an application. Table 2 presents the comparison of application-mapping techniques that have been presented in the literature. The authors in [8] have presented the summary on fault-tolerant application mapping techniques that have optimized the parameters such as communication cost, latency, area, and energy consumption. This survey highlights several methodologies that are proposed to address the core faults in an application. The authors in [12] have presented the fault-aware resource management (FARM) technique to improve system reliability by performing run-time mapping. The FARM technique suggests that the mapping of spare core onto the router (randomly selected from the topology) has shown significant performance improvement over the selection of routers specific to edges and center. The authors in [13], [14], [19] have presented a fault-tolerant spare core allocation (FASA) approach by providing the locations for the spare core to be placed in the Mesh topology. The technique FASA has mapped the spare core near to the most communicating core of an application. The techniques FARM and FASA have fixed the position of spare cores to be placed in the Mesh network. Since, FARM and FASA have performed mapping of healthy cores in an application first and then selected the spare core to be placed in the network. This has resulted in a high communication cost and network latency. To improve system reliability, an energy efficient fault-tolerant application mapping technique has been presented in [10], [16]–[18]. They have selected a region that satisfies the constraint of less number of hops between the routers. Within the region selected, the spare core has been placed in the center of the region. This has resulted in significant improvements over the techniques FARM and FASA, in terms of communication cost and energy consumption while providing reliability to system. The major advantage is that the authors [10], [16]–[18] have considered the spare core while mapping an application onto Mesh topology. The authors in [15] have presented flexible spare core placement while performing application mapping onto the Mesh topology. It has shown significant improvements over the approaches [10], [16]–[18] in terms of communication cost and dynamic performance parameters.

From Table 2, it can be noticed that most of the approaches have focused on 2D Mesh topology. Though, the Torus topology has many advantages in terms of communication cost, power consumption, and network latency, there are very few approaches that have considered it for the fault-tolerant application-mapping. One of our previous

TABLE 2. Comparison of the recent works in the context of application-mapping onto NoCs and its FPGA implementation.

Reference	Year	Problem addressed	Topology	3D/2D	Methodology	FPGA Implementation
[12]	2011	Spare core placement	Mesh	2D	Spare cores are placed either towards side or uniform in the topology using FARM technique	No
[13] [14] [19]	2012 2012 2013	Spare core placement	Mesh	2D	After mapping fault-free cores in an application, the position for spare core is found using FASA	No
[16] [17] [18] [10] [9]	2015 2016 2017 2018 2018	Spare core placement	Mesh	2D	FCSM: Mapping a spare core near to the failed core (mostly at the center) in a fixed Node Average Distance (NAD) area	Yes [9]
[22]	2018	Spare core placement	Torus	2D	Mapping a spare core onto Torus topology using PSO such that it explores the best suitable position which minimize the communication cost	No
[23]	2019	Core placement without any faults	Mesh	2D 3D	Application cores have been mapped onto routers using self adaptive mechanism in chicken swarm optimization algorithm	No
[21]	2019	Core placement without any faults	Mesh	2D	Genetic Algorithm (GA) is used to map the tasks to cores followed by the cores to NoC routers. A dual population based technique has been used to improve the initial population.	No
[24]	2020	Core placement without any faults	Mesh	2D 3D	Comparative analysis and categorization of application mapping techniques have been presented.	No
[20]	2020	Core placement without any faults	Mesh	3D	Simulated Annealing (SA) algorithm has been used for the placement of cores onto Mesh NoCs. Area aware cost function has been proposed using the SA algorithm.	No

works presented in [22], has targeted Torus topology and performed fault-tolerant application-mapping. We have extended the work proposed in [10], [16]–[18] and compared the results in terms of communication cost. However, the work addressed in [22] has limited to static simulations. Recently, the authors in [20], [21], [23], [24] have addressed the problem of application mapping onto NoC topologies without considering any core faults in the application. In [23], a self-adaptive mapping approach has been presented to address the problem of application-mapping onto Mesh-based NoC design. Their approach focuses on fault-free mapping of application cores onto Mesh topology using self-adaptive chicken swarm optimization (SCSO) algorithm. However, the applicability of the SCSO algorithm to map the application cores onto the Torus-topology is to be known. In [24], a comparative analysis on different application mapping techniques onto NoC design and categorization of the trends in implementation of NoCs have been detailed. An area aware cost function has been proposed by the authors in [20]. They have used a variant of the Simulated Annealing (SA) algorithm and proposed a solution to map the cores onto the routers in Mesh topology. However, the SA algorithm based solution is limited to Mesh topology and the applicability of the SA to the problem of fault-tolerant application mapping has to be known. A dual-population based Genetic Algorithm (GA) has been proposed in [21] to address the problem of application-mapping in NoCs. They have fine-tuned the initial population to achieve best quality of mapping solution. Though, the GA algorithm has shown improvements for Mesh topology, the applicability of GA algorithm to the Torus topology has to be known. From Table 2 and the survey [8], it can also be noted that the realization of fault-tolerant mapping solutions on an FPGA is achieved by very few approaches. The authors in [9] have presented the FPGA implementation of fault-tolerant application mapping onto Mesh topology. They have shown the significance of spare core while performing the FPGA implementation. The FPGA implementation presented in [9] has focused on the demonstration of fault-tolerance provided to an application. The details on the router architecture and resource utilization on an FPGA is missing in the approach [9]. Overall, it can be summarized that there exists many approaches that have addressed the problem of fault-tolerant application-mapping onto 2D Mesh topology [8]. These approaches have not focused on the applicability of the methodology to Torus topology. Therefore, there is a need for the investigation of applicability of the fault-tolerant application mapping techniques and evaluations of performance parameters for Torus based NoCs. Also, there is a room for exploring the realization of the fault-tolerant application mapping solutions on an FPGA to understand and analyse the practical behavior of NoCs. According to recent survey [8], this work can be classified as a hardware redundancy based mapping solution that has been proposed to address the core faults while performing application mapping onto Torus topology.

III. PROBLEM FORMULATION

Application mapping in NoC is similar to *Quadratic Assignment Problem* and it is observed to be an NP-Hard [6], [25]. In the DSM level, system reliability can be improved by performing mapping of application cores along with spare core onto NoC topology. The communication characteristics of an application can be represented in the form of a graph known as *Application Core Graph (ACG)*. The connections between the routers using links in the topology is known as *Topology Graph (TG)*. If *'C'* is the set of cores in an ACG, *'E'* is the set of edges annotated with bandwidth (in Mega bits per second), *'R'* is the set of routers in the TG, then connecting each core in *{C}* to a router in *{R}* to minimize the cost function is known as application mapping problem. The application-mapping problem complexity can be defined as, if there are *'n'* cores to be mapped onto *'n'* routers, then the possible combinations of solution is *'n!'*. If the number of cores and the routers are increased then the complexity of the problem may increase further. Similarly, the problem of fault-tolerant application mapping is defined as connecting each core that includes spare core in *{C}* of an ACG to the routers in TG such that cost function is minimized. To address the problem of fault-tolerant application mapping, we have also considered the spare core while performing application mapping onto Torus topology.

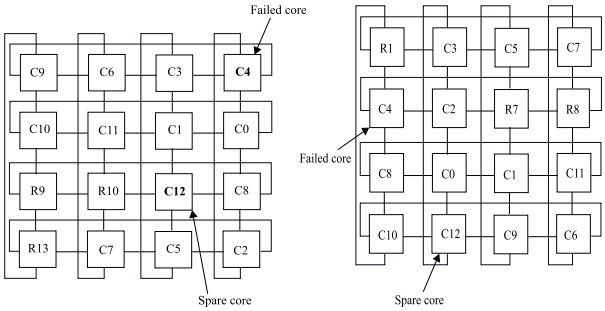


FIGURE 2. Fault-tolerant MPEG-4 application mapped onto 4 × 4 Torus topology.

IV. METHODOLOGY

In this section we present the exact method ILP, meta-heuristic PSO formulations, NoC router architecture, addressing scheme, routing algorithm, and run time fault injection model for software and FPGA based simulations. Since, Torus topology is a modified version of the Mesh with wrap around links, there will be limited modifications in the ILP and PSO formulations proposed by the authors in [15]. The key modifications in ILP and PSO are finding a path between the source core and destination cores that are mapped onto the routers in the Torus topology. Fig. 1 shows the MPEG application core graph having twelve cores (C0-C11), among these cores, core C4 is the highest communicating one. Fig. 2(a) and 2(b) show the fault-tolerant MPEG application mapped onto 4×4 Torus topology (having sixteen routers numbered from R1 to R16) using FSCM [10] and our approach, respectively. Among these sixteen router positions, thirteen are occupied by the application and three of them are left idle.

As it can be observed from Fig. 2, unlike Mesh topology, the connection pattern between the corner and edge routers in the Torus topology are using wrap around links. Therefore, the ILP and PSO formulations proposed by the authors in [15] may not be applied directly for Torus topology. Hence, there is a need for the modifications in ILP and PSO formulations that consider the path using wrap around links in the Torus topology.

A. ILP FORMULATIONS

ILP aims to minimize the objective function i.e., communication cost for the problem of fault-tolerant application mapping onto Torus topology based NoC. For the problem defined in Section IV, the ILP can be formulated as follows. For an edge $e_{ab} \in E$ in the ACG, the cores c_a, $c_b \in C$ have a communication requirement with a bandwidth Bw_{ab}. If cores c_a, c_b are mapped to the routers r_i, $r_j \in R$, then there exists a path $P_{c_a c_b}^{r_i r_j}$ having $D_{r_i r_j}$ number of hops between them. The equations (2) to (10) represent the description of variables, parameters, objective function, mapping constraints, and path constraints, respectively used for the formulation of an ILP. Please note that the equations (2-3) and (5-8) are similar to the formulations of Mesh topology presented in [15]. However, to understand the ILP process flow the equations (modified from [15]) are necessary to be mentioned.

1) VARIABLES AND PARAMETERS

- The mapping variable $M_{c_a}^{r_i}$ is of binary type. If $M_{c_a}^{r_i}$ is one, then core c_a is connected to router r_i, otherwise it is zero.

$$M_{c_a}^{r_i} = \begin{cases} 1, & \textit{Connection exists} \\ 0, & \textit{No connection} \end{cases} \tag{2}$$

- The path variable $P_{c_a c_b}^{r_i r_j}$ is of binary type. If $P_{c_a c_b}^{r_i r_j}$ is one, then there exists a path between the cores c_a, c_b via routers r_i, r_j, otherwise it is zero.

$$P_{c_a c_b}^{r_i r_j} = \begin{cases} 1, & \textit{Path exists} \\ 0, & \textit{No path} \end{cases} \tag{3}$$

- The distance variable $D_{r_i r_j}$ is of integer type. If $r_i \neq r_j$, then the lower and upper limits of distance variable in mxn Torus topology is 1 and $2(m - 1)$. If r_i is equal to r_j, then the distance between the routers is zero.

$$D_{r_i r_j} = \mathbb{Z}^+, 1 \leq D_{r_i r_j} \leq 2(m-1) \tag{4}$$

2) OBJECTIVE FUNCTION

$$\textit{Minimize} \left[\sum_{\forall e_{ab} \epsilon E} Bw_{ab} * \left(\sum_{\forall r_i r_j \epsilon R} D_{r_i r_j} * P_{c_a c_b}^{r_i r_j} \right) \right] \tag{5}$$

3) MAPPING CONSTRAINTS

For fault-tolerant application mapping we have considered the spare core and performed mapping onto routers in the Torus topology.

- Each core $c_a \epsilon C$ including spare core c_s is mapped onto routers $r_i \epsilon R$ in the topology.

$$\forall\, c_a \epsilon C; \sum_{r_i \epsilon R} M_{c_a}^{r_i} = 1 \tag{6}$$

- Each router $r_i \epsilon R$ can have atmost one core $c_a \epsilon C$ connected to it.

$$\forall\, r_i \epsilon R; \sum_{c_a \epsilon C} M_{c_a}^{r_i} \leq 1 \tag{7}$$

From equations (6) and (7), the application cores (including spare core) are mapped onto the routers in the topology. Next, the communication path has to be established using the routing algorithm.

4) PATH CONSTRAINTS

The communication path has to be established for the cores in the ACG. In the event of core failure, the tasks have to be migrated from the failed core to spare core. Unlike ILP formulations presented for Mesh topology in [15], the path constraints for Torus topology are different. We consider the path and distance variables to calculate the number of hops required for the communication. For a given source

and destination core pair, we have used modified XY routing algorithm to find out the routing path and corresponding distance between the routers.

- The pre-calculated distance is stored in the distance array shown below.

$$D_{r_i r_j} = \begin{bmatrix} D_{r_1 r_1} & D_{r_1 r_2} & \cdots & D_{r_1 r_j} \\ D_{r_2 r_1} & D_{r_2 r_2} & \cdots & D_{r_2 r_j} \\ \vdots & \vdots & \ddots & \vdots \\ D_{r_i r_1} & D_{r_i r_2} & \cdots & D_{r_i r_j} \end{bmatrix} \quad (8)$$

- As we have already mentioned in equation (4), that the distance between the routers of same index will be zero.

$$D_{r_i r_j} = \begin{bmatrix} 0 & D_{r_1 r_2} & \cdots & D_{r_1 r_j} \\ D_{r_2 r_1} & 0 & \cdots & D_{r_2 r_j} \\ \vdots & \vdots & \ddots & \vdots \\ D_{r_i r_1} & D_{r_i r_2} & \cdots & 0 \end{bmatrix}$$

- If the cores are mapped onto routers, then there exists a path between the cores via routers (represented in equation (9)). However, there should not be any path in the topology between the failed core c_f and other healthy cores $c_a \in C$ of an ACG. This can be represented by equation (10).

$$\forall e_{ab} \epsilon E, a \neq b \neq f;\ M_{c_a}^{r_i} + M_{c_b}^{r_j} - P_{c_a c_b}^{r_i r_j} \leq 1 \quad (9)$$

$$\forall e_{ab} \epsilon E, a == f; b \neq f\ M_{c_s}^{r_k} + M_{c_b}^{r_j} - P_{c_s c_b}^{r_k r_j} \leq 1 \quad (10)$$

The equations (2) to (10) are given to the ILP solver tool CPLEX [26]. The solution obtained is the fault-tolerant application mapping onto Torus topology based NoC design. This completes the exact method formulations. As the problem complexity is increased, the number of variables used for the ILP formulation increases which results in the out of memory status. Therefore, for the higher network sizes meta-heuristic PSO is proposed.

B. PSO FORMULATION

PSO was proposed by Eberhart and Kennedy in 1995 [27]. It is inspired by the nature of birds swarm and fish schooling to address the problems that involve global and local optimization. Most of the researchers have been successful in applying the discrete version of PSO (DPSO) to address the problem of application mapping in NoC. With this motivation, the authors [15], [22] have applied the DPSO to solve the problem of fault-tolerant application mapping onto Mesh and Torus topology, respectively. The detailed description of PSO can be known from [22]. Since, PSO is one of the contributions in this paper; therefore, authors feel it is necessary to discuss in brief about PSO for the fault-tolerant application mapping onto Torus topology. The step by step procedure required for the formulation of PSO are as follows.

1) INITIALIZATION OF PARTICLES

The first step is to initialize the particles randomly in the problem space to find the optimum solution. Prior to initialization,

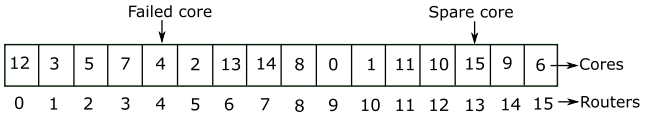


FIGURE 3. Particle Structure for MPEG ACG shown in Fig. 1.

particle structure has to be formulated. Therefore, particle structure can be represented as an array of elements whose entries are cores (including spare core) and indices are routers in the Torus topology. Fig. 3 shows the particle structure of MPEG ACG shown in Fig. 1. Since, Torus topology differs from Mesh in terms of wrap around links only, therefore the particle structure formulated for Mesh [15] and Torus are same. However, the solution differs in the formulation of fitness function in terms of hop array.

2) FITNESS FUNCTION

In PSO, the solution's quality depends on the fitness function, i.e., communication cost defined in equation (1). For better fitness values, the communication cost has to be minimized for each edge in the ACG while providing fault-tolerance to the system. From equation (1), it is evident that the communication cost is dependent on the number of hops between the routers. As mentioned in Section IV.A, the number of hops between the routers in Torus topology is calculated using the modified XY routing algorithm. Therefore, the hop array obtained for Mesh topology might not be useful to Torus topology. Since the router connection pattern and the routing algorithm used for Torus are different from those of Mesh topology, there is a need to formulate PSO for Torus topology. The hop array for Torus topology is shown in equation (8), which differentiates from Mesh topology in terms of the distance between the corner and edge routers.

3) CREATION OF NEW GENERATION

The initial generation of particles is created randomly. The evolution of particles is partially guided by *local best* and *global best*. The *local best* is the minimum communication cost, the particle has seen so far in the process of evolution. The *global best* is the minimum communication cost for a particular generation. The new generation is created by applying the *swap sequences* on a particle. For detailed explanation regarding *local best, global best, swap sequences*, please refer to the PSO section in [15].

4) TERMINATION CRITERIA

PSO can be terminated in two different ways. The first and frequently used terminating condition is to check the generation count. If the generation count for an independent run of PSO is expired, then the PSO is terminated. Secondly, for a certain number of generations if the fitness value is unchanged, then the PSO is terminated. For the experimentation, we have considered the first terminating condition for PSO.

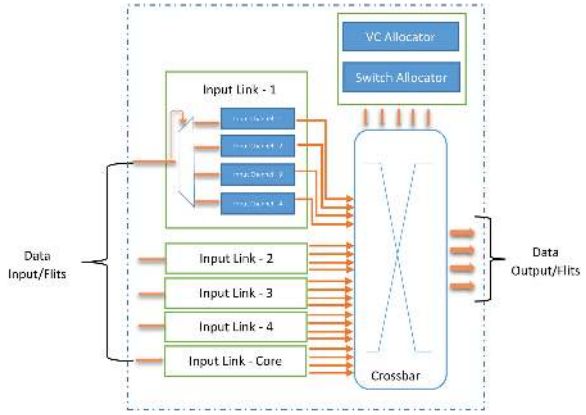


FIGURE 4. NoC router architecture modified version of [28].

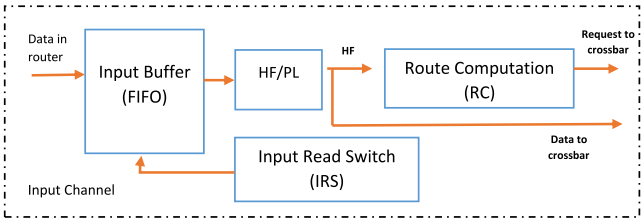


FIGURE 5. Input channel of the router [28].

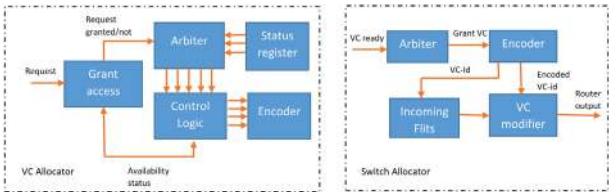


FIGURE 6. Output channel of the router [28].

This briefs the PSO formulation for the problem of fault-tolerant application mapping onto Torus topology.

C. NoC ROUTER ARCHITECTURE AND ADDRESSING SCHEME

In this section, we present the NoC router architecture and addressing scheme used for the implementation of Torus topology based NoC design onto an FPGA. The basic router architecture is taken from the work presented in [28]. The overview of NoC router architecture (modified version of [28]) is shown in Fig. 4.

We have used five port (four links and one core) Virtual Channel (VC) based NoC router. It has one input and output channel. The data flow in input channel and output channel is shown in Fig. 5 and 6, respectively. The application data in NoC is transferred using packet-switching technique. According to this technique, the data packet is divided into Header Flit (HF), Payload Flit (PF), and Tailer Flit (TF). The flits entering into a router will pass through the input channel, via crossbar, and reach the output channel. The input buffer (FIFO) in the input channel collects the flit and pass it to the route computation (RC) unit. In RC unit, the routing algorithm is responsible to request the corresponding output

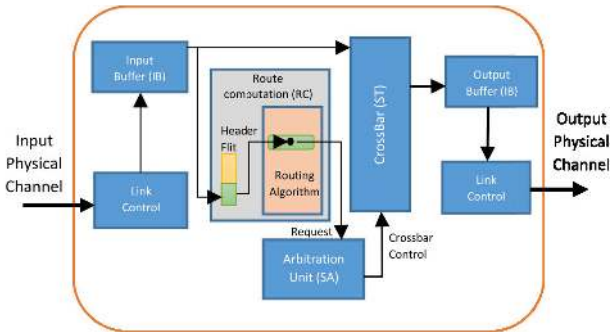


FIGURE 7. The overall data-flow in NoC router [28].

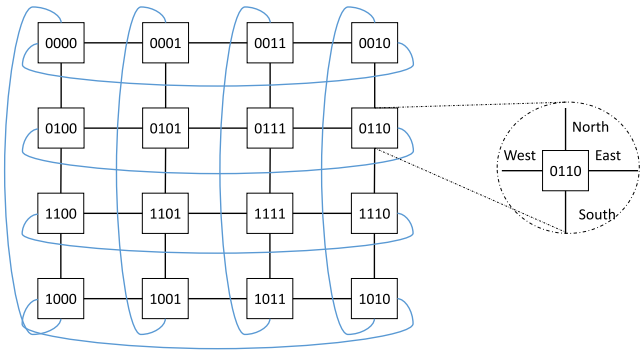


FIGURE 8. Router addressing scheme for 4 × 4 Torus topology.

port to pass the flits from source to destination. The input read switch (IRS) monitors the status of the FIFO. Based on the availability of the FIFO, the next flits will be processed. The output channel majorly consists of VC allocate and Switch allocate modules. They involve in switching the data from an input channel to the output port. The VC allocate module monitors the status of the VCs used in the output links and grant a response to the request received from the input channel. The arbiter is used to grant the response based on the requests received. In the switch allocate module, the output port is assigned with the help of an encoder. In one of the output ports selected, the flits from the input channel will be passed through it by updating its VC-id. The overall data-flow noticed from input port of a router to the output port can be realized using the Fig. 7. It can be noticed that from the entire data flow process that the router addresses in HF and routing algorithm are the key blocks, which can request the corresponding output link for successful delivery of data flits. Therefore, in this work, the router addressing scheme and routing algorithm have also been proposed for the implementation of Torus topology on an FPGA.

Fig. 8 shows the router addressing scheme proposed for the 4 × 4 Torus topology with the indication of North, South, East, and West links of a router. We have used the concept of reflected binary (RB) code for addressing the routers. According to RB coding scheme, the neighbouring routers are differentiated by one bit only. This adds an advantage in terms of the comparison between two routers in the topology by checking single bit change in the router addresses. The router addressing scheme for 4 × 4 Torus topology requires

four bits D[3:0]. The router architecture used for the 4 × 4 Torus topology is of five ports (north, south, east, west, and local). As we can observe from the Fig. 5, the routers can be compared by considering one binary bit in the address. This can be scalable to higher network size of Torus topology.

D. ROUTING ALGORITHM AND RUN TIME FAULT-INJECTION MODEL

1) ROUTING ALGORITHM

In this section, we present the routing algorithm designed for 4 × 4 Torus topology by considering the router addressing scheme discussed in Section IV.C. Algorithm 1 shows the routing algorithm proposed for the Torus topology. Input and output to this algorithm are Header Flit (HF) and current router link request, respectively. Once the HF is received at the current router, the destination router address D[3:0] is taken and compared with the current address C[3:0]. According to this algorithm, the current and destination routers are compared bit wise. Since the router addressing scheme is based on binary, therefore, bit-wise comparisons are done.

As we have mentioned in Section IV.C, the router arbitration unit grants the access to the output links requested by routing algorithm. Therefore, the routing algorithm proposed will request the output links of the current router. According to the Algorithm 1, there are four different conditions to be checked before requesting the current router's output links. The description of each condition is given below.

1) **East or West link:** The first bit C[0] in current address and D[0] in destination address are compared. If they are not equal, then the first bit C[0] and second bit C[1] in the current address will be compared. If C[0] and C[1] are equal, then the proposed algorithm requests the current router's *East link*. If C[0] and C[1] are not equal, then the current router's *West link* is requested. If the first bit in current address C[0] and destination address D[0] are equal, then the proposed algorithm compares with its next bit, i.e., the second bit of the current and destination address.
2) **West or East link:** The second bit C[1] in the current address and D[1] in destination address are compared. If they are not equal, then the first bit C[0] and second bit C[1] in the current address will be compared. If C[0] and C[1] are equal, then the proposed algorithm requests the current router's *West link*. If C[0] and C[1] are not equal, then the current router's *East link* is requested. If the second bit in current and destination address are equal, then the proposed algorithm will compare with its third bit, i.e., C[2] and D[2].
3) **North or South link:** The third bit C[2] in the current address and D[2] in destination address are compared. If they are not equal, then the third bit C[2] and fourth bit C[3] in the current address will be compared. If C[2] and C[3] are equal, then the proposed algorithm requests the current router's *South link*. If C[2] and C[3] are not equal, then it requests *North link* of the current router. If the third bit of the current C[2] and destination address D[2] are equal, then the proposed algorithm compares with its next bit, i.e., C[3] and D[3].
4) **South or North link:** The fourth bit in the current address C[3] and destination address D[3] are compared. If they are not equal, then the third C[2] and fourth bit C[3] in the current address will be compared. If C[2] and C[3] are equal, then the proposed algorithm requests the current router's *North link*. If C[2] and C[3] are not equal, then it requests *South link* of the current router. If the fourth bit of the current C[3] and destination address D[3] are equal, then the proposed algorithm requests the *local link*.

Algorithm 1 Routing Algorithm for Torus Topology

Input : Header Flit (HF)
Output: Link request of current router

```
Read the destination address D[3:0] from HF
for Each HF received at the router do
    Compare the destination address D[3:0] with current address C[3:0]
    if (C[0] ≠ D[0]) then
        if (C[0] == C[1]) then
            request East link
        else
            request West link
    else if (C[1] ≠ D[1]) then
        if (C[0] == C[1]) then
            request West link
        else
            request East link
    else if (C[2] ≠ D[2]) then
        if (C[2] == C[3]) then
            request South link
        else
            request North link
    else if (C[3] ≠ D[3]) then
        if (C[2] == C[3]) then
            request North link
        else
            request South link
    else
        request local link
end
Once the link connection is established PF, TF follow the path
```

2) RUN TIME FAULT-INJECTION MODEL

We have proposed a run time fault-injection model for the FPGA implementation of fault-tolerant application mapping onto Torus topology. In general, there are several ways to inject the faults on an FPGA to test the design. The authors in [29] have proposed an emulation based fault-injection

TABLE 3. FPGA switch operations used for fault injection model.

Fault Information (SW11[s2])	Spare Core Activation (SW11[s3])	Switch Operation
Low	Low	No failure
Low	High	Invalid
High	Low	Core failed and Spare core is not activated
High	High	Core failed and Spare core is activated
Please note: Low-0; High-1;		

control and monitoring technique. They have injected the faults into a specific location of the design, i.e., flip-flops, and observed the functional characteristics while monitoring the state of flip-flops. In contrast to it, our proposed model shown in Algorithm 2 has the flexibility to inject the faults into the cores. Instead of injecting the faults in selected flip-flops of the design, our proposed model considers the failures in a higher level of the design like cores and deactivates it. Since we assume that the most communicating core has high chances of failure, the fault is injected into it using the switches on a Kintex-7 KC705 FPGA. Compared to the work proposed in [29], our proposed model is a proof-of-concept that shows the fault-tolerance can be achieved with the help of spare cores. However, the methodology proposed in [29] can be considered as an extension to this work.

The inputs to our proposed model are Header Flit (HF), fault information (FI), and spare core activation (SCA). The output is to establish a routing path from the spare core to the other cores in an application. Since we have assumed that the most communicating core is failed in an application, the fault is injected into it. However, the fault can be injected into any one of the cores of an application. The term fault injection in this work refers to the deactivation of the most communicating core from the network. Once the fault has been injected in runtime, the failed core will not exchange the flits from other cores in an application. In such a case, based on the spare core activation, the communications associated with failed core will be taken up by the spare core.

Table 3 represents the FPGA onboard switch operations used for the fault-injection model. The first column represents the status of *FI*, the second column represents the status of *SCA*, and the third column represents the switch operations. There are multiple switches on the KC705 FPGA board. We have used the 4-bit DIP switch (SW11) (denoted by the SW11[s4:s1]) to configure the fault-injection model. The Low and High represent the DIP switch status 0 and 1, respectively. Out of the four bits in the DIP switch SW11, the second bit SW11[s2] is used for the injection of fault in the cores, and the third bit SW11[s3] is used for the activation of the spare core. The first bit and fourth bit of SW11 has not been used for the fault-injection model. If SW11[s2] is high, then it denotes that the fault has been injected into the core; otherwise, there is no fault injected in the core. If SW11[s3] is high, then it denotes that the spare core is activated in the design; otherwise, there is no fault-tolerance provided to the design. From Table 3, it can be observed that there

Algorithm 2 Run Time Fault-Injection Model

Input : Header Flit (HF), DIP switch status SW11[s4:s1]
Output: Establish the routing path from the spare core

```
Read the HF, the switch SW11[s2] and SW11[s3] status from the KC705 FPGA
for each HF received at the router do
    if ((SW11[s2] == 0)&&(SW11[s3] == 1)) then
        Establish the routing path between the fault-free cores and spare core.
        Invalid operation
    if ((SW11[s2] == 1)&&(SW11[s3] == 0)) then
        Establish the routing path between the cores except the failed one.
        Unreliable operation
    if ((SW11[s2] == 1)&&(SW11[s3] == 1)) then
        if (source core is failed) then
            Establish the routing path from spare core.
            Spare core sends the HF.
        end
        if (destination core is failed) then
            Establish the routing path from spare core.
            Spare core receives the HF.
        end
    else
        Fault-free cores exchange the HFs
end
Payload Flit (PF), Tailer Flit (TF) follows the routing path established by spare core
```

are four different switch modes that can be used to verify the functionality of the fault-injection model. Fig. 9 shows different switch configurations discussed in Table 3. Each one of these configurations is detailed below.

1) **No failure:** If the second bit (SW11[s2]) and third bit (SW11[s3]) of the switch SW11 is configured to low, then it is considered as fault-free mode. Fig. 9(a) shows the SW11 configuration, it can be seen that the second and third bit is configured to 0 or low. In fault-free mode, the data will be exchanged between the healthy cores of an application.
2) **Invalid:** If the second bit of the switch SW11 is configured to Low and the third bit of the switch SW11 is configured to high, then it is an invalid mode. In this mode, only the spare core is activated which sends the redundant data to the other cores of the application. Therefore, it is considered as the invalid mode.
3) **Fault-injected but the spare core is not activated:** If the second bit of the switch SW11 is configured to High and the third bit of the switch SW11 is configured to Low, then the fault is injected into the core which is assumed to be failed. In the event of failure, the spare

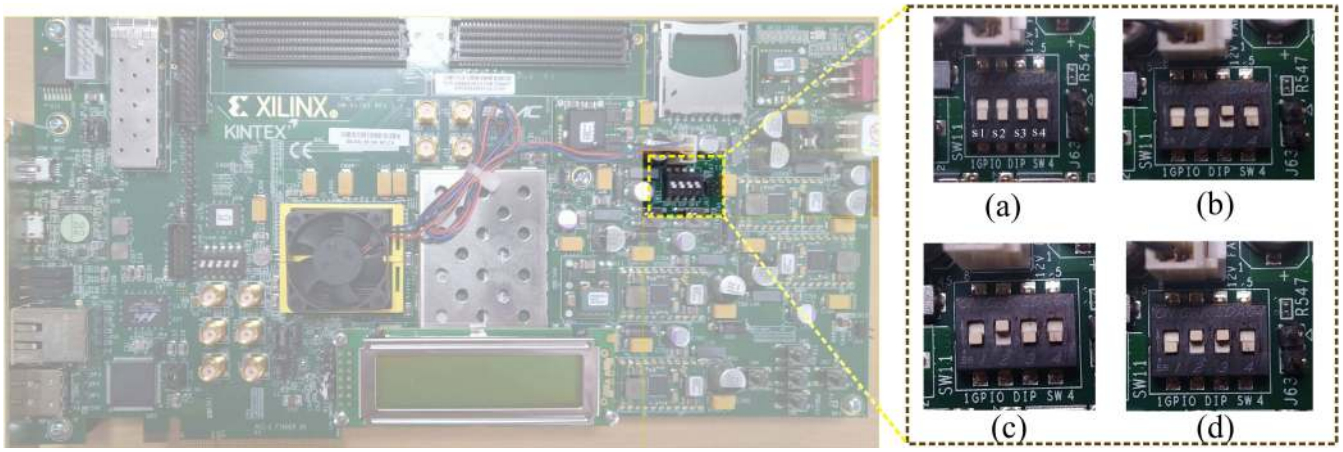


FIGURE 9. DIP Switch SW11 configuration on KC705 FPGA board, (a) No failure, (b) Invalid, (c) fault is injected but spare core not activated, and (d) fault is injected and spare core is activated.

core need to send the data flits to the cores associated with the failed one. Since the spare core is not activated, it will not send the data flits to the other cores. This results in the degradation of system performance.

4) **Fault injected and the spare core is activated:** If the second bit and third bit of the switch SW11 is High, then the fault is injected into the core and the spare core is activated. In the event of core failure, the spare core will send or receive the data flits to or from the cores associated with the failed one. Based on the role of failed core as either source or destination, the spare core will send or receive the data flits by establishing a routing path.

V. EXPERIMENTAL RESULTS

In this section, we demonstrate the fault-tolerant application mapping using software and hardware (FPGA) implementation. The major aim of this study is to understand and analyse the performance parameters of an application with inclusion of spare core while providing fault-tolerance to system. We have taken multimedia application benchmarks [6] and synthetic applications [11] for the experimentation. Further, to analyze the fault-tolerant application mapping solution obtained using our approach and the approaches FSCM [10], SA algorithm [20], GA [21] an FPGA implementation is done. For a fair comparison with our approaches (ILP/PSO), we have extended the approaches based on FSCM technique [10], SA algorithm [20] and the GA [21] to Torus topology by considering the core faults. The experimentations are carried out with the same configuration settings used for our approach and the approaches FSCM [10], SA algorithm [20], and the GA [21]. The experimental flow is shown in Fig. 10 and the results are organized based on the software and FPGA implementation.

The primary input to the experimentations are application-core graphs and the outputs are static, dynamic, and on-chip parameters. The fault-tolerant application-mapping algorithms such as PSO, SA, GA, and FSCM are evaluated using the software and hardware platform and its efficiency is analyzed in terms of solution quality. The quality of mapping solution can be ranked by the communication cost defined in equation (1). In software implementation, we have performed static simulations for our approaches (ILP, PSO) and FSCM [10], SA [20], GA [21]. To understand the static behavior of fault-tolerant application mapping solutions, we have performed the experimentations with the following configurations.

- Scaling the number of cores in the applications (multimedia and synthetic).
- Scaling the Torus network size from 5×5 to 12×12
- Scaling the percentage of router faults in 9×9 Torus network.

Similarly, the dynamic behavior of the proposed solutions is analysed using the cycle accurate NoC simulator and compared the results in terms of network latency (in clock cycles), throughput (in flits/cycle/core), and router power consumption (in mW). In hardware implementation, the fault-tolerant application mapping information is prototyped onto FPGA and on-chip parameters namely resource utilization, application runtime on an FPGA, and static/dynamic power consumption are analyzed. To compare our approach and FSCM [10], SA algorithm [20], and GA [21] the applications are run on an FPGA with the respective mapping information obtained in the software implementation stage and the overall runtime is noted. These application mapping algorithms are validated on an FPGA by providing real application traffic patterns in terms of flits. We have used Virtual Input Output (VIO) IP core to assign the cores to the routers and send the data flits. Based on the analysis, the efficacy of the fault-tolerant application mapping algorithms is known.

A. EXPERIMENTAL SETUP

The algorithms PSO (our approach), SA [20], and GA [21] are coded in high-level language and performed static simulations. They are independently run for 30 times and the best results are reported. The parameters used for the PSO

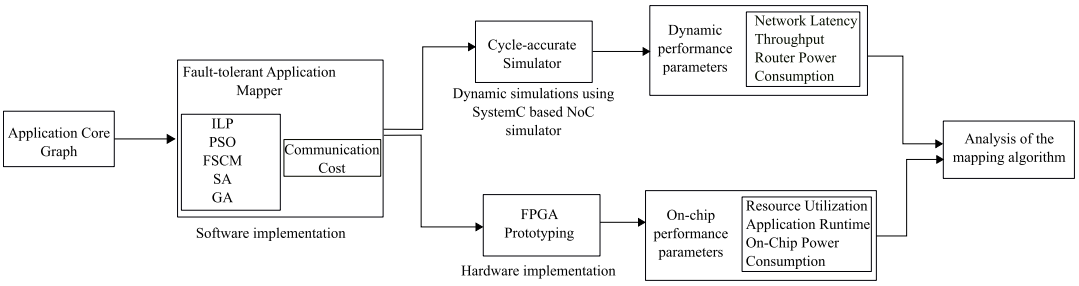


FIGURE 10. Experimental flow of the fault-tolerant application mapping onto Torus topology based NoC.

algorithm are the number of particles (1000), number of generations (200), acceleration co-efficient alpha (0.5), and beta (0.3). The parameters used for the SA algorithm are initial temperature (40), damping rate (0.95), annealing loop (500), and thermal equilibrium loop (10). Similarly, the parameters used for the GA are population size (500), number of generations (100), mutation probability (0.9), and crossover probability (0.1). We have used IBM CPLEX-ILOG tool [26] to run ILP formulations, cycle-accurate NoC simulator [28] for dynamic simulations, and Kintex KC705 FPGA for hardware implementation. The simulator settings are detailed below.

- *Simulation time*: 2,00,000 clock cycles
- *Saturation time*: 10,000 clock cycles
- *Flit size*: 32 bits
- *Clock period*: 5 ns
- *Number of flits per packet*: 64
- *Traffic*: application specific
- *Router ports*: 5 (4 global, 1 local)
- *Mapping information*: Our approach (PSO), FSCM [10], SA [20], and GA [21]
- *Routing algorithm*: Modified XY

Please note for the experimentation (software and FPGA) the most communicating core in the application is assumed to be failed. This assumption is valid in most of the approaches [9], [10], [12]–[14], [16]–[19] reported in the literature.

1) COMPARISON OF COMMUNICATION COST BETWEEN OUR APPROACHES (ILP AND PSO) AND THE APPROACHES FSCM [10], SA [20], AND GA [21] BY SCALING THE TORUS NETWORK SIZE

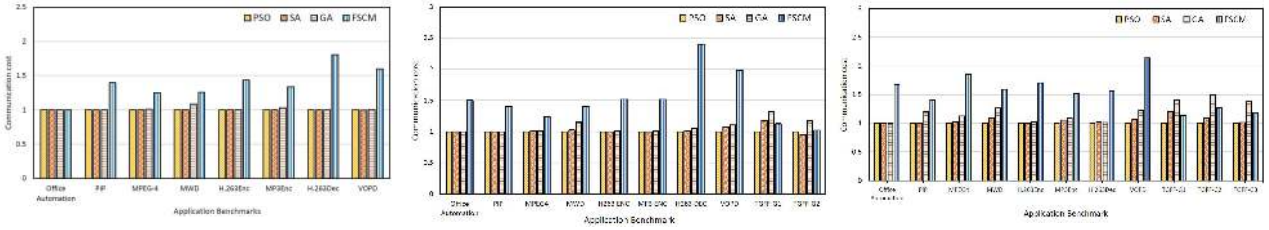
This section presents the experimental results obtained by mapping the application cores (including spare core) onto the routers by scaling the network size. We have assumed that the most communicating core has high chances of failure for the applications considered and performed fault-tolerant application mapping. Table 4 shows the comparison of communication cost between our approach and the approaches FSCM [10], SA [20], and GA [21] by scaling the Torus network size from 5×5 to 12×12. In Table 4, column one represents the application to be mapped onto Torus topology. The second column represents the number of cores present in the application. Columns three to seven, eight to twelve, thirteen to seventeen represent the communication cost results for the 5×5, 9×9, and 12×12, respectively. From the Motivation section, it has been seen that; if there is a fixed region for mapping the cores (including spare core), then it has resulted in a high communication cost. With this motivation, we have modified the approaches SA [20] and the GA [21] without considering any fixed region. This can be considered as a major modification to the approaches SA [20] and GA [21] while extending it to Torus topology. As we can observe from Table 4, the communication cost results for the 5×5, 9×9, 12×12 Torus network obtained using our approaches (ILP and PSO) are less than the approaches FSCM [10], SA [20], and GA [21]. From the experimental results, it has been observed that the FSCM technique used by the authors in [10] has placed the spare core in the center of the NAD region selected in the topology. With the change in communication requirement of the applications, the FSCM [10] has fixed the position for the placement of the spare core in the network. In contrast to FSCM, our approaches (ILP/PSO) and the approaches SA [20] and GA [21] have provided the flexibility to place the spare core while performing the fault-tolerant application mapping. Fig. 11 shows the comparison of the communication costs obtained by the approaches SA [20], GA [21], and FSCM [10]. In Fig. 11, X-axis represents the application benchmarks, and Y-axis represents the communication cost results normalized to our approach PSO algorithm. It can be noted that the approaches SA [20] and the GA [21] have resulted in less communication cost compared to the FSCM technique [10]. From Fig. 11(a), it can be seen that the modified approaches SA [20] and GA [21] have performed well for the smaller application benchmarks. As the number of cores and the network size are increased, the approaches SA [20] and GA [21] have shown very little improvements (Fig. 11(b) and 11(c)) over our approach PSO algorithm. This is due to the increase in the search space from 25 routers (in 5×5) to 81 routers (in 9×9) and 144 routers (in 12×12). As the search space is increased, the algorithms SA [20] and GA [21] have not converged to the best solution. On the other hand, though the search space is increased, our approach PSO has converged to the best solution. This is possible due to the swarm intelligence policy that exists in the PSO algorithm.

Fig. 12 shows the average percentage improvement in communication cost obtained using our approach over the approaches FSCM [10], SA [20], and GA [21] by scaling the network size, from 5×5 to 9×9 and 12×12,

TABLE 4. Communication cost comparison between our approaches (ILP/PSO) and the approaches FSCM [10], SA [20], GA [21], by varying the Torus topology size.

Application	No. of cores	5x5					9x9					12x12				
		Our approach		FSCM	SA	GA	Our approach		FSCM	SA	GA	Our approach		FSCM	SA	GA
		ILP	PSO	[10]	[20]	[21]	ILP	PSO	[10]	[20]	[21]	ILP	PSO	[10]	[20]	[21]
Office Automation	5	2363	2363	2365	2363	2363	2363	2363	3547	2363	2363	2363	2363	3946	2363	2363
Picture in Picture (PiP)	8	640	640	896	640	640	640	640	896	640	640	640	640	896	640	768
Moving Pictures Expert Group (MPEG-4)	12	3501	3501	4386	3501	3529	3533*	3533	4386	3570	3590	3569*	3533	6537	3632	3972
Multi Window Displayer (MWD)	12	1120	1120	1408	1120	1216	1280*	1280	1792	1312	1472	1536*	1344	2144	1472	1696
H.263 Encoder	12	23.04	23.04	33.08	23.04	23.04	23.04	23.04	34.98	23.05	23.09	57.97*	23.05	39.40	23.18	23.61
MP3Encoder (MPEG-1 ALIII)	13	16.52	16.54	22.08	16.54	17.02	17.02*	17.02	25.88	17.05	17.06	20.95*	17.09	25.90	18.06	18.68
H.263 Decoder	14	19.63	19.66	35.44	19.63	19.66	19.82*	19.89	47.64	20.18	20.95	24.27*	20.17	31.57	20.52	20.49
Video Object Plane Decoder (VOPD)	16	3993	4009	6398	3993	4025	4009*	4073	8076	4357	4496	8224*	4105	8785	4395	5046
TGFF - G1	32	NA					11695*	10000	11256	11763	13206	NR	10882	12357	13152	15256
TGFF - G2	64	NA					52896*	50421	51425	48103	59117	NR	50807	64621	55617	73996
TGFF - G3	128	NA					NA					NR	97056	115276	98307	133705

NA represents that the application cannot be mapped onto the topology; NR represents that ILP has not read the file; (*) represents that ILP has not run completely



(a) 5x5 (b) 9x9 (c) 12x12

FIGURE 11. Normalized communication cost comparison of PSO algorithm with SA algorithm [20], GA [21], and FSCM approach [10] by scaling the Torus topology size from 5 × 5 to 12 × 12.

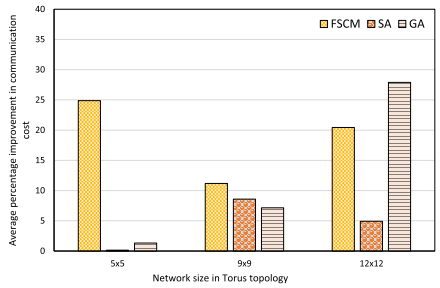


FIGURE 12. Average percentage improvements in communication costs obtained by scaling the network size using our approach over the approaches FSCM [10], SA [20], and GA [21].

respectively. On an average, by scaling the network size, our approach has shown an improvement of 18.83%, 4.55%, and 12.12% in communication cost over the approaches FSCM [10], SA [20], and GA [21], respectively. This shows our approach's efficiency in mapping the cores (including spare core) onto the different network sizes of Torus topology.

2) COMPARISON OF COMMUNICATION COST BETWEEN OUR APPROACH (PSO) AND THE APPROACHES FSCM [10], SA [20], GA [21] BY SCALING THE PERCENTAGE OF ROUTER FAULTS IN 9 × 9 TORUS TOPOLOGY

In the previous experiment, we have seen the efficacy of the approaches by increasing the network size. This experiment demonstrates our approach's efficiency while performing fault-tolerant application mapping onto 9 × 9 Torus topology by changing the percentage of router faults. Please note that the same kind of experiments can be done with any size of the network. This kind of study helps to analyze the efficiency of the approaches PSO, FSCM, SA, and GA in finding the best possible router locations for mapping the cores (including spare core) onto the network. The term *percentage of router faults* is defined as the number of unavailable routers in the *mxn* for mapping the cores of an application. For example, if we consider 15% router faults in 9 × 9 Torus topology, then out of 81 routers, 12 of them are unavailable for mapping [15]. As the percentage of router faults is increasing, the selection of routers for mapping an application plays a key role in the quality of the solution. Further, the quality of a mapping solution depends on the communication cost.

Table 5 shows the communication cost results comparison between our approaches (ILP and PSO) and the approaches FSCM [10], SA [20], and GA [21] by varying the percentage of router faults in 9 × 9 network. The term NA represents that the concerned application is not suitable for mapping because the number of routers required for mapping is less than the cores. As we can observe from Table 5, with an increase in the percentage of router faults, our approach could place the cores (including spare core) efficiently. The communication cost obtained using our approach is minimum when compared with the approaches FSCM [10], SA [20], and GA [21]. We have normalized the communication cost values to our approach, and the results for different application benchmarks are shown in Fig. 13. In Fig. 13, the X-axis represents the application benchmarks, and Y-axis represents the communication cost normalized to our approach PSO algorithm. As mentioned earlier, compared to our approach and the approaches SA [20] and GA [21], the FSCM technique has fixed the NAD region to map the cores (including spare ones) onto the routers. This resulted in high communication costs because the number of hops required to communicate between the healthy cores and the spare core is high.

TABLE 5. Communication cost comparison between our approaches (ILP/PSO) and the approaches FSCM [10], SA [20], GA [21], by scaling the percentage of router faults.

Application	No. of cores	15%				35%				50%			
		Our approach	FSCM [10]	SA [20]	GA [21]	Our approach	FSCM [10]	SA [20]	GA [21]	Our approach	FSCM [10]	SA [20]	GA [21]
Office Automation	5	2363	3946	2363	2363	2363	3946	2363	2365	2363	3946	2363	2365
PiP	8	640	896	640	704	640	896	640	832	640	896	704	960
MPEG-4	12	3674	8201	3802	3917	3675	8887	3906	4149	4570	9507	4580	5166
MWD	12	1472	2880	1440	1600	1632	2976	1728	1824	1760	3148	1824	1920
H.263Encoder	12	25.49	46.40	39.72	23.10	25.52	53.86	44.32	25.59	28.45	58.98	46.12	31.93
MP3Encoder	13	18.04	40.34	25.22	18.14	18.71	40.58	25.88	20.72	18.71	44.64	29.90	22.03
H.263Decoder	14	20.36	47.59	22.26	21.48	21.12	48.70	23.36	20.79	21.12	51.90	28.03	22.83
VOPD	16	4451	7912	5099	4999	4521	8773	5547	5030	4661	9151	6184	6484
TGFF-G1	32	9593	10975	13220	14758	9813	14352	14066	17582	10093	14520	14840	18951
TGFF-G2	64	47316	50004	85312	70691	NA							

NA represents that the application cannot be mapped onto Torus topology

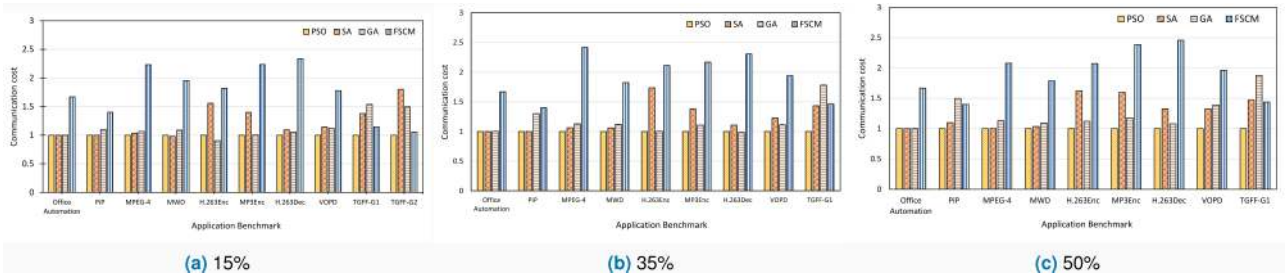


FIGURE 13. Normalized communication cost comparison of PSO algorithm with SA algorithm [20], GA [21], and FSCM approach [10] by scaling the percentage of router faults in Torus topology size of 9 × 9.

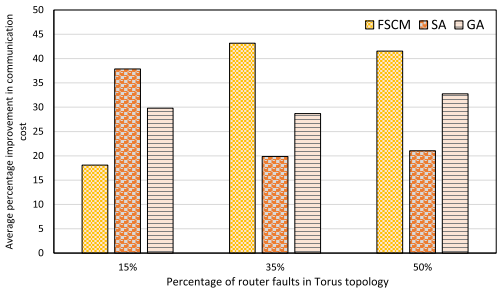


FIGURE 14. Average percentage improvements in communication costs obtained by scaling the percentage of router faults using our approach over the approaches FSCM [10], SA [20], and GA [21].

From Fig. 13, it can be observed that as the number of cores in an application and the percentage of router faults are increased, our approach PSO has shown significant improvements over the approaches SA [20], GA [21], and FSCM [10]. Fig. 13(a), 13(b), and 13(c) show the normalized communication cost results for the applications mapped onto 9 × 9 Torus topology with 15%, 35%, and 50% router faults, respectively. It can be observed that from Fig. 13, our approach and the approaches SA [20] and GA [21] have performed better compared to the approach FSCM [10]. This is due to the utilization of the entire search space by the approaches PSO, SA, and GA over the approach FSCM, to map the cores (including the spare core) onto the Torus network.

With increase in the router faults, the search space for the algorithms PSO, SA and GA is limited. Within the limited search space, the approaches SA and GA could not perform well because of the poor convergence in obtaining the minimum communication cost for GA [21] and the premature convergence in SA algorithm [20]. Fig. 14 shows the average percentage improvement in communication cost obtained using our approach over the approaches FSCM [10], SA [20], and GA [21] by scaling the router faults, in 9×9 network from 15% to 35% and 50%, respectively. On an average, by scaling the router faults our approach has shown an improvement of 34.27%, 26.26%, and 30.41% in communication cost over the approaches FSCM [10], SA [20], and GA [21], respectively. This improvements are due to the efficient placement of spare core in the network over the FSCM [10] and better convergence of the PSO algorithm over the SA algorithm [20] and GA [21]. Compared to the scaling in network size, the average percentage improvements in communication cost achieved using our approach is high while we scale the router fault-percentage in the network. It is due to efficiency of the PSO algorithm that has the capability to explore entire search space of the solutions.

3) COMPARISON OF DYNAMIC SIMULATION RESULTS BETWEEN OUR APPROACH AND THE APPROACHES FSCM [10], SA [20], GA [21]

This experiment demonstrates the dynamic behavior of the application benchmarks simulated using the cycle-accurate NoC simulator [28]. The inputs to the simulator are mapping information (obtained for different applications using our approach and the approaches FSCM [10], SA [20], GA [21]), routing algorithm, application traffic, simulation, and saturation time. The outputs are average network latency (in clock cycles), throughput (in flits/cycle/core), and router power consumption (in mW). For the router power consumption

TABLE 6. Average network latency, throughput, and router power consumption comparison between our approaches ILP, PSO and the approaches FSCM [10], SA [20], GA [21] in 9 × 9 Torus topology.

Application	No. of cores	Average Network Latency (in clock cycles)				Throughput (in flits/cycle/core)				Router Power Consumption (in mW)			
		Our approach	FSCM [10]	SA [20]	GA [21]	Our approach	FSCM [10]	SA [20]	GA [21]	Our approach	FSCM [10]	SA [20]	GA [21]
Office Automation	5	72.00	74.00	74.00	74.00	0.0234	0.0223	0.0182	0.0165	211.89	214.37	211.88	211.90
PiP	8	77.35	79.77	77.38	80.22	0.0034	0.0031	0.0032	0.0023	213.17	214.29	213.64	213.96
MPEG-4	12	74.92	78.01	75.37	74.26	0.0123	0.0101	0.0137	0.0097	214.36	214.42	214.79	214.51
MWD	12	78.41	92.02	77.01	77.92	0.0039	0.0037	0.0032	0.0028	214.30	214.32	214.35	214.27
H.263Encoder	12	74.41	76.64	74.43	75.03	0.0015	0.0013	0.0013	0.0016	214.27	214.28	214.98	215.11
MP3Encoder	13	77.07	80.68	74.96	76.42	0.0404	0.0371	0.0495	0.0453	214.66	214.92	214.82	214.84
H.263Decoder	14	76.33	82.40	74.32	75.03	0.0510	0.0439	0.0484	0.0431	214.96	215.03	214.25	214.12
VOPD	16	74.99	80.48	75.09	75.50	0.0097	0.0088	0.0078	0.0093	214.40	214.47	214.87	214.91
TGFF-G1	32	75.81	77.46	76.87	76.21	0.0096	0.0093	0.0095	0.0091	214.58	214.61	215.14	215.78
TGFF-G2	64	79.18	80.36	79.89	81.28	0.0171	0.0153	0.0165	0.0169	215.61	215.67	215.96	215.21

NA represents that the application cannot be mapped onto Torus topology

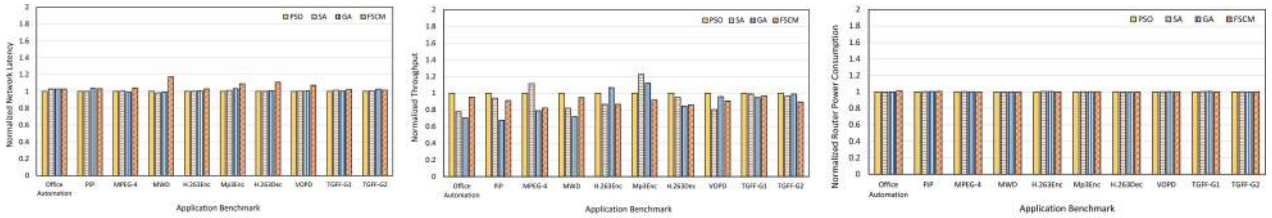


FIGURE 15. Normalized network latency, throughput, and router power consumption comparison of PSO algorithm with SA algorithm [20], GA [21], and FSCM approach [10] by performing dynamic simulations.

calculations, we have used ORION 2.0 tool [30]. Table 6 shows the comparison of dynamic simulation results between our approach and the approaches FSCM [10], SA [20], GA [21] for 9 × 9 Torus topology. In Table 6, first column represents the application type, and the second column represents the number of cores. Columns three to six, seven to ten, eleven to fourteen represent the network latency, throughput, router power consumption results obtained using our approach and the approaches FSCM [10], SA [20], GA [21], respectively. Fig. 15 shows the results of the approaches FSCM [10], SA [20], GA [21] normalized to the results obtained using our approach. In Fig. 15, X-axis represents application benchmarks shown in Table 6 and Y-axis represents normalized values of dynamic simulation parameters namely network latency (in Fig. 15(a)), throughput (in Fig. 15(b)), and router power consumption (in Fig. 15(c)). As it can be seen from Fig. 15(a), the network latency for the approach FSCM [10] is comparatively higher than that of our approach PSO algorithm, SA algorithm [20], and GA [21]. From Fig. 15(b), it is evident that the throughput for the approach FSCM [10] is less than our approach. This improvements using our approach could be possible because of the efficient placement of spare core in the network. However, for the router power consumption there are little improvements using our approach over the approaches FSCM [10], SA [20] and GA [21]. This is due to the static power consumption and switching activity of the routers in the topology. In addition to these, the leakage power consumption of the routers will also add to the total power consumption of the network. Therefore, we could not see much improvements in the router power consumption.

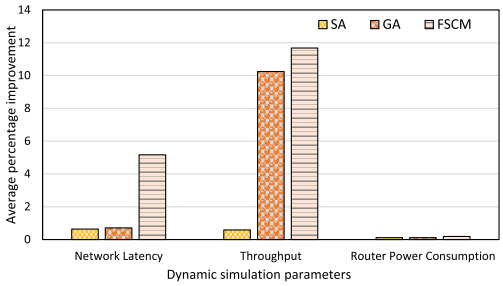


FIGURE 16. Average percentage improvements in dynamic simulations using our approach over the approaches FSCM [10], SA [20], and GA [21].

Fig. 16 shows the average percentage improvements in dynamic simulations performed for the fault-tolerant application mapping onto Torus topology using our approach and the approaches FSCM [10], SA [20], and GA [21]. In Fig. 16, the X-axis represents the simulation parameters namely network latency, throughput, and the router power consumption. The Y-axis represents the average percentage of improvements using our approach over the approaches FSCM [10], SA [20] and GA [21], respectively. On an average, our approach has achieved an improvement of 5.67%, 0.44 %, and 3.69% in terms of dynamic simulation parameters over the approaches FSCM [10], SA [20], and GA [21], respectively.

It has been observed from the dynamic simulation results that the mapping of spare core onto the topology affects the system performance. From the software implementation, i.e., static and dynamic results, it has been noticed that in the event of core failures, selection of mapping algorithm plays a major role in analyzing the performance of an application. This completes the software implementation of the

fault-tolerant application mapping onto Torus topology. However, the software implementation gives only an estimation of system performance. To realize the practical behavior of the mapping solutions obtained using our approach and the approaches proposed in the literature, hardware prototyping of the fault-tolerant application mapping solutions is highly essential. This study helps us to analyze the communication latency behavior of different application benchmarks. Next, we look into FPGA implementation of the proposed fault-tolerant application mapping techniques.

B. FPGA IMPLEMENTATION

This section presents the experimental results of the fault-tolerant application mapping solutions validated using an FPGA. From the software implementation results, we have observed that fault-tolerant application mapping performed using our approach has achieved significant improvements over the approaches FSCM, SA, and GA. As mentioned earlier, to analyze an application's practical behavior, an FPGA prototyping of fault-tolerant mapping of applications has been carried out. As part of FPGA prototyping, we have used the fault-injection model proposed in Algorithm-2 of Section IV.D. The faults are injected in the run-time of an application, and the communication latency or application run-time is calculated. Since communication latency plays a major role in time-critical applications, it is necessary to analyze the fault-tolerant mapping solutions by injecting the core faults and running them on an FPGA.

With the specifications mentioned above, 4×4 Torus topology is implemented on an FPGA and the fault-tolerant application mapping solutions obtained using our approach, FSCM [10], SA [20], and GA [21] are validated. The figure-of-merit for the validation process is the application run time on FPGA, which is dependent on the number of hops between the routers. Further, the number of hops required for the communication between the cores varies from one application mapping approach to another. In other words, it is dependent on the fault-tolerant application mapping algorithm. The step by step procedure for the validation of fault-tolerant application mapping solutions onto Torus topology is detailed below.

1) 4×4 TORUS TOPOLOGY IMPLEMENTATION ON FPGA BOARD

As part of the validation, firstly a communication platform has to be implemented on an FPGA. For the FPGA implementation of Torus topology, we have considered five port VC based wormhole router architecture [28]. Initially, single NoC router is synthesized with the addressing scheme and routing algorithm discussed in Section IV.C. Later, the 4×4 Torus topology is generated by connecting the routers as shown in Fig. 2. After the generation of 4×4 Torus topology, the design has been tested with different use cases to ensure the routing algorithm and proposed addressing scheme is working. This completes the 4×4 Torus topology implementation on an FPGA.

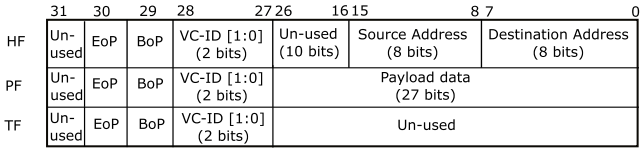


FIGURE 17. Packet format used for transferring the data.

We have scaled the Torus topology from 4×4 to 5×5, 9×9, 12×12 and attempted to implement them on the FPGA board. Table 7 shows the FPGA resource utilization and on-chip power (in Watts) calculated using Xilinx Xpower analyzer for the Torus topologies 4×4, 5×5, 9×9, and 12×12. The resource utilization shown in this Table consists of post-synthesis and post-implementation of the proposed fault-injection model for the 4×4, 5×5, and 12×12 Torus topology. The major reason to calculate the resource utilization independently for the post-synthesis and post-implementation is to know the number of Lookup Tables (LUTs) and Flip-Flops (FFs) required for the design. Due to the limited resources on KC705 FPGA, the Torus topologies 5×5, 9×9, and 12×12 have only been synthesized. They have not been implemented on the FPGA. Once the Torus topologies are synthesized on the FPGA, on-chip static and dynamic powers have been calculated using the Xilinx power analyzer tool. This gives the resource utilization and power profiles that can be used for the validation of the application-mapping solutions. Next, fault-tolerant application mapping solutions are validated on the KC705 FPGA board.

2) FAULT-TOLERANT APPLICATION MAPPING PROTOTYPING ON FPGA

The fault-tolerant application mapping solutions obtained using our approach and the approaches FSCM [10], SA [20], GA [21] are taken. Since the 4×4 Torus topology is implemented on an FPGA, the applications having the number of cores less than sixteen are mapped onto it. While mapping an application onto routers in the topology, the spare cores are also considered to provide fault-tolerance. Once the application is mapped onto the Torus topology, the real application traffic traces are given to the cores via Virtual Input Output (VIO) IP core. The application traffic traces are taken from the NoC simulator [28] and the number of flits (Header Flit (HF), Payload Flit (PF), Tailer Flit (TF)) are calculated.

Fig. 17 shows the packet format used for transferring the application data. To calculate the number of flits required per edge in the ACG is given by equation (11). From Fig. 17, it can be noticed that the number of data bits per PF is 27 bits. Therefore, for a given application Bandwidth (Bw), the number of flits required can be calculated using equation (11).

$$No.\ of\ Flits = (HF + \frac{Bw\ (in\ bits)}{27} + TF) \quad (11)$$

TABLE 7. Estimation of resource utilization and on-chip power utilization report of different Torus topology sizes synthesized and implemented on Kintex KC705 FPGA [31].

Topology size	Resource utilization				On-chip power (in Watts)					
	Post-synthesis		Post-implementation		Post-synthesis			Post-implementation		
	LUTs	FFs	LUTs	FFs	Dynamic	Static	Total	Dynamic	Static	Total
4x4	89350	108560	99944	128018	0.670	0.164	0.843	0.848	0.165	1.013
5x5	117347	143642	–	–	0.861	0.166	1.027	–	–	–
9x9	414153	502798	–	–	2.062	0.173	2.235	–	–	–
12x12	756105	914650	–	–	3.423	0.185	3.608	–	–	–

(–) represents that the design has not been implemented on an FPGA due to limited resources

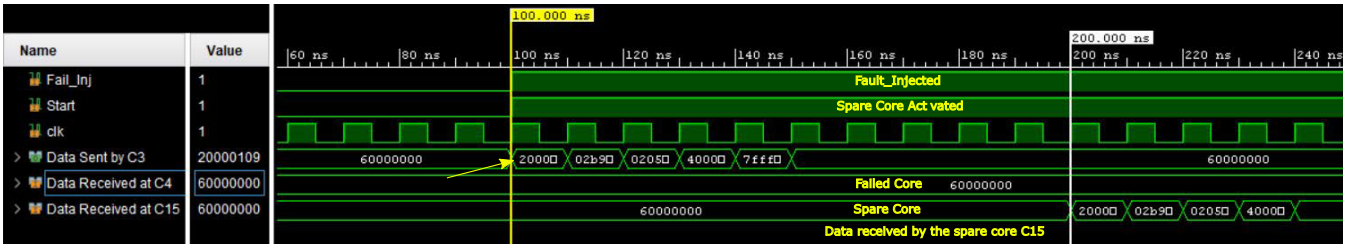


FIGURE 18. Screenshot of the post-implementation sending and receiving the data flits for an edge C3 - C4 in the MPEG-4 application ACG shown in Fig. 1.

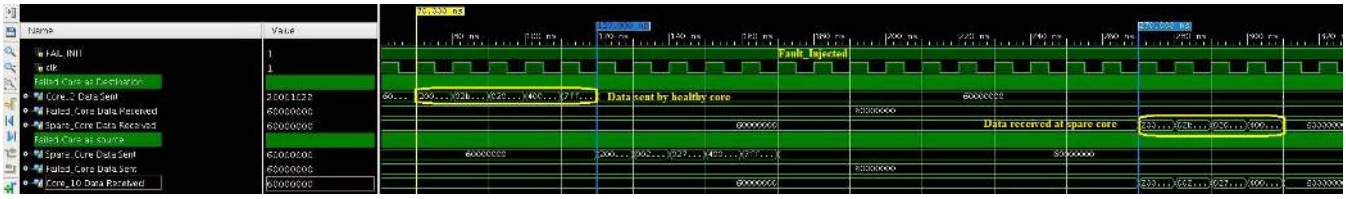


FIGURE 19. Screenshot of the post-implementation sending and receiving the data flits for an edge C2 - C4 (failed core) in the MPEG-4 application ACG shown in Fig. 1.

We have used the VIO IP core to send and receive the data flits from the source core to the destination core. Since the inputs, i.e., data flits to be given to core, are of 32 bits, the available on-board switches are not sufficient. Hence, the VIO IP core has been used to send and receive the data flits from source core to destination core. Fig. 18 and Fig. 19 show the screenshot of the post-implementation waveform showing the functional validation of spare core sending and receiving the data in the event of core-failure. The screenshot of the post-implementation of data flits sent and received for an edge C3-C4 in the MPEG-4 ACG shown in Fig. 1. There are six different signals whose transitions are captured in Fig. 18. The signals are *Fault_Inj*, *Start*, *clk*, *Data Sent by C3*, *Data Received at C4*, and *Data Received at C15*. The signals *Fault_Inj* and *Start* represent the status of fault injected and activation of spare core. They are connected to the DIP switch SW11 (as shown in Fig. 9) available on FPGA board [31]. The operation of these switches is shown in Table 3. The signal *clk* represents the FPGA clock working at 100 MHZ frequency. The signals *Data Sent by C3*, *Data Received at C4*, and *Data Received at C15* represents the transition of data flits sent and received from the cores. The core C3 has to send the data to core C4 in the fault-free case. In the event of failure, the data from C4 will be sent to spare core C15. The mapping information used for this experimentation is shown in Fig. 2(a) and Fig. 2(b).

From Fig. 18, it can be observed that if the signal *Fault_Inj* is high, then the core C4 is failed (most communicating). In this case, the data from core C3 will be sent to spare core C15 instead of C4. It is also evident from the screenshot shown in Fig. 18, the failed core C4 has not received any data from the core C3, instead, the data has been received by the spare core C15. This shows that the functionality is matching with the description shown in Table 3. Since the cores C3 and C15 are connected to routers R2 and R14 using the wrap around link; therefore, the hop count is one. For one-hop count, the time taken for sending or receiving the data flits (32 bits) is 100 ns. Similarly, we have experimented with other edges of the MPEG-4 ACG, and the results are shown in Table 8. In Table 8, columns one and two represent the edge of the MPEG-4 ACG. Columns three and four represent the bandwidth (in Megabits per second) and the number of flits (HF+PF+TF) per edge in the application. Columns five and six represent the number of hops required for an edge to communicate using the FSCM [10] and our approach mapping information (shown in Fig. 2(a) and 2(b)), respectively.

TABLE 8. Comparison of communication latency (application run time) between our approach and FSCM [10] for an MPEG-4 application running on an FPGA.

Edge in the ACG		Bandwidth (Mbps)	No. of flits/edge (HF+PF+TF)	No. of Hops		Time taken to deliver one flit (ns)		Time taken to deliver all flits per edge (seconds)	
Source Core	Destination Core			FSCM [10]	Our approach	FSCM [10]	Our approach	FSCM [10]	Our approach
0	4	190	7378870	2	1	150	100	1.11	0.74
1	4	1	38838	1	2	100	150	0.01	0.01
2	4	60	2330171	2	2	150	150	0.35	0.35
2	5	40	1553448	1	2	100	150	0.16	0.23
3	4	600	23301691	2	1	150	100	3.50	2.33
3	5	40	1553448	1	1	100	100	0.16	0.16
4	8	1	38838	1	2	100	150	0.01	0.01
4	9	910	35340897	3	1	200	100	7.07	3.53
4	10	32	1242759	3	1	200	100	0.25	0.12
6	7	250	9709039	1	1	100	100	0.97	0.97
6	9	670	26020221	1	1	100	100	2.60	2.60
6	10	173	6718656	2	1	150	100	1.01	0.67
6	11	500	19418076	1	1	100	100	1.94	1.94
Total time (in Seconds) taken to run MPEG-4 application on 4x4 Torus topology based NoC implemented on FPGA								19.11	13.66

TABLE 9. Comparison of application run time on an FPGA between our approach and the approaches FSCM [10], SA [20], and GA [21] for 4 × 4 Torus topology.

Application	Runtime on an FPGA (in Seconds)				Percentage of Improvements		
	Our approach	SA [20]	GA [21]	FSCM [10]	SA [20]	GA [21]	FSCM [10]
PiP	2.36	2.45	2.45	2.86	3.67	3.67	17.48
MPEG-4	13.66	14.23	14.73	19.11	4.00	7.26	28.52
MWD	4.35	4.83	5.22	5.72	9.93	16.66	23.95
H.263Enc	0.90	0.94	0.94	0.13	4.25	4.25	30.77
Mp3Enc	0.60	0.67	0.67	0.09	10.44	10.44	33.33
263Dec	0.80	0.80	0.82	1.12	0	2.43	28.57
Average	**3.77**	**3.98**	**4.13**	**4.67**	**5.38**	**7.45**	**27.10**

Columns seven and eight represent the time taken to send or receive one flit for the FSCM [10] and our approach, respectively. Columns nine and ten represent the total time taken to deliver the data flits (shown in column four) per an edge using the FSCM [10], and our approach, respectively. The total time took to run the application (all edges in the ACG) using the FSCM [10] and our approach, in the event of core faults, is 19.11 and 13.66 seconds, respectively. This is due to the mapping of the spare core along with other cores onto the topology by the FSCM [10] and our approach. It can be observed that the flexibility provided by our approach to map the spare core along with healthy cores of an application can result in less communication latency. Similarly, we have calculated the communication cost and application run time for the applications shown in the first column of Table 4. Please note that in Table 8, the detailed comparisons are shown between our approach and the FSCM [10] only. A similar method is used to calculate the application run-time on an FPGA for the approaches SA [20] and GA [21], and results are shown in Table 9. As it can be seen from Table 9, our approach has achieved significant improvements in terms of application runtime on an FPGA over the approaches FSCM [10], SA [20], and GA [21]. On an average there is an improvement of 5.38%, 7.45%, 27.10% using our approach over the approaches SA [20], GA [21], and FSCM [10], respectively. These improvements are due to the selection of possible router locations for the cores (including spare core) in 4 × 4 Torus topology using our approach over the approaches SA [20], GA [21], and FSCM [10].

VI. LIMITATIONS OF THE STUDY

The fault-tolerant application-mapping onto NoCs is well researched area and there exists many approaches in the literature [8]. The meta-heuristic algorithms PSO, SA, and GA are not the only remedies to solve the problem of fault-tolerant application-mapping. The possible limitation of the meta-heuristic PSO algorithm proposed in this paper is fine-tuning of parameters namely number of particles, number of generations, local and global swarm confidence values. Selection of these parameters require thorough investigation of the convergence of PSO algorithm in the context of fault-tolerant application mapping. There exists several meta-heuristic algorithms which advances the PSO, SA, and GA algorithms in terms of fine-tuning parameters. Though, the work addressed in this paper is limited to PSO, SA, and GA algorithms, there is still a room to explore other meta-heuristics presented in the literature.

VII. CONCLUSION

In this paper, we have presented fault-tolerant Torus topology based NoC design and validated the fault-tolerant solutions via FPGA implementation. The techniques ILP and PSO are proposed as the solutions for the fault-tolerant application mapping onto the Torus topology. An FPGA implementation of the fault-tolerant application mapping onto Torus topology has performed. For the FPGA implementation, fault-injection model, router addressing scheme, and routing algorithm has been proposed. The experimentations have been performed on the multimedia and synthetic application benchmarks. The behavior of fault-tolerant application mapping techniques is analysed using the software and hardware implementations. The results have shown significant improvements in terms of software and FPGA implementation. In software implementation, parameters such as communication cost, number of hops, network latency, throughput, and router power consumption calculated using our approach are superior when compared to the approaches reported in the literature. In FPGA implementation, communication latency or application run time obtained using our approach is superior when

compared with approaches reported in the literature. Future work includes extending the proposed techniques to consider multiple core failures by assuming the link, and router faults in the NoC topologies. A multi-application fault-tolerant mapping onto Torus topologies can also be considered as one of the possible extensions to this work.

REFERENCES

[1] W. J. Dally and B. Towles, "Route packets, net wires: On-chip inteconnectoin networks," in *Proc. 38th Conf. Design Autom. - DAC*, 2001, pp. 684–689.

[2] L. Benini and G. De Micheli, "Networks on chips: A new SoC paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, 2002.

[3] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, "HERMES: An infrastructure for low area overhead packet-switching networks on chip," *Integration*, vol. 38, no. 1, pp. 69–93, Oct. 2004.

[4] G. Moore, "Moores law," *Electron. Mag.*, vol. 38, no. 8, p. 114, 1965.

[5] M. Radetzki, C. Feng, X. Zhao, and A. Jantsch, "Methods for fault tolerance in networks-on-chip," *ACM Comput. Surv.*, vol. 46, no. 1, pp. 1–38, Oct. 2013, doi: 10.1145/2522968.2522976.

[6] P. K. Sahu and S. Chattopadhyay, "A survey on application mapping strategies for network-on-chip design," *J. Syst. Archit.*, vol. 59, no. 1, pp. 60–76, Jan. 2013.

[7] M. E. Gomez, J. Duato, J. Flich, P. Lopez, A. Robles, N. A. Nordbotten, O. Lysne, and T. Skeie, "An efficient fault-tolerant routing methodology for meshes and tori," *IEEE Comput. Archit. Lett.*, vol. 3, no. 1, p. 3, Jan. 2004.

[8] N. Kadri and M. Koudil, "A survey on fault-tolerant application mapping techniques for network-on-chip," *J. Syst. Archit.*, vol. 92, pp. 39–52, Jan. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1383762118301498

[9] N. K. R. Beechu, V. M. Harishchandra, and N. K. Y. Balachandra, "Hardware implementation of fault tolerance NoC core mapping," *Telecommun. Syst.*, vol. 68, no. 4, pp. 621–630, Aug. 2018.

[10] N. K. R. Beechu, V. M. Harishchandra, and N. K. Y. Balachandra, "An energy-efficient fault-aware core mapping in mesh-based network on chip systems," *J. Netw. Comput. Appl.*, vol. 105, pp. 79–87, Mar. 2018.

[11] R. P. Dick, D. L. Rhodes, and W. Wolf, "TGFF: Task graphs for free," in *Proc. 6th Int. Workshop Hardw./Softw. Codesign (CODES/CASHE)*, Seattle, WA, USA, 1998, pp. 97–101, doi: 10.1109/HSC.1998.66624.

[12] C.-L. Chou and R. Marculescu, "FARM: Fault-aware resource management in NoC-based multiprocessor platforms," in *Proc. Design, Autom. Test Eur.*, Mar. 2011, pp. 1–6.

[13] F. Khalili and H. R. Zarandi, "A fault-aware low-energy spare core allocation in networks-on-chip," in *Proc. NORCHIP*, Nov. 2012, pp. 1–4.

[14] 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.*, Dec. 2012, pp. 421–428.

[15] 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.*, vol. 15, no. 1, p. 23, 2019.

[16] B. N. Kumar and D. Sharma, "Communication energy constrained spare core on NoC," in *Proc. 11th Conf. Ph.D. Res. Microelectron. Electron. (PRIME)*, Jun. 2015, pp. 21–24.

[17] B. N. K. Reddy, M. H. Vasantha, and Y. B. N. Kumar, "A gracefully degrading and energy-efficient fault tolerant NoC using spare core," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2016, pp. 146–151.

[18] N. K. R. Beechu, V. M. Harishchandra, and N. K. Y. Balachandra, "High-performance and energy-efficient fault-tolerance core mapping in NoC," *Sustain. Comput., Informat. Syst.*, vol. 16, pp. 1–10, Dec. 2017.

[19] F. Khalili and H. R. Zarandi, "A fault-tolerant core mapping technique in networks-on-chip," *IET Comput. Digit. Techn.*, vol. 7, no. 6, pp. 238–245, Nov. 2013.

[20] J. M. Joseph, D. Ermel, L. Bamberg, A. García-Oritz, and T. Pionteck, "Application-specific SoC design using core mapping to 3D mesh NoCs with nonlinear area optimization and simulated annealing," *Technologies*, vol. 8, no. 1, p. 10, Jan. 2020.

[21] J. Fang, H. Zong, and H. Zhao, "DI_GA: A heuristic mapping algorithm for heterogeneous network-on-chip," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 490, Apr. 2019, Art. no. 042021.

[22] P. V. Bhanu, P. Kulkarni, S. J., 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)*, Jul. 2018, pp. 97–100.

[23] A. Alagarsamy, L. Gopalakrishnan, S. Mahilmaran, and S.-B. Ko, "A self-adaptive mapping approach for network on chip with low power consumption," *IEEE Access*, vol. 7, pp. 84066–84081, 2019.

[24] W. Amin, F. Hussain, S. Anjum, S. Khan, N. K. Baloch, Z. Nain, and S. W. Kim, "Performance evaluation of application mapping approaches for Network-on-Chip designs," *IEEE Access*, vol. 8, pp. 63607–63631, 2020.

[25] W. E. Donath, "Complexity theory and design automation," in *Proc. 17th Design Autom. Conf. Design Autom. DAC*, 1980, pp. 412–419.

[26] IBM Corporation, "V12. 1: Users manual for CPLEX," *Int. Bus. Mach. Corp.*, vol. 46, no. 53, p. 157, 2009.

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

[28] S. Kundu, J. Soumya, and S. Chattopadhyay, "Design and evaluation of mesh-of-tree based network-on-chip using virtual channel router," *Microprocessors Microsyst.*, vol. 36, no. 6, pp. 471–488, Aug. 2012.

[29] Z. U. Abideen and M. Rashid, "EFIC-ME: A fast emulation based fault injection control and monitoring enhancement," *IEEE Access*, vol. 8, pp. 207705–207716, 2020.

[30] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi, "ORION 2.0: A power-area simulator for interconnection networks," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 1, pp. 191–196, Jan. 2012.

[31] M. A. Finlayson, *KC705 Evaluation Board for the Kintex-7 FPGA Users Guide UG 810*. San Jose, CA, USA: Xilinx, 2019.

P. VEDA BHANU received the bachelor's degree in electronics and communication engineering from Jawaharlal Nehru Technological University at Hyderabad, Telangana, India, in 2015, and the master's degree in embedded systems from the National Institute of Electronics and Information Technology at Calicut, Kerala, India, in 2017. He is currently pursuing the Ph.D. degree with the Department of Electrical and Electronics Engineering, Birla Institute of Technology and Science (BITS), Pilani, Hyderabad-Campus, Telangana.

From 2016 to 2017, he was an Electronic Design Intern with Panacea Medical Technologies, Bengaluru, India. From 2017 to 2020, he was a Junior Research Fellow with the Department of EEE, BITS-Pilani, Hyderabad Campus working for the Department of Science and Technology, Government of India, sponsored project. His research interests include network-on-chip (NoC) design, FPGA-based system design, optimization of performance parameters in NoC based multi-processor system-on-chips (MPSoCs) design, and high-performance computing.

RAHUL GOVINDAN received the bachelor's degree in engineering from the Electrical and Electronics Department, BITS Pilani, Hyderabad Campus, in 2020. His research interests include digital design, computer architecture, VLSI design, and FPGA implementation.

PLAVA KATTAMURI is currently pursuing the bachelor's degree in engineering in electronics and communication engineering from BITS Pilani, Hyderabad Campus, India. Her research interests include the field of VLSI design, hardware security, and FPGA implementations.

J. SOUMYA received the bachelor's degree in electronics and communication engineering from Jawaharlal Nehru Technological University at Hyderabad, Telangana, India, in 2007, and the master's and Ph.D. degrees in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, India, in 2010 and 2015, respectively.

From 2011 to 2012, she was a Scientist 'SC' with Indian Space Research Organization (ISRO), Bengaluru, India. From 2014 to 2015, she was a faculty with the National Institute of Technology (NIT), Goa. India. Since 2015, she has been an Assistant Professor with the Department of EEE, BITS-Pilani, Hyderabad campus, Telangana, India. Her research interests include network-on-chip design, reconfigurable computing, fault-tolerant system design, and real-time systems. As a Principal Investigator, she has been implementing several funded projects from DST, Government of India, and has been collaborating with various research groups in India and abroad. Her research interests led to a credit of more than 25 publications in peer-reviewed journals and reputed international conferences held in India and abroad.

LINGA REDDY CENKERAMADDI (Senior Member, IEEE) received the master's degree in electrical engineering from the Indian Institute of Technology, New Delhi, India, in 2004, and the Ph.D. degree in electrical engineering from the Norwegian University of Science and Technology, Trondheim, Norway, in 2011. He worked for Texas Instruments in mixed signal circuit design before joining the Ph.D. program at NTNU. After finishing his Ph.D., he worked in radiation imaging for an atmosphere space interaction monitor (ASIM mission to International Space Station) with the University of Bergen, Norway, from 2010 to 2012. He is currently working as an Associate Professor with the University of Agder, Campus Grimstad, Norway. His main research interests include cyber-physical systems, autonomous systems, and wireless embedded systems.