IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 27, 2021, accepted May 10, 2021, date of publication May 21, 2021, date of current version June 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3082852

Fault-Tolerant Application-Specific Topology-Based NoC and Its Prototype on an FPGA

P. VEDA BHANU[1], RAHUL GOVINDAN[1], RAJAT KUMAR[1], VISHAL SINGH[1], J. SOUMYA[1], (Member, IEEE), AND LINGA REDDY CENKERAMADDI[2], (Senior Member, IEEE)

[1]Department of Electrical and Electronics Engineering, Birla Institute of Technology & 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 and Engineering Research Board (SERB), Government of India, under Project ECR/2016/001389, 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 Application-Specific Networks-on-Chips (ASNoCs) are suitable communication platforms for meeting current application requirements. Interconnection links are the primary components involved in communication between the cores of an ASNoC design. The integration density in ASNoC increases with continuous scaling down of the transistor size. Excessive integration density in ASNoC can result in the formation of thermal hotspots, which can cause a system to fail permanently. As a result, fault-tolerant techniques are required to address the permanent faults in interconnection links of an ASNoC design. By taking into account link faults in the topology, this paper introduces a fault-tolerant application-specific topology-based NoC design and its prototype on an FPGA. To place spare links in the ASNoC topology, a meta-heuristic algorithm based on Particle Swarm Optimization (PSO) is proposed. By taking link faults into account in ASNoC design, we also propose an application mapping heuristic and a table-based fault-tolerant routing algorithm. Experiments are carried out for a specific link and any link fault in fault-tolerant topologies generated by our approach and approaches reported in the literature. For the experimentation, we used the multi-media applications Picture-in-Picture (PiP), Moving Pictures Expert Group (MPEG) - 4, MP3Encoder, and Video Object Plane Decoder (VOPD). Experiments are run on software and hardware platforms. The static performance metric communication cost and the dynamic performance metrics network latency, throughput, and router power consumption are examined using software platform. In the hardware platform, the Field Programmable Gate Array (FPGA) is used to validate proposed fault-tolerant topologies and analyze performance metrics such as application runtime, resource utilization, and power consumption. The results are compared with the existing approaches, specifically Ring topology and its modified versions on both software and hardware platforms. The experimental results obtained from software and hardware platforms for a specific link and any link fault show significant improvements in performance metrics using our approach when compared with the related works in the literature.

INDEX TERMS Network-on-Chip, application-specific design, fault-tolerance, FPGA, communication latency, spare link, communication cost.

I. INTRODUCTION

The Network-on-Chip (NoC) interconnection paradigm has emerged as a promising solution for addressing communication issues in Multi-Processor Systems-on-Chips (MPSoCs) [1]. The key components of NoCs' communication infrastructure are cores, routers, and links. The NoC cores communicate via routers and links using, a packet-based switching technique [2]. Regular or irregular topologies can be used to design NoCs. In irregular topologies, the connections between routers via links vary depending on the application, whereas regular topologies have fixed

The associate editor coordinating the review of this manuscript and approving it for publication was Jenny Mahoney.

connection patterns. The selection of topology for the design of an NoC determines the majority of the design parameters such as network latency, throughput, and efficiency. In comparison to regular topologies, irregular topologies (also known as application-specific topologies), are more flexible and efficient in terms of communication cost, network latency, and power consumption [3]. As a result, application-specific topology-based NoCs (ASNoCs) are regarded as the preferred designs [3].

Thermal hotspots in ASNoCs can occur at any component level in the nano-scale era due to high integration density. This might lead to permanent failure of a system. As a result, system performance and reliability are degraded [4]. Therefore, fault-tolerant techniques are proposed as solutions to address the faults in ASNoC [5]. As previously stated, the connection pattern between routers and links is fixed in regular NoCs and follows a specific pattern when the topology is scaled. The connection pattern between routers and links in an irregular topology is not fixed and gets changed as the topology is scaled. Regular NoCs can address router and link failures simultaneously. This is because the designer is already aware of the connection pattern at this early stage. As a result, packets in the network can be rerouted using fault-tolerant routing algorithms. On the contrary, considering router and link faults in ASNoC topologies is extremely challenging. Unlike in regular topology, the router connection pattern is unknown in the early stages of application-specific topology. Because application-specific topology design differs from one application to the next, the same logic proposed for one topology may not be suitable for the other. There is non-uniformity in the application-specific topology design phase, which increases the designs complexity. The router and link failures must be considered as two distinct dimensions in the problem space when designing a fault-tolerant ASNoC. If the router in the application-specific topology fails, the corresponding links associated with the router are rendered useless. As a result, using redundant routers to reroute data packets using fault-tolerant methodologies is the best solution. If a link in the application-specific topology fails, the router associated with the failed link must decide whether to reroute packets through the alternate link. If there is no alternate link, communication is halted, and the desired response is not obtained. Similarly, if the design includes an alternate link, the packets are re-routed.

In comparison to router failure, link failure in application-specific topologies is more complex and difficult. Hence, this work addresses link faults and proposes a fault-tolerant ASNoC design based on spare links. For each application, fault-tolerant application-specific topologies are generated. This methodology includes topology design and a table-based fault-tolerant routing algorithm that re-routes packets in the topology. However, when designing a fault-tolerant application-specific topology, router failures must not be overlooked. Therefore, it has been regarded as an extension of this work. Because interconnection links play an important role in transferring application data among cores in ASNoC design, they must be prioritized when designing and implementing fault-tolerant ASNoC designs.

To date, the majority of fault-tolerant ASNoC designs have been evaluated using NoC simulators available in the literature [6]–[8]. These simulators, however, can only provide an estimate of parameters, which may not be accurate when the design is ported to an FPGA. Therefore, FPGA implementation is critical for analyzing and comprehending the practical behavior of fault-tolerant ASNoCs in terms of application runtime, resource utilization, and on-chip power dissipation. The authors of [5] implemented fault-tolerant ASNoCs for MP3Encoder applications on an FPGA while taking link faults into account. For the generation of fault-tolerant ASNoC, they used a modified version of Ring topology. Because it is a modified version of Ring topology, the number of hops between routers for a single-link fault in the topology will be high. This is due to the fact that the topology only has one alternate routing path for data packet transmission. The same group of authors in [5] proposed several changes to the Ring topology to allow for multiple routing paths. According to [9], Ring topology is the best application-specific topology for application cores with a size less than or equal to sixteen. However, the high average hop count between the routers in the Ring topology has a negative impact. Similarly, in one of their recent works [10], the average hop count in the Ring topology was improved by adding a link between the topology's routers. This new link will provide two alternate routing paths for data from the source core to the destination core. However, in the event of a single link failure, the fault-tolerant ASNoC designs generated by the approaches [5], [9], [10] resulted in an excess hop count. The main reason is that they used the Ring topology as the foundation for the fault-tolerant ASNoC design. As is well known, if the connection pattern between the routers remains constant, there will be a limited scope of improvement in performance parameters such as communication cost, network latency, and application runtime on an FPGA. Hence, there is a need to improve the performance parameters of the fault-tolerant ASNoC design by allowing users to customize the topology while it is being generated.

In light of this background, this paper proposes the design of fault-tolerant ASNoCs and their implementation on an FPGA. Picture-in-Picture (PiP), Moving Pictures Expert Group (MPEG) - 4, MP3Encoder, and Video Object Plane Decoder (VOPD) were used as multi-media application benchmarks. The following are the significant contributions of the paper.

1) A meta-heuristic Particle Swarm Optimization (PSO) based solution has been proposed to address the link faults in the ASNoC design.
2) A fault-tolerant routing algorithm has been proposed by considering the link faults in ASNoC design.
3) An application mapping heuristic algorithm is proposed and generated application-specific topology.

4) To analyze the hardware resource utilization and application runtime on an FPGA, an FPGA implementation of a fault-tolerant ASNoC design was performed.

Our approach has addressed the problem of link faults and generated fault-tolerant ASNoCs by considering (a) specific link (SL) fault in the topology, and (b) any link fault in the topology. The remainder of the paper is organized as follows. Section II details the literature survey. Section III discusses the faults and their effect on the interconnection links in the ASNoC topology. Section IV briefs the reliability analysis of the interconnection links. Section V explains the methodology used in our approach. Section VI discusses the experimental results. Section VII concludes the paper followed by the limitations of the work discussed in Section VIII.

II. LITERATURE SURVEY

This section summarizes the literature review on link faults in ASNoC design. The authors of [5] took a single link failure into account in the ASNoC design and implemented the fault-tolerant ASNoC on an FPGA. They dealt with the single link failure by implementing alternate routing options for the source and destination core pairs. The results show that there are few advantages over the ring topology-based ASNoC. The authors of [9] presented a two-step method for combining regular and irregular topologies to provide fault-tolerance to the application. In the first step, they used the Genetic Algorithm (GA) to generate the topology with the fewest number of routers and links possible. The authors then used the reconfiguration technique to switch between the routers in the topology in the second step. The authors of [10] described a fault-tolerant ASNoC created by modifying the ring topology with GA. They modeled the base topology as a Ring and reduced the distance between the two routers by incorporating spare links into the topology. In the modified Ring topology, they used the meta-heuristic GA to map the cores onto the routers. This resulted in improved performance when compared to their previously published work [9]. The authors of [11] addressed the link failures that can occur in the design of 2D application-specific NoCs. They created fault-tolerant topologies for the MP3Encoder application and compared them to ring topology. In the fault-tolerant ASNoC design, they proposed the Simulated Annealing (SA) technique for mapping cores onto routers. The main limitation of the works [5], [9]–[11] is that the authors regarded the Ring topology as a key platform for fault-tolerant ASNoC design. If the interconnection pattern between the routers remains constant, there is very little chance of improving the generation of fault-tolerant ASNoCs for various applications. This is due to the fixed routing path between the routers in the topology. In the event of a link failure, there are only a few possible routing paths for data packets to travel from the source router to the destination router.

The authors of [12] investigated permanent flaws in the Network Interface (NI) of an application-specific 3D NoC design. They took into account a single fault that could occur in the NI and used spare NIs as part of the fault-tolerance. The results show that the overhead in power consumption is minimal when compared to other approaches. However, the approach proposed by the authors [12] has taken into account NI failures and is limited to 3D NoC. The authors of [13] presented a fault-tolerant ASNoC for link faults based on Integer Linear Programming (ILP). They took into account link faults in the application-specific NoC and provided an alternate routing path. The authors' technique can be extended to any number of link faults, but there is an excessive overhead in terms of links, switches, and other components added to provide fault-tolerance to the system. Their focus, however, is not on the fault-tolerant application-specific NoC design's FPGA implementation. The authors of [14] presented a fault-tolerant application-specific NoC design that takes link faults into account. By proposing a greedy algorithm, they attempted to generate an application-specific topology with the lowest possible communication cost.

The authors of [15] presented a reliable custom topology for various applications. They used the Ant Lion Optimization (ALO) algorithm to generate a reliable custom topology with optimized power consumption and area. The proposed methodology was synthesized on an FPGA using a custom router design. The simulation results show that the desired output was obtained, and there is also a discussion of the experimental results. However, identifying the custom topology created with the ALO algorithm is extremely difficult. The reliability metric used in the [15] approach is also unknown. Though they have presented the custom and reliable topology for NoC design, there are very few details on how the custom topologies generated using the ALO algorithm are implemented. The authors of [16] presented the Discrete Antlion Trapping Mechanism (DTAM) to generate a custom topology for NoC design. They created the custom topology by taking into account three cores connected to one router in the topology. Experiments are carried out for various applications, and the results are compared to standard NoC topologies. The proposed DTAM technique outperformed previous approaches reported in the literature. However, determining the reliability of custom topologies generated with one or two cores using the DTAM technique is extremely difficult. While generating ASNoC topologies, most of the approaches proposed in the literature have focused less on the link faults. Furthermore, there are only a few approaches [5] that have performed an FPGA implementation of a fault-tolerant ASNoC. Therefore, this work is critical in addressing link failures while developing fault-tolerant ASNoC topologies and prototyping on an FPGA.

III. FAULTS AND THEIR EFFECT ON INTERCONNECTION LINKS IN NoCs

The number of transistors integrated on a single chip is increasing, according to the International Technology Roadmap for Semiconductors (ITRS) 2.0 report [17]. Because of the massive integration density of transistors and process variations in technology, the components of NoC, primarily links, are prone to faults [18]. According to [4],

TABLE 1. Different types of faults that can occur in NoC and their effect on the on-chip communication.

Fault-type	Source	Cause	Effect
Transient	High energy neutrons	Radio active impurities	Flipping of bits in memory cells and logic circuits
	Alpha particles strike in ICs		Single event upset and single event transition
Intermittent	Process variation, Electromigration, Negative Bias Temperature Instability (NBTI), Hot Carrier Injection (HCI)	Aging	Crystal defects, Metal contamination in dielectric material, Breakdown in dielectric
	Wear-out, Process Voltage Temperature		Continuous effects lead to permanent fault in system
	Electromagnetic Interference (EMI)	Cross-talk	Signal delay, glitches, damped voltage oscillations
	Electrostatic Discharge (ESD)		Destructive breakdown of a device
Permanent	Physical failure, Dynamic temperature variation	Temperature instability	Performance degradation

NoC faults are classified into three types: transient, intermittent, and permanent.

- The faults which occur randomly for one or several cycles are considered as *Transient faults*.
- The faults which repeatedly occur at the exact location or tend to occur in bursts are considered as *Intermittent faults*.
- The faults which can occur due to open or short circuit of the transistor and slow/delayed response of transistor or wire are considered as *Permanent faults*. This includes logic faults and delay faults, resulting in an incorrect logic value and delay of the response.

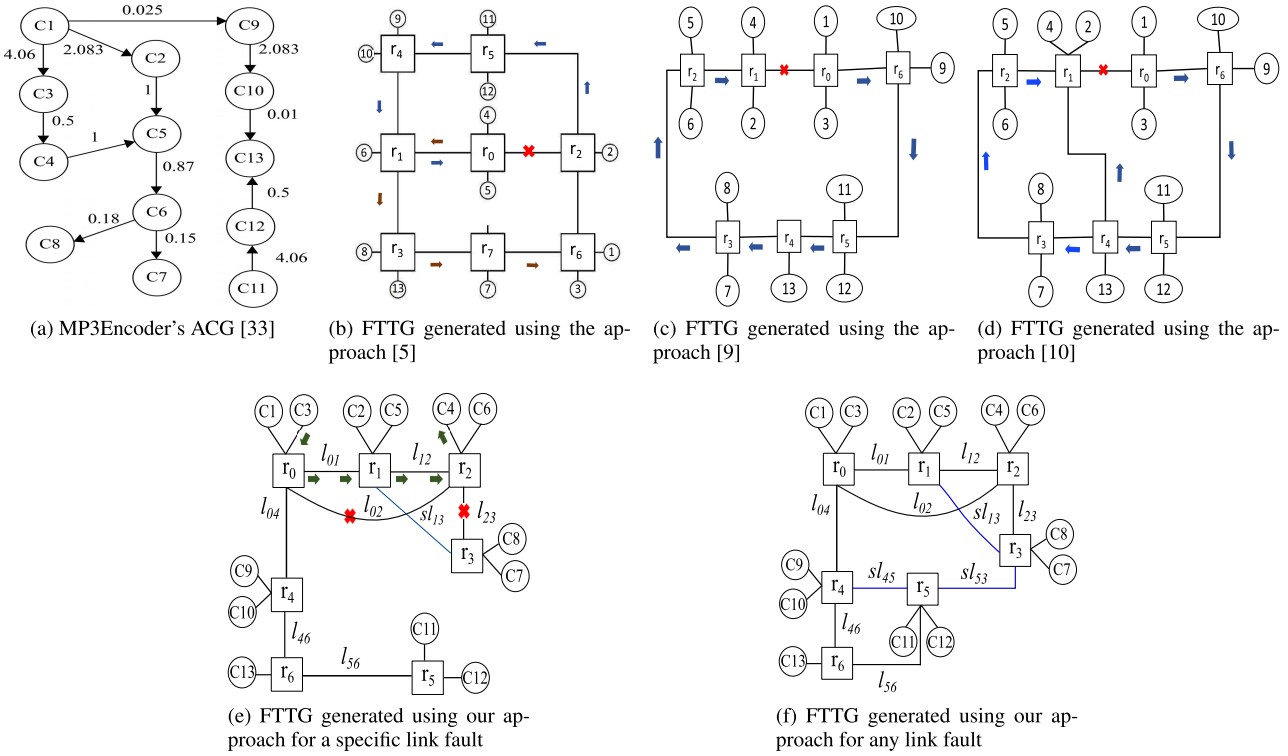
Table 1 shows different root causes of failures and their effect on NoC communication infrastructure. The type of fault is represented by the first column. The source and cause of the faults are represented by the second and third columns. The fourth column depicts the impact of the system's faults. Transient faults are primarily caused by α-particle striking in integrated circuits and high energy radiations emitted by impurities in the material [19], [20]. The most common effect of transient faults in the NoC is bit flipping. Intermittent and permanent faults severely degrade overall system performance. Intermittent faults are primarily caused by the aging of electronic components and crosstalk between wires in integrated circuits. Process variations are the primary cause of transient faults [21]–[23], electromigration (EM) [24], negative bias temperature instability (NBTI) [25], hot carrier injection (HCI) [26], wear out [27], process voltage temperature variations [28], electromagnetic interference (EMI) [29], and electrostatic discharge (ESD) [30]. The intermittent faults' bursty nature causes the system to fail permanently. Physical failure and temperature variations of the integrated circuits (ICs) will result in permanent faults in the NoC-based system.

The impact of these faults on NoC links will have a significant impact on overall system performance degradation. This is demonstrated with a real-time industrial application from the Avionics domain [31]. In any real-time system, decisions must be made quickly in order to address the issue within the task's deadline. If the required data is not received on time or is corrupted, the decision cannot be made in favor of the system's current state. There are 46 communicating tasks in the avionics application that perform these functionalities, with task periods ranging from 20 ms to 500 ms and communication volumes ranging from a few bytes to 10 MB. Assume that each task in the avionics application is assigned to a core, and that each core is linked to a router in an application-specific NoC. There is a communication delay between the cores if a link fails in the application-specific NoC. The data from the source core to the destination core will arrive after a time delay. This delay can have an impact on an application's overall behavior. In addition to the delay, the link fault causes data bits to be flipped from '1' to '0' or vice versa, resulting in an undesirable response. For example, during the task of weapon aiming, if the control signal required to trigger the weapon is '1,' but the control signal data bit is flipped from '1' to '0' due to a link fault in the application-specific NoC. This means that the aircraft will not fire against the enemy aircraft because the weapon will not be activated. If the weapon is not triggered within 20 ms (assuming that the task of weapon aiming is to be completed in less than 20 ms), there is a risk of disaster. The aircraft could be shot down by an enemy plane. This has the potential to endanger human life. As a result, in the event of a link failure, efficient fault-tolerant strategies that can mitigate faults in application-specific NoC-based MPSoCs are required to improve an application's performance and achieve the desired response.

IV. RELIABILITY ANALYSIS

The interconnection link plays a critical role in the communication between the cores in the application through the routers in the topology. To determine the failure of an interconnection link during the design phase, the reliability metric Mean Time To Failure (MTTF) must be calculated. In one of our previous works [32], we calculated the MTTF of interconnection links for various application-specific topologies. A similar technique was used to determine the MTTF of interconnection links for various application-specific topologies generated by our approach and approach [5]. As mentioned in the previous section, link faults are primarily caused by the EM effect. As a result, in our previous work [32], the MTTF is calculated based on the EM effect using the Black's equation given below.

$$MTTF = \frac{X}{J^2}(e^{\frac{EA}{kT}})$$



(a) MP3Encoder's ACG [33]

(b) FTTG generated using the approach [5]

(c) FTTG generated using the approach [9]

(d) FTTG generated using the approach [10]

(e) FTTG generated using our approach for a specific link fault

(f) FTTG generated using our approach for any link fault

FIGURE 1. **FTTGs generated for MP3Encoder's ACG using the approaches [5], [9], [10], and our approach for a specific link and any link fault.**

where *'X'* is an area of the conductor, *'J'* is the current density, *'EA'* is the activation energy, *'k'* is the Boltzmann constant, and *'T'* is the temperature (ambient and joule heating). For the experiment, the link length and width are 1 mm and 1 μm, respectively. According to the approach [32], the MTTF of a link is determined by the communication traffic that passes through it. If a link in the topology has a high communication bandwidth, it has a lower MTTF value, and vice versa. As a result, the links with the highest communication bandwidth are prioritized, and fault-tolerance is provided by adding spare links to the topology.

V. METHODOLOGY

This section describes the methodology used for the FPGA implementation of a fault-tolerant ASNoC design. Prior to FPGA implementation, we generate fault-tolerant ASNoCs for various applications using our approach and the approaches [5], [9], [10] on a software platform. They have generated the Fault-Tolerant Topology Graphs (FTTGs) of an application by considering each link fault in the topology. Our method solves the problem by taking into account a specific and any link fault in the application-specific topologies of several applications. Link failures are detected during the design process using the MTTF analysis presented in Section IV. Based on the MTTF analysis, our approach generates fault-tolerant ASNoCs for links with a shorter time to failure. However, the predictions for a design may not come true. In such cases, our approach can generate a fault-tolerant ASNoC that mitigates any link fault in the topology. The software and FPGA implementation of the proposed fault-tolerant ASNoC design are presented in this paper.

A. DEFINITIONS

The *Application Core Graph (ACG)* is a communication graph that represents the application requirements. It includes the number of cores, edges, and communication requirements (in *Mega bits per second (Mbps)*) between the cores. The *Topology Graph (TG)* represents the connection between the routers and links required for a particular application. FTTG denotes the connection between routers and links (including spare links) which attempt to achieve the fault-tolerance for a specific application. Please note that the interconnection links used in our design are bi-directional and there are very limited chances of the deadlock occurrence. Fig. 1(a) shows the MP3Encoder's ACG (modified from [33]), Fig. 1(b), 1(c), and 1(d) show the FTTGs generated using the approaches [5], [9], [10]. Fig. 1(e) and 1(f) show the FTTGs generated using our method by considering the specific link and any link fault, respectively. Because the approaches in [5], [9], [10] generated the FTTG by connecting more than one core to one router, our methodology proposed in this paper generated FTTGs by considering a maximum of two cores per router for a fair comparison. Our method, on the other hand, can generate FTTGs by taking into account one or more than two cores per router. Figure 2 depicts

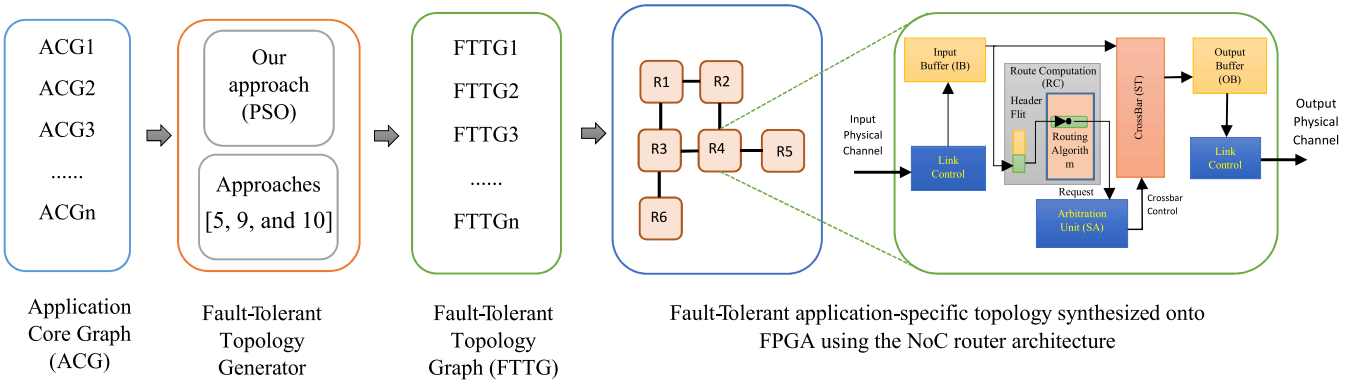


FIGURE 2. Design flow followed using our approach and the approaches [5], [9], [10] for the FPGA implementation of fault-tolerant ASNoC design.

the design flow used in our method and the methodology proposed in [5], [9], [10]. The inputs to our method and the approaches used in [5], [9], [10] are ACGs, and the output is an application's FPGA implementable FTTGs. To generate FTTGs for an application, we used the meta-heuristic PSO algorithm, the approaches [5], [9] used SA-based algorithm, and the approach [10] used GA-based technique. The performance parameters such as network latency, communication cost, router power consumption are calculated for the FTTGs generated using our approach and the approaches [5], [9], [10]. Consequently, the FTTGs generated by our approach, as well as the other approaches, are prototyped on an FPGA. The estimation of resources and communication latency, i.e. the time required for applications to run on an FPGA, are computed.

B. PSO FOR THE SPARE LINK PLACEMENT IN THE APPLICATION-SPECIFIC TOPOLOGY

1) INTRODUCTION TO PSO

The authors Eberhart and Kennedy proposed the PSO [34] optimization technique, which was inspired by the nature of bird flocking and fish schooling. The particle, according to this technique, is a solution flying in the problem space to find the best global solution. The fitness function determines the particle's quality. The fitness function is either minimized or maximized depending on the type of problem.

2) APPLICATION OF PSO FOR THE SPARE LINK PLACEMENT IN ASNoC

The PSO algorithm consists of four major steps, the first of which is to generate a population of particles at random. The second step is to define the fitness function, the third step is to generate the PSO, and the fourth step is to terminate the PSO. The *Particle Structure* must be defined before we can create a population of particles.

Step-1: The particle structure is a 2D array which consists of two distinct arrays. The first array elements represent the likelihood of selecting a router in the TG to add the spare link. The second array elements represent the likelihood of adding a spare link to the router chosen from the first array. This is demonstrated by the ACG of the MP3Encoder in Fig. 1. (a). Figure 3 depicts the particle structure for spare link placement. The particle has a length of 14, with the first seven elements used for router selection and the next seven elements used for link selection.

0.1	0.5	0.7	0.9	0.2	0.4	0.3
0	1	2	3	4	5	6

0.7	0.9	0.4	0.3	0.8	0.1	0.2
0	1	2	3	4	5	6

FIGURE 3. Particle structure.

We have assumed that the link between routers r_2 and r_3 is failed (Fig. 1(e)). However, any link, can be considered a failure in an application's FTTG. As shown in Fig. 1(e), there are seven routers available, numbered from r_0 to r_6, and the possible spare links that can be added for each router are pre-calculated, as shown below. Only one of these options will be added to the TG of the MP3Encoder application.

- $\mathbf{r}_0$: $\{sl_{03}\}$;
- $\mathbf{r}_1$: $\{sl_{13}\}$;
- $\mathbf{r}_2$: $\{\emptyset\}$;
- $\mathbf{r}_3$: $\{sl_{30}, sl_{31}, sl_{34}, sl_{35}, sl_{36}\}$;
- $\mathbf{r}_4$: $\{sl_{43}\}$;
- $\mathbf{r}_5$: $\{sl_{53}\}$;
- $\mathbf{r}_6$: $\{sl_{63}\}$;

For the failed link l_{23} (shown in Fig. 1(e)), from the first array of a particle (shown in Fig. 3), third array index has the highest probability, i.e., 0.9. Therefore, the third router (r_3) is selected to add a spare link. To select the spare link among different possibilities, we refer to the second array of the particle structure. In the second array, the third index has a probability of 0.3. The spare link whose selection range has the probability (obtained from second array) is considered for adding in TG. The selection range for each router can be calculated as follows. If a router has 'n' possible spare links to be added, then the selection range lies in between 0 - 1 with an interval of 1/n. As the router r_3 has five possible links $\{sl_{30}, sl_{31}, sl_{34}, sl_{35}, sl_{36}\}$, the selection range varies from 0 to 1 with an interval of 0.2 (1/5 = 0.2). If the probability

obtained from the second array of the particle structure is in between 0 - 0.2, then the spare link sl_{30} is selected. Similarly, if the probability is in between 0.2 - 0.4, 0.4 - 0.6, 0.6 - 0.8, 0.8 - 1, then the spare links sl_{31}, sl_{34}, sl_{35}, sl_{36} are selected, respectively. Since the third index in the second array of the particle has a probability of 0.3, the spare link sl_{31} is selected for adding in the TG. Please note that the spare links sl_{13} and sl_{31} are same. However, the particle length and the pre-calculated router, link dataset differ from one topology to the next. The empty link dataset (for $\mathbf{r}_2$) indicates that there is no possible spare link that can be added to the design to achieve fault-tolerance for the given failed link.

Step-2: The *Fitness Function*, which is the communication cost calculated for an application-specific topology, is defined as the second step in PSO. The communication cost (see equation 1) is defined as the product of the bandwidth between the cores and the number of hops required to complete the application data transfer.

$$Cost = \sum_{\forall\ edges} (BW_{edge} * Hop\ Count) \tag{1}$$

The PSO is designed to reduce the communication cost overhead caused by the addition of a spare link in the application-specific topology. Because the fitness function defines the quality of the particle, if the communication cost is the lowest, the value of that particle has achieved the best fitness.

Step-3: The third step is to generate new particle generations. It is possible to accomplish this by performing the swap operation on the particle population. The details of swap operators and new generation creation can be found in [35].

Step-4: The terminating condition must be defined in the fourth step of PSO. There are two methods for terminating the PSO. If the generation count for one independent run has expired in the first method, the PSO can be terminated. The PSO is terminated in the second method if the communication cost is observed to be constant over a predefined number of successive generations. The best communication cost results are reported following the termination of PSO. We considered the first method of terminating the PSO for our experiment.

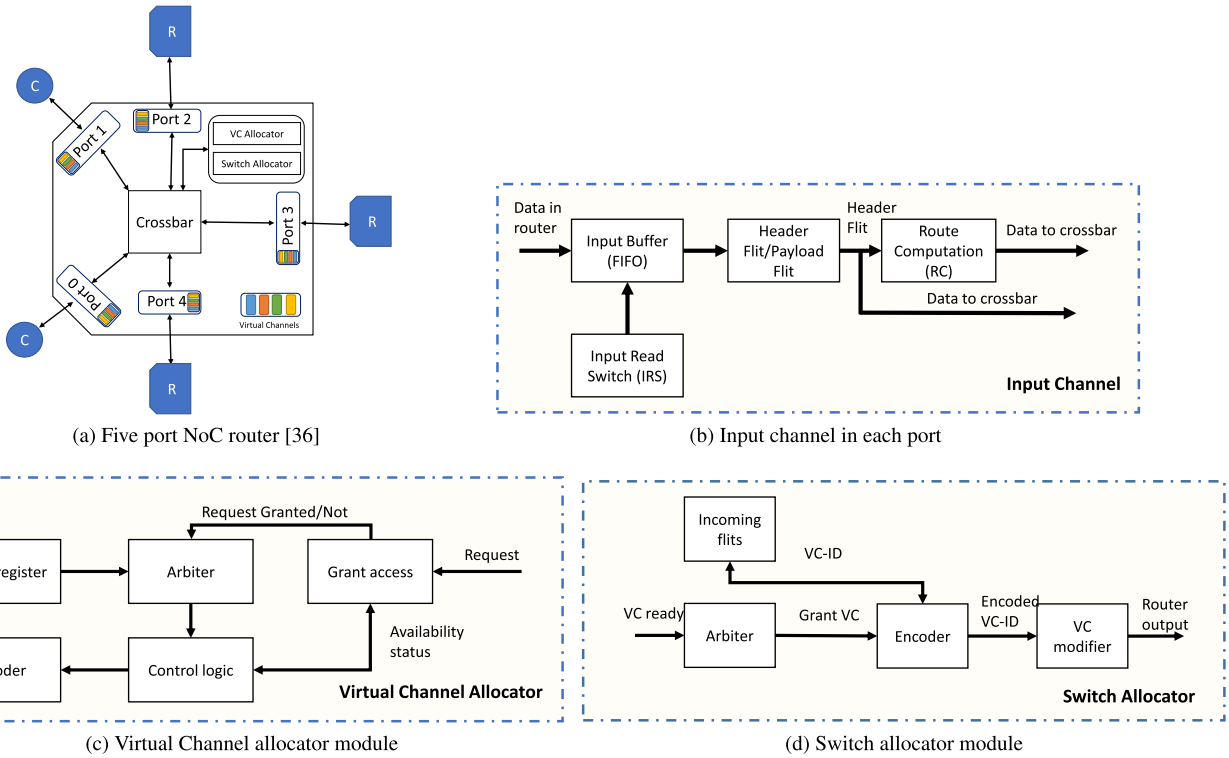
C. NoC ROUTER ARCHITECTURE

The NoC router architecture is taken from the work reported in the literature [36]. It has been developed using Verilog and implemented on Kintex KC705 evaluation board-based FPGA. We have used Xilinx Vivado 2016.2 tool for prototyping the NoC design onto FPGA. The detailed specifications of the NoC router can be seen from Table 2. However, any NoC router architecture can be synthesized and implemented on an FPGA for the fault-tolerant ASNoC design. Please note that the traffic patterns used for the experiments are application-specific. Because the application communication requirement is known, traffic patterns are generated in accordance with the application communication requirement. In comparison to application-specific traffic patterns, synthetic traffic patterns are not well suited for carrying out experiments for various applications. Synthetic traffic patterns are used to understand network behavior in the absence of applications running on it. In the current work, we focused on the application requirements and generated the custom topology accordingly. As a result, application-specific traffic is directly available and appropriate for the problem addressed in this work. Understanding network behavior under different traffic patterns, on the other hand, can be considered an extension of the work.

TABLE 2. NoC router specifications [36] used in this work.

NoC router synthesized onto FPGA	
Architecture	Wormhole router
Topology	Application-specific
Number of Virtual Channels (VCs)	4/3/2/1
Buffer type	FIFO buffer
Packet Length	4 Flits
Flit width	32 bits
FIFO Depth	6/8
Routing algorithm	Table-based
Flow control	Credit based
Arbiter type	Round Robin
Traffic Pattern	application-specific

Fig. 4 shows the NoC router architecture implemented on an FPGA. It is a five-port router. Out of five ports, two of them are dedicated to the cores, and three of them are for neighboring routers in the topology. Fig. 4(a) shows the block level representation of the NoC router. Each port in the router has four virtual channels. These virtual channels are used to avoid the occurrence of the deadlock and livelock while routing the data packets. Fig. 4(b) shows the internal block-level design of the channels in the input port. The data from the input links received at each port of the router are stored in the buffers and forwarded to the next module to identify the header and payload flit. Among the several data flits, the header flit is passed to the route computation unit of the input channel. Based on the destination address present in the header flit, the request will be sent to the corresponding output channel via a crossbar. Fig. 4(c) and 4(d) are the critical functional modules present in the output link of the router. The virtual channel allocator module and switch allocator module are the two basic logic blocks that can pass the data from the input channel to the output physical link. The virtual channel allocator module provides access to the input channel to pass the data flits to the output channel, while the switch allocator module establishes the path by providing access to the output links of the router. The fault-tolerant topologies are generated by considering a maximum of two cores connected to one router. The router port connections for the topologies generated using one core connected to one router and two cores connected to one router are shown in Fig. 5. With the configuration shown in Fig. 5, the fault-tolerant application-specific topologies are implemented on an FPGA. Thus the overall data flow in the router from the input channel of the link to the output physical link is shown



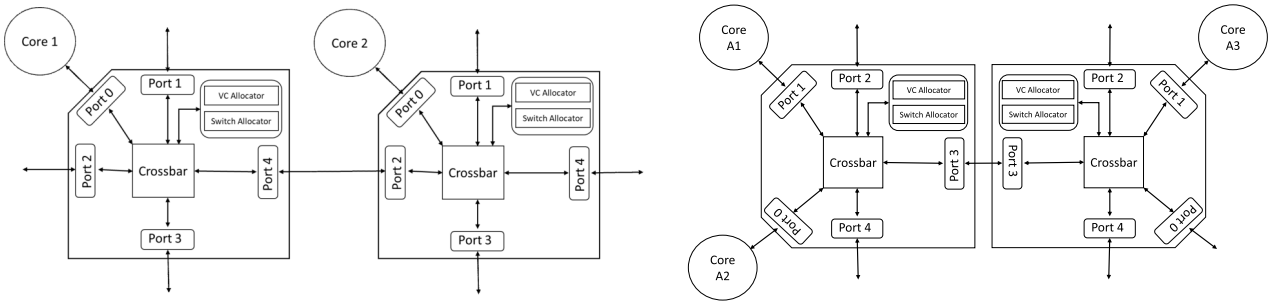
(a) Five port NoC router [36]

(b) Input channel in each port

(c) Virtual Channel allocator module

(d) Switch allocator module

FIGURE 4. NoC router architecture overview.



(a) One core connected to one router

(b) Two cores connected to one router

FIGURE 5. Router port connections used for generating fault-tolerant application-specific NoC.

in Fig. 6. This completes the detailed explanation of the NoC router architecture implemented on an FPGA.

D. FAULT-TOLERANT ROUTING ALGORITHM

Routing algorithms play a crucial role in deciding the network latency of a system. The routing algorithm resides in the RC unit of the NoC router. In the ASNoC design, the connection of routers is irregular when compared to the regular topologies like Mesh. Therefore, generalized routing algorithm might not fit for ASNoCs. Since the topology varies from one application to another, the best suitable routing algorithm for the ASNoCs is Table-based.

Algorithm 1 shows the table-based fault-tolerant routing algorithm. Inputs to the algorithm are HF, failed link (specific link or any link), router and link connection information, and

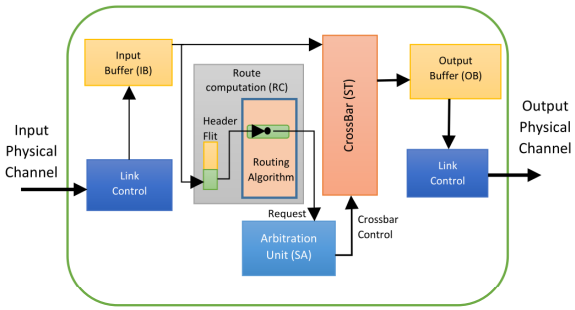


FIGURE 6. Overall data-flow in the NoC router (modified from [36]).

output is the fault-tolerant routing path established by using the spare link. Since our approach provides the FTTG of an application, the number of routers, links, and connections

Algorithm 1: Table-Based Fault-Tolerant Routing Algorithm

Input : Header Flit (HF), Failed link (specific link or any link in the FTTG), and router-link connection information

Output: Request the link that establishes fault-tolerant routing path

```
for all HFs do
    if current router address ≠ destination router address then
        if failed link is present then
            if alternate path exists then
                request the link present in the table-memory
            else
                request the spare link connected to the current router
        else
            request the link present in the table-memory
    else
        request the link connected to the core
```

between the routers is assumed to be known. Based on the FTTGs information, our approach used the shortest path algorithm, namely Dijkstra's [37], for finding the available path between the routers in the FTTG. This information is stored in the memory of the RC unit in the NoC router, which is analogues to the approaches [5], [9], [10]. The authors in [5] have also stored the complete routing path in the Path Table (PT), accordingly the links are requested.

Once the HF is received at every router, the source router address and the destination router address are taken from HF. The destination router address is compared with the current router address, and if they are not equal, then the HF has to pass from the current router to the next router. If there exists a failed link in the current router, then there are two possibilities in providing fault-tolerance. The first possibility is to check for the alternate paths from the routing table-memory. If there are any such alternate paths, then the corresponding link is requested. The second possibility is to request the spare link if there are no alternative paths to route the HF. The HF is traversed until the current router address, and destination router address becomes the same. If both destination and current router addresses are equal, then the link that is connected to the core is requested. This process continues for all the HFs received at the router. Once the HF establishes the fault-tolerant routing path, then the payload and tailer flit follows the path to deliver the application data. The proposed fault-tolerant routing algorithm is verified for the deadlock and live-lock avoidance using the channel dependency graph of the fault-tolerant application-specific topologies generated using our approach. In addition to this verification, the router architecture used in this work includes four virtual channels per link. The virtual channels help in avoiding the deadlock and live-lock while separating the application traffic [38]. For the fault-tolerant application-specific topology shown in Fig. 1(e), the routing path and the number of links involved in the path are shown in Table 3. The symbol (–) represents that the cores present in the edge are mapped to the same router. Therefore, the number of global links, i.e., the link between the routers involved in the path, is 0. From Table 3, it is evident that a maximum of two links are utilized for routing the data between the cores of an application. Fig. 7 shows the illustration of deadlock and live-lock occurrence with the help of a channel dependency graph (CDG) of the fault-tolerant topology of the MP3Encoder application shown in Fig. 1(e). According to [38], the nodes and arcs in the CDG of a topology represent the links present in the topology and the corresponding routing path of the topology. In the CDG shown in Fig. 7, there is only one arc between the nodes L_{12} and SL_{13}, which means that there exists a path between the routers via links in the topology. There are no arcs between the other nodes of the CDG because those nodes are utilized only once, and there is no immediate use of the links in the topology. Since there are no cyclic loops formed in the CDG, the proposed fault-tolerant topology and routing algorithm are deadlock and live-lock free.

TABLE 3. Routing path and the number of links involved in the path for the identification of deadlock and live-lock.

Edge in the MP3Encoder application	Routing path	Global links involved in the routing path
C1-C2	R0-R1	L01
C1-C9	R0-R4	L04
C1-C3	R0	–
C3-C4	R0-R2	L02
C2-C5	R1	–
C9-C10	R4	–
C4-C5	R2-R1	L12
C5-C6	R1-R2	L12
C6-C8	R2-R1-R3	L12-SL13
C6-C7	R2-R1-R3	L12-SL13
C10-C13	R4-R6	L46
C13-C12	R6-R5	L56
C12-C11	R5	–

– represents that the cores are mapped to single router and number of global links involved is 0

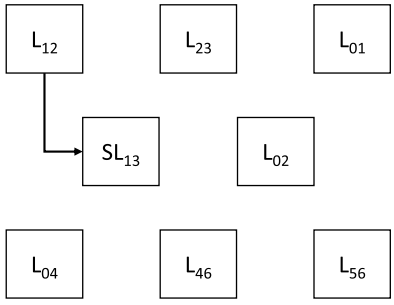


FIGURE 7. Channel Dependency Graph (CDG) for the fault-tolerant topology of MP3Encoder application shown in Fig. 1(c).

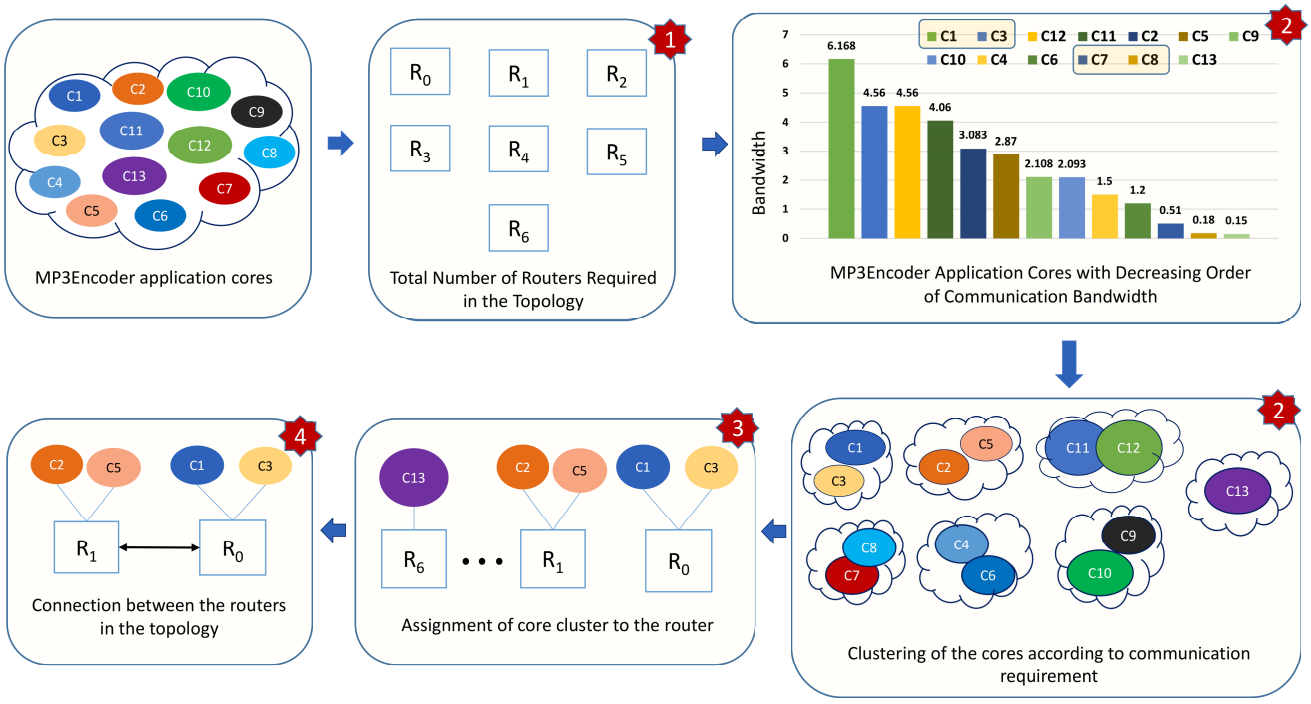


FIGURE 8. Mapping of MP3Encoder Application before generating the application-specific topology.

E. APPLICATION MAPPING HEURISTIC

This section presents the application mapping heuristic used for the generation of application-specific topology. Inputs to the application mapping heuristic are ACG and number of cores per one router. The output is the mapping of an application and generation of application-specific topology.

The step by step explanation of mapping heuristic is detailed below.

- **Step 1:** To map application onto the topology, number of routers that can accommodate the cores are calculated. They are calculated based on the number of cores connected per router. Since number of cores connected per router (x) is taken as input; accordingly, the routers are calculated.

$$No.\ of\ routers\ =\ |\frac{Total\ no.\ of\ cores}{x}|\ +\ 1 \quad (2)$$

  For example if the number of cores in ACG is 13, two cores per one router are connected, then the number of routers required for mapping are 7.
- **Step 2:** For all edges in ACG, arrange the edges in the decreasing order of the communication requirement (i.e., Bandwidth). Once the edges are arranged in the decreasing order, there might be cores which are common in two or more edges. Therefore, the common cores to more than two or more edges are merged into one. The cores in ACG are arranged in the decreasing order of their communication requirement. This arrangement of sequence is known as order of core sequence.
- **Step 3:** This is one of the key steps in the process of application mapping. Since the number of cores connected per router is an input, from the order of core sequence generated in Step 2, 'x' number of cores are taken sequentially and connected to each router.
  For example, if two cores are connected per router, then from the order of core sequence, first two cores are connected to first router, followed by other cores to remaining routers, and so on. This can be illustrated with the ACG shown in Fig. 1(a). The order of core sequence is {C1, C3, C2, C5, C4, C6, C7, C8, C9, C10, C11, C12, C13}, and the routers are $\{r_0, r_1, r_2, r_3, r_4, r_5, r_6\}$. Therefore, the mapping obtained using our approach is cores (C1, C3), (C2, C5),.., (C13) are connected to router $r_0, r_1,.., r_6$ (see Fig. 1(e)).
- **Step 4:** Once the cores in ACG are connected to each router, the router to router connection is established based on the core communication requirement. This completes the mapping technique and generates the application-specific topology.

Once the mapping of an application is done, the communication cost is calculated. This can be illustrated with the example of the MP3Encoder application shown in Fig. 1(a). Fig. 8 shows the different steps involved in generating the fault-free application-specific topology using the application mapping heuristic discussed above. The primary step involved in the mapping heuristic is to find the number of routers required for the communication in the topology. In the second step, the core clusters are identified based on

the descending order of the communication bandwidth of the application. In the MP3Encoder application, there are thirteen cores; these cores are arranged in the decreasing order of the core communication requirement with the neighboring cores in the application. In the third step, from the array of sorted cores, the first two cores are mapped onto one router, and the subsequent cores are mapped onto other routers in the topology. The cores C1 and C3 have the highest bandwidth shared with the other cores in the application. Therefore, the cores C1 and C3 are mapped onto router R_0. Similarly, cores C2 and C5 are having the highest bandwidth shared with the other cores in the application. Therefore, the cores C2 and C5 are mapped onto the router R_1. This process is repeated until all the cores of an application are mapped onto the routers selected for generating the topology.

VI. EXPERIMENTAL RESULTS

This section presents the experimental results obtained by generating the fault-tolerant application-specific topology for the multi-media applications such as PiP, MPEG-4, MP3Encoder and VOPD [14], [33], [39]. The experiments were performed on software platform followed by its implementation on an FPGA. The results are compared with the approaches [5], [9], [10] presented in the literature.

A. EXPERIMENTAL SETUP

For software implementation, PSO is coded in high-level language C++. The PSO is independently run for 30 times, and the best results are reported in this paper. The dynamic simulations are performed using the cycle-accurate System-C based NoC simulator [36]. For FPGA implementation, the fault-tolerant application-specific topology is implemented on an FPGA using Verilog Hardware Description Language (HDL). We have used the Kintex KC705 FPGA evaluation board [40] for the experiments.

B. SOFTWARE IMPLEMENTATION

The experiments are carried out for the applications and the communication costs are calculated for the FTTGs generated using our approach and the approaches [5], [9], [10]. The experimental results obtained by considering specific link fault and any link fault in the FTTG are reported under static results. The performance metric obtained from the static results for each application is communication cost. The experimental results obtained by running the System-C based cycle-accurate NoC simulator [36] are reported under dynamic results. The performance metric obtained from the dynamic results are network latency (in clock cycles), throughput (in flits/cycle/core), and router power consumption (in mW).

1) STATIC COMMUNICATION COST RESULTS

The communication cost is calculated using equation (1) after generating the FTTGs for different applications. For the MP3Encoder application, FTTGs generated using the approach [5], [9], [10] and our approach are shown in Fig. 1. We have considered a specific link fault and any link fault between the routers in FTTGs shown in Fig. 1. For a fair comparison between our approach and the approaches [5], [9], [10], we have considered the link fault between the routers (r_a and r_b) having the cores (c_a and c_b) communicating with high bandwidth. However, any link can be assumed to be failed in the topology graph; accordingly, the FTTGs can be generated. According to the mapping technique used by the approaches [5], [9], [10] and our approach (see Algorithm 2), one of the edges C6-C8 (shown in Fig. 1(a)) has been mapped onto the routers r_4, r_2 in approach [5], r_3, r_4 in approaches [9], [10], respectively, and r_2, r_3 in our approach. Therefore, the highest bandwidth link for the FTTGs generated using each approach will be different. Table 4 shows the comparison of communication cost results for the MP3Encoder FTTG generated by considering a specific link fault using the approaches [5], [9], [10] and generating using our approach. The first column represents the edge in MP3Encoder ACG and the second column represents the Bandwidth requirement between the cores in each edge. The columns three to ten represent the hop count and the communication cost for the FTTGs generated using the approaches [5], [9], [10] and our approach, respectively. In the event of a specific link fault, i.e., highest bandwidth link in the MP3Encoder FTTG, the overall communication cost obtained using the approaches [5], [9], [10] and our approach are 11.02, 18.36, 13.22, and 5.84, respectively.

The approaches [5], [9], [10] have generated FTTGs based on Ring topology and its modified versions. In [9], they have used Ring topology to generate the FTTGs for the applications having number of cores less than or equal to sixteen. The primary reason for considering the Ring topology is that the connection pattern between the routers is circular in shape. In the event of link failure in Ring topology, there exists an alternate path to route the data packets between the cores of an application. For the highest bandwidth link failure between the routers r_1 and r_2 in the Ring topology (shown in Fig. 1(c)), the number of hops between the cores connected to these routers is given by n-1, where n is the total number of routers in the topology. The total number of routers in the MP3Encoder FTTG is seven, therefore, the number of hops required to communicate between the cores C4 and C3 is six. For all the edges in the MP3Encoder ACG, in the event of link failure, the communication cost obtained using the approach [9] is 18.36.

The authors in [5] have modified the Ring topology such that there exist two alternate paths for a single link fault in the topology. They have added an extra router to provide two alternate routing paths for a single link failure. As it can be seen from Fig. 1(b), for the failed link between the routers r_1 and r_3, there exist two alternate paths for the cores C2 and C5 to communicate. The first path is from routers r_3 to r_1 through r_6, r_5, and r_2 constituting to four hops. The second path is from routers r_3 to r_1 through r_7, r_8, r_4, and r_2 constituting to five hops. Out of the two alternate paths, the path with minimum hop count is considered as

TABLE 4. Communication cost results comparison for a specific link fault between the approaches [5], [9], [10] and our approach (FTAS-SL) for the MP3Encoder FTTGs shown in Fig. 1.

Edge in MP3Encoder ACG	Bandwidth (Mbps)	Approach [5]		Approach [9]		Approach [10]		Our approach (FTAS-SL)	
		Hops	Communication cost	Hops	Communication cost	Hops	Communication cost	Hops	Communication cost
C4-C3	0.5	4	2	6	3	4	2	2	1
C5-C4	1	0	0	1	1	1	1	1	1
C6-C8	0.18	1	0.18	1	0.18	1	0.18	1	0.18
C6-C7	0.15	2	0.3	1	0.15	1	0.15	1	0.15
C5-C6	0.87	1	0.87	0	0	0	0	1	0.87
C2-C5	1	4	4	1	1	1	1	0	0
C1-C2	2.083	1	2.083	6	12.49	4	8.33	1	2.083
C1-C3	4.06	0	0	0	0	0	0	0	0
C1-C9	0.025	3	0.075	1	0.025	1	0.025	1	0.025
C9-C10	2.083	0	0	0	0	0	0	0	0
C10-C13	0.01	2	0.02	2	0.02	2	0.04	1	0.04
C12-C13	0.5	3	1.5	1	0.5	1	0.5	1	0.5
C12-C11	4.065	0	0	0	0	0	0	0	0
Overall communication cost			**11.02**		**18.36**		**13.22**		**5.84**

the solution. The hop count obtained is multiplied with bandwidth between the cores is computed as communication cost. For an edge C2-C5, the communication cost obtained is 2. Similarly, for other edges in MP3Encoder ACG, the overall communication cost is calculated according to equation (1). Therefore, in the event of link failure between the routers r_1 and r_3 in MP3Encoder FTTG, the overall communication cost obtained is 11.02. To minimize the communication cost in the fault-tolerant topology, the authors in [10] added an extra link between the routers in the Ring topology. The spare link provides two alternate routing paths to deliver the data from source to destination cores. In the event of link failure between the routers r_1 and r_2, compared to the Ring topology, the addition of spare link has reduced the hop count. In the MP3Encoder ACG, the cores in an edge C4-C3 communicate with four hops resulting in a communication cost of 2. This has improved the hop count compared to the basic Ring topology. With the addition of spare link to the Ring topology, the approach [10] has shown improvements over their previous work [9]. Similarly, in the event of link failure, the overall communication cost obtained using the approach [10] is 13.22.

Unlike the approaches [5], [9], [10], our approach has not considered the Ring topology to build FTTGs. Instead, they are generated by connecting the routers as per the communication requirement of an application. Our approach has considered the problem as topology generation rather than the mapping problem. The approaches [5], [9], [10] have used the meta-heuristics SA and GA to map the cores onto the routers of Ring topology. In contrast, our approach has used meta-heuristic PSO to find the best position to add the spare link in the topology. The approaches [5], [9], [10] have considered that out of x number of router ports, two ports can be used for mapping the cores onto routers and two ports for connecting the neighbouring routers in the topology. On contrary, out of x number of router ports, our approach utilizes x-2 number of ports to connect the routers via links and two ports for connecting the cores. In the event of link failure between the routers r_0 and r_2, there exists an alternate path to route the data from the source to destination core. If there is no alternate path available in the topology, our approach adds the spare link between the routers. The cores C4 and C3 communicate through the routers r_0 to r_2 via r_1 leading to two hops. For all the edges in MP3Encoder ACG our approach has resulted in a communication cost of 5.84.

Compared to the approaches [5], [9], [10], there is an average percentage improvement of 47%, 68.19%, and 55.82% in communication cost, respectively. This is due to the fact that these approaches have used the base topology as Ring. In the event of the link failure, our approach has resulted in less hop count. On the other hand, in the Ring topology and its modified variants the hop count in the event of link failure is high. This is because there is very limited flexibility to route the data in alternate paths having less hop count. In our approach, spare links are added to the topology to create alternate routing paths such that hop count can be minimized. This resulted in significant percentage of improvements using our approach over the counterparts. The similar trend can be seen in other applications namely PiP, MPEG-4, and VOPD. Table 5 to Table 7 represent the communication cost obtained for a specific link failure in the FTTGs of PiP, MPEG-4, and VOPD applications, respectively. In Table 5 to Table 7, the first column represents the edges in an application. The second column represents the bandwidth between the edges of an application. The columns three to ten represent the communication cost obtained for the highest bandwidth link failure in the topology using the approaches [5], [9], [10] and our approach. As mentioned earlier, due to the addition of the spare link to the FTTGs generated using our approach there are significant improvements in communication cost. Fig. 9 represents the percentage of improvements in communication cost obtained using our approach over the approaches [5], [9], [10] reported in the literature. In Fig. 9, the X-axis represents the multi-media application and Y-axis represents the percentage of improvements in communication cost. As it can be seen from Fig. 9, there is an

TABLE 5. Communication cost results comparison for a specific link fault (highest bandwidth link) between the approaches [5], [9], [10] and our approach (FTAS-SL) for the PiP application.

Edge in Picture-in-Picture ACG	Bandwidth (Mbps)	Approach [5]		Approach [9]		Approach [10]		Our approach (FTAS-SL)	
		Hops	Communication cost	Hops	Communication cost	Hops	Communication cost	Hops	Communication cost
C2-C1	128	0	0	0	0	0	0	0	0
C2-C3	64	3	192	3	192	2	128	2	128
C3-C4	64	0	0	0	0	0	0	0	0
C4-C7	64	2	128	2	128	1	64	1	64
C6-C7	64	1	64	1	64	1	64	1	64
C7-C8	64	0	0	0	0	0	0	0	0
C1-C5	64	2	128	2	128	2	128	1	64
C5-C6	64	0	0	0	0	0	0	0	0
Overall communication cost			**512**		**512**		**384**		**320**

TABLE 6. Communication cost results comparison for a specific link fault (highest bandwidth link) between the approaches [5], [9], [10] and our approach (FTAS-SL) for the MPEG-4 application.

Edge in Moving Pictures Expert Group - 4 (MPEG-4) ACG	Bandwidth (Mbps)	Approach [5]		Approach [9]		Approach [10]		Our approach (FTAS-SL)	
		Hops	Communication cost	Hops	Communication cost	Hops	Communication cost	Hops	Communication cost
C1-C5	190	3	570	2	380	2	380	1	190
C5-C9	0.5	1	0.5	1	0.5	1	0.5	1	0.5
C5-C2	0.5	1	0.5	1	0.5	1	0.5	1	0.5
C5-C3	60	3	180	3	180	3	180	2	120
C5-C4	600	1	600	5	3000	2	1200	2	1200
C5-C11	32	2	64	3	96	1	32	1	32
C5-C10	910	3	2730	0	0	0	0	0	0
C4-C6	40	3	120	3	120	3	120	2	80
C3-C6	40	1	40	1	40	1	40	0	40
C10-C7	670	0	0	5	3350	2	1340	2	1340
C11-C7	173	3	519	2	346	2	346	1	173
C7-C12	500	4	2000	1	500	1	500	1	500
C7-C8	250	3	750	1	250	1	250	1	250
Overall communication cost			**7574**		**8263**		**4269**		**3887**

TABLE 7. Communication cost results comparison for a specific link fault (highest bandwidth link) between the approaches [5], [9], [10] and our approach (FTAS-SL) for the VOPD application.

Edge in Video Object Plane Decoder (VOPD) ACG	Bandwidth (Mbps)	Approach [5]		Approach [9]		Approach [10]		Our approach (FTAS-SL)	
		Hops	Communication cost	Hops	Communication cost	Hops	Communication cost	Hops	Communication cost
C1-C2	70	2	140	0	0	0	0	0	0
C2-C3	362	0	0	1	362	1	362	1	362
C3-C4	362	1	362	0	0	0	0	0	0
C4-C5	362	0	0	1	362	1	362	1	362
C4-C16	49	2	98	1	49	1	49	1	49
C16-C5	27	2	54	2	54	2	54	2	54
C5-C6	357	1	357	0	0	0	0	0	0
C6-C7	353	0	0	1	353	1	353	1	353
C7-C8	300	4	1200	7	2100	4	1200	2	600
C8-C9	313	5	1565	7	2191	4	1232	2	626
C8-C10	500	0	0	0	0	0	0	0	0
C10-C9	407	5	2035	7	2849	4	1628	2	814
C12-C9	16	3	48	4	64	1	16	1	16
C12-C6	16	4	64	1	16	1	16	1	16
C12-C13	16	1	16	1	16	1	16	1	16
C11-C12	16	1	16	1	16	1	16	1	16
C15-C13	16	1	16	1	16	1	16	1	16
C15-C11	16	1	16	1	16	1	16	1	16
C13-C14	157	1	157	1	157	1	157	1	157
C14-C15	16	0	0	0	0	0	0	0	0
Overall communication cost			**6144**		**8621**		**5493**		**3473**

average percentage improvement of 54.58%, 44.16%, 29.54% in communication cost using our approach over the approaches [5], [9], and [10], respectively. As more number of alternate routing paths are provided in the topology, the percentage of improvements in communication cost using our approach is decreased. From this it is evident that the

TABLE 8. Communication cost results comparison for any link fault in the FTTG generated for PiP application using the approaches [5], [9], [10] and our approach (FTAS-AL).

Approach [5]		Approach [9]		Approach [10]		Our approach	
Fault-free communication cost: 384		Fault-free communication cost: 384		Fault-free communication cost: 320		Fault-free communication cost: 256	
Link fault between the routers in the FTTG	Communication cost	Link fault between the routers in the FTTG	Communication cost	Link fault between the routers in the FTTG	Communication cost	Link fault between the routers in the FTTG	Communication cost
r_0-r_1	448	r_0-r_1	512	r_0-r_1	384	r_0-r_1	320
r_0-r_2	384	r_0-r_2	384	r_0-r_2	320	r_0-r_2	256
r_1-r_4	448	r_2-r_3	512	r_2-r_3	320	r_2-r_3	320
r_1-r_2	448	r_3-r_1	384	r_3-r_1	320	r_3-r_1	256
r_3-r_4	512			r_1-r_2	384	r_1-r_2	320
r_2-r_4	512					r_0-r_3	320
Average	**458.66**		**448**		**345.60**		**298.66**

TABLE 9. Communication cost results comparison for any link fault in the FTTG generated for MPEG-4 application using the approaches [5], [9], [10] and our approach (FTAS-AL).

Approach [5]		Approach [9]		Approach [10]		Our approach	
Fault-free communication cost: 4754		Fault-free communication cost: 3183		Fault-free communication cost: 2999		Fault-free communication cost: 2789	
Link fault between the routers in the FTTG	Communication cost	Link fault between the routers in the FTTG	Communication cost	Link fault between the routers in the FTTG	Communication cost	Link fault between the routers in the FTTG	Communication cost
r_0-r_1	4754	r_0-r_1	8263	r_0-r_1	5539	r_0-r_1	4059
r_0-r_3	4756	r_1-r_2	6029	r_1-r_2	4499	r_1-r_2	2789
r_1-r_2	4754	r_2-r_3	3529	r_2-r_3	2999	r_0-r_2	2790
r_2-r_3	9394	r_3-r_4	3340	r_3-r_4	3079	r_0-r_3	3233
r_1-r_4	4914	r_4-r_5	3563	r_4-r_5	2999	r_3-r_4	2789
r_4-r_5	4754	r_5-r_0	3567	r_5-r_0	3001	r_1-r_4	4289
r_5-r_6	4818			r_0-r_3	3183	r_4-r_5	2789
r_3-r_6	8154					r_5-r_2	2789
Average	**5787.25**		**4715.16**		**3614.14**		**3190.87**

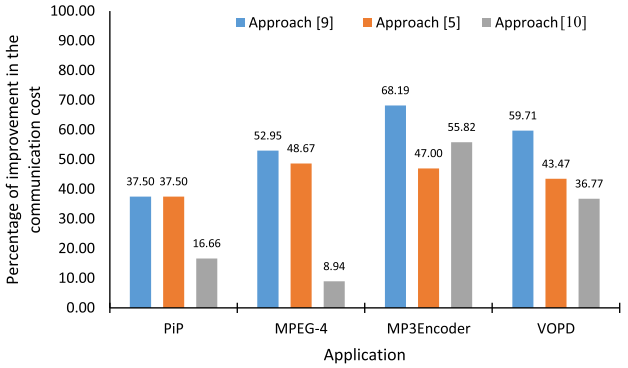


FIGURE 9. Percentage of improvements in communication cost using our approach over the approaches [5], [9], [10] for a specific link fault in the topology.

communication cost can be improved by adding the spare links to the topology.

If the link fault information is not known in design time, then we have proposed a solution that mitigates any link failure in the FTTGs of an application. Table 8 to Table 11 show the comparison of communication cost results between the approaches [5], [9], [10] and our approach by considering each link fault in the FTTGs of the applications PiP, MPEG-4, MP3Encoder, and VOPD. As it can be observed from Table 8 to Table 11, for the approaches [5], [9], [10] and our approach, the fault-free and fault-tolerant communication cost is calculated. In these Tables, under each approach there are two columns, one column represents the link fault between the routers in the topology and the other one represents the communication cost for the corresponding link failure. Table 10 shows the comparison of communication cost between our approach and the approaches [5], [9], [10] for the FTTG shown in Fig. 1. Fig. 1(f) shows the FTTG for any link fault in the topology generated for MP3Encoder application. The communication cost without any link fault in the FTTG of MP3Encoder application is 5.32. By considering each link fault in the fault-free topology of MP3Encoder application, the spare links are added or alternate routing path is selected. For the highest bandwidth link failure, the communication cost is high and for the lowest bandwidth link failure the communication cost is less. For example, if the link between the routers r_0 and r_1 is failed (see Fig. 1(f)), then the alternate routing path is selected. The link l_{01} is transferring highest bandwidth (2.083 Mbps) of application data between the cores C1 and C2 (see Fig. 1(a)) through routers r_0, r_2, and r_1 leading to two hops. In fault-free case, the hop count between the cores C1 and C2 to communicate is 1. Due to the link failure, the hop count has been increased to 2. Therefore, the overall communication cost for the link failure between the routers in MP3Encoder FTTG shown in Fig. 1(f) is 7.40. Similarly, for the highest bandwidth carrying link failure in other approaches [5], [9], [10] the overall communication cost is 15.39, 15.87, and 11.71, respectively. As discussed previously, the approaches [5], [9], [10] have routed the data in alternate path having high hop count. This led to high communication cost using the approaches [5], [9], [10]. On an average, for each link failure in the FTTGs generated using our approach (FTAS-AL) and the approaches [5], [9], [10], the overall communication cost for MP3Encoder application (shown in Table 10) are 5.98, 8.77, 9.06, and 6.96, respectively. The average percentage improvements in communication cost using our approach over the approaches [5], [9], [10] are 31.81%, 33.99%, and 14.08%, respectively. This improvements are achieved with efficient placement

TABLE 10. Communication cost results comparison for any link fault in the FTTG generated for MP3Encoder application using the approaches [5], [9], [10] and our approach (FTAS-AL).

Approach [5]		Approach [9]		Approach [10]		Our approach	
Fault-free communication cost: 7.03		Fault-free communication cost: 5.46		Fault-free communication cost: 5.46		Fault-free communication cost: 5.32	
Link fault between the routers in the FTTG	Communication cost	Link fault between the routers in the FTTG	Communication cost	Link fault between the routers in the FTTG	Communication cost	Link fault between the routers in the FTTG	Communication cost
r_0-r_1	10.50	r_0-r_1	15.46	r_0-r_1	9.46	r_0-r_1	7.40
r_0-r_2	8.02	r_1-r_2	7.11	r_1-r_2	6.12	r_0-r_2	5.82
r_1-r_3	8.59	r_2-r_3	5.46	r_2-r_3	5.46	r_1-r_2	7.18
r_1-r_4	7.56	r_3-r_4	8.26	r_3-r_4	5.46	r_1-r_3	5.65
r_2-r_5	7.05	r_4-r_5	5.76	r_4-r_5	6.47	r_0-r_4	5.37
r_2-r_6	15.39	r_5-r_6	5.54	r_5-r_6	5.47	r_4-r_6	5.33
r_3-r_7	7.32	r_6-r_0	15.87	r_6-r_0	5.53	r_4-r_5	5.32
r_6-r_7	7.03			r_0-r_3	11.71	r_5-r_6	5.82
r_4-r_5	7.55						
Average	**8.77**		**9.06**		**6.96**		**5.98**

TABLE 11. Communication cost results comparison for any link fault in the FTTG generated for VOPD application using the approaches [5], [9], [10] and our approach (FTAS-AL).

Approach [5]		Approach [9]		Approach [10]		Our approach	
Fault-free communication cost: 2664		Fault-free communication cost: 2555		Fault-free communication cost: 2539		Fault-free communication cost: 2539	
Link fault between the routers in the FTTG	Communication cost	Link fault between the routers in the FTTG	Communication cost	Link fault between the routers in the FTTG	Communication cost	Link fault between the routers in the FTTG	Communication cost
r_0-r_1	2792	r_0-r_1	8675	r_0-r_1	4378	r_0-r_1	3559
r_1-r_2	2664	r_1-r_2	2555	r_1-r_2	2539	r_1-r_2	2539
r_2-r_3	2718	r_2-r_3	3593	r_2-r_3	3058	r_2-r_3	3598
r_3-r_4	5658	r_3-r_4	2747	r_3-r_4	2635	r_3-r_4	2713
r_4-r_5	5504	r_4-r_5	2975	r_4-r_5	2749	r_4-r_5	2749
r_5-r_7	3264	r_5-r_6	5021	r_5-r_6	3772	r_5-r_6	2603
r_6-r_7	6144	r_6-r_7	4727	r_6-r_7	3615	r_6-r_7	2917
r_7-r_8	2664	r_7-r_0	4697	r_7-r_0	3610	r_7-r_0	2539
r_8-r_0	3420			r_0-r_4	2587	r_0-r_2	2892
						r_0-r_5	2571
Average	**3869.77**		**4373.75**		**3215.88**		**2868**

of spare link between the routers and selection of alternate routing path in the topology. The similar trend can be seen in other applications namely PiP in Table 8, MPEG-4 in Table 9, and VOPD in Table 11. As it can be observed from the Tables 8, 9, and 11, the communication cost for each link failure in the FTTG is calculated for the approaches [5], [9], [10] and our approach. In all the applications, our approach has shown significant improvements in communication cost over the approaches [5], [9], [10]. Fig. 10 shows the percentage of improvements in communication cost using our approach over the counterparts by considering each link fault in the topology. As it can be noted from the Fig. 10, the X-axis represents the applications (PiP, MPEG-4, MP3Encoder, VOPD), Y-axis represents the percentage of improvements in communication cost using our approach over the approaches [5], [9], [10]. The average percentage improvements in communication cost using our approach over the approaches [5], [9], [10] for any link fault in the FTTG is 34.36%, 33.52%, and 12.54%, respectively. This shows the efficiency of our approach in providing fault-tolerance to the design with less communication cost for a specific link failure and any link failure as well. This completes detailed analysis of fault-tolerant application-specific topology generation for the multi-media applications PiP, MPEG-4, MP3Encoder, and VOPD. Next, the dynamic simulation results are analysed for the topologies generated using our approach and other approaches by considering a specific link and any link fault in the FTTGs.

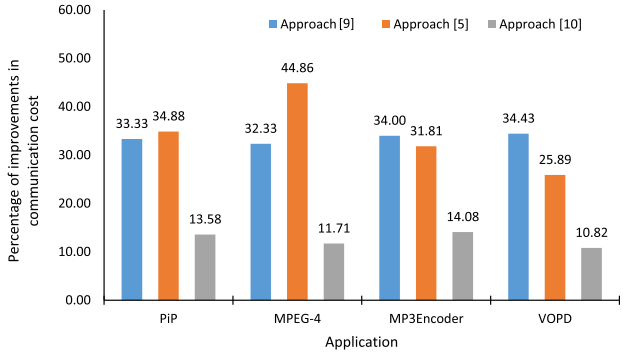


FIGURE 10. Percentage of improvements in communication cost using our approach over the approaches [5], [9], [10] for the FTTGs generated for any link fault.

2) DYNAMIC SIMULATION RESULTS

We have used System-C based cycle-accurate NoC simulator [36] to evaluate the performance metrics such as average network latency (in clock cycles), throughput (in flits/cycles/core), and router power consumption (in mW). We have considered the link fault in the FTTGs generated by the approaches [5], [9], [10], and our approach for the multi-media applications PiP, MPEG-4, MP3Encoder, and VOPD. Table 12 shows the NoC simulator configurations used while performing the dynamic simulations. With the parameter values mentioned in Table 12, the experiments have performed for FTTGs generated using our approach and the approaches [5], [9], [10].

TABLE 12. NoC simulator [36] configuration parameters and its values.

Parameter	Value
Flit size	32
Traffic	Application-specific
Routing algorithm	Table-based routing
Simulation time	1,00,000 (clock cycles)
Saturation time	10,000 (clock cycles)
Switching type	Wormhole
Number of routers	Approach [5]: PiP :- 5, MPEG-4 :- 7, MP3Encoder :- 8, VOPD :- 9 Approach [9], [10]: PiP - 4, MPEG-4 :- 6, MP3Encoder :- 7, VOPD :- 8
	Our approach: PiP - 4, MPEG-4 :- 6, MP3Encoder :- 7, VOPD :- 8
Ports per router	5
Clock period	5 ns
Flit length	32 bits
Flits per one packet	64 (1 Header, 62 Payload, & 1 Tailer)

Table 13 and Table 14 show the dynamic simulation results for the FTTGs generated using the approaches [5], [9], [10], and our approach by considering a specific link and any link fault. For a specific link fault in the FTTGs generated, the link with high communication bandwidth is considered for the dynamic simulations. For any link fault in the FTTGs generated, we have considered three different types of links having high, low, and medium communication bandwidth as failure. The results shown in Table 14 are calculated by taking the average of three different types of failed links. In Table 13 and Table 14, the first column represents the application, columns two to five represent the network latency, columns six to nine represent the throughput, columns ten to thirteen represent the router power consumption for the FTTGs generated using our approach and the approaches [5], [9], [10], respectively. A topology is said to be efficient if it has low network latency, high throughput, and low power consumption. As it can be seen from Table 13, the average network latency for our approach is less than the approaches [5], [9], [10]. This is because the approaches [5], [9], [10] have considered the base topology as Ring. If the topology has fixed connection patterns, then there exists minimal alternate routing paths. Therefore, in the event of link failure, from the static results it has been observed that the hop count for the approaches [5], [9], [10] is relatively higher than our approach. Since network latency is directly proportional to the hop count in the network [35], the approaches [5], [9], [10] have resulted in high network latency. On contrary, our approach has not fixed any topology to generate the FTTGs for different applications. Instead, based on the communication requirement between the cores in an application the FTTGs are generated for different applications. This added an advantage of selecting the routers to add a spare link in the event of link failure in the FTTGs of an application. From the static results, it has been observed that the FTTGs generated using our approach has less hop count compared to its counterparts. On an average, for specific link fault, our approach has shown significant improvements of 5.61%, 6.37%, and 3.72% in network latency over the approaches [5], [9], [10], respectively.

Compared to the approaches [9], [10], the throughput obtained for a specific link fault in the FTTG using our approach is high, but it is low when compared with [5]. According to [41], the throughput is defined as the ratio of total number of data packets received with in the time window to the maximum receive time. Therefore, the throughput of a network depends on the number of intermediate routers participated in the routing path for each edge in the application. If more number of intermediate routers are required to transmit the application traffic between the cores through the routers, the time required to pass the data packets between the cores will be high. Hence, the throughput of the network will be reduced if there are more number of routers involved in the network. As it can be observed from Table 12, the number of routers required to generate FTTGs using the approach [5] is one more than our approach and the approaches [9], [10]. This is because the approach [5] has added an extra router to provide two routing paths in the event of link failure. Since the throughput depends on the number of routers, evidently the FTTGs generated using the approach [5] has high throughput. On the contrary, the throughput for the FTTGs generated using the approaches [9], [10] is less than our approach. This is because these approaches use Ring topology as a base and generates FTTG. Throughput for any network is inversely proportional to the distance the data packets have travelled in the network [35]. Since, the approaches [9], [10] use longer data path to route the packets, the throughput is comparatively lesser than our approach. On an average, compared to the approach [5] there is an overhead of 4.19% in the overall throughput of the network obtained using our approach. However, there are improvements of 24.67% and 5.76% in throughput using our approach over the approaches [9], [10], respectively.

The router power consumption is calculated based on the switching activity of the router. It is directly proportional to the number of hops that the data packets have to pass in the network. As it can be seen from Table 13, the router power consumption for our approach and the approaches [9], [10] is less than the approach [5]. It is because the approaches [9], [10] and our approach have used less number of routers compared to its counterpart. According to [5], in the event of specific link failure, the data packets from the source to destination router require more routers to pass. Hence, the switching activity required for transmitting the packets is high. If the switching activity is high, then the router power consumption will be high [35]. Therefore, compared to other approaches [9], [10], there is an average improvement of 14.57% in power consumption using our approach over the approach [5].

Fig. 11 shows the overall percentage of improvements in dynamic simulation results for a specific link failure in the FTTGs generated using our approach over the approaches [5], [9], [10]. On X-axis, the approaches [5], [9], [10] are taken and on Y-axis the improvements in performance metrics such as network latency, throughput, and power consumption are shown. It can be noted that for a specific link fault in the FTTGs of different applications, our approach has shown improvements in performance metrics. These improvements are due to efficient generation of fault-tolerant application-

TABLE 13. Comparison of Dynamic simulation results for a specific link fault in the FTTGs generated using our approach and the approaches for different applications.

Application	Network Latency (in clock cycles)				Throughput (in flits/cycle/core)				Router power consumption (in mW)			
	Our approach	Approach [5]	Approach [9]	Approach [10]	Our approach	Approach [5]	Approach [9]	Approach [10]	Our approach	Approach [5]	Approach [9]	Approach [10]
PiP	75.42	85.07	75.69	83.02	0.00362	0.00362	0.00285	0.00302	71.32	89.18	71.29	71.32
MPEG-4	77.52	80.43	84.20	78.03	0.01279	0.01454	0.00922	0.01251	107.67	126.03	107.84	107.69
MP3Encoder	73.64	75.58	78.86	73.73	0.05290	0.04686	0.04429	0.04652	126.63	144.48	127.84	126.67
VOPD	74.40	78.23	83.20	78.27	0.00910	0.01043	0.00610	0.00982	143.05	161.35	143.37	143.49
Average	**75.24**	**79.83**	**80.49**	**78.26**	**0.01960**	**0.01886**	**0.01561**	**0.01797**	**112.17**	**130.26**	**112.59**	**112.29**

TABLE 14. Comparison of Dynamic simulation results for any link fault in the FTTGs generated using our approach and the approaches for different applications.

Application	Network Latency (in clock cycles)				Throughput (in flits/cycle/core)				Router power consumption (in mW)			
	Our approach	Approach [5]	Approach [9]	Approach [10]	Our approach	Approach [5]	Approach [9]	Approach [10]	Our approach	Approach [5]	Approach [9]	Approach [10]
PiP	78.63	81.33	79.59	80.37	0.00356	0.00312	0.00348	0.00293	71.32	89.13	71.34	71.31
MPEG-4	75.90	78.54	80.00	76.71	0.01289	0.01248	0.00935	0.01246	107.53	125.66	107.58	107.64
MP3Encoder	75.37	75.63	76.33	73.57	0.05212	0.04690	0.05050	0.04728	126.76	145.28	127.51	126.80
VOPD	74.97	75.90	78.35	75.80	0.00933	0.00998	0.00831	0.00925	143.07	161.13	143.29	143.23
Average	**76.22**	**77.85**	**78.57**	**76.61**	**0.01948**	**0.01812**	**0.01791**	**0.01798**	**112.17**	**130.30**	**112.43**	**112.25**

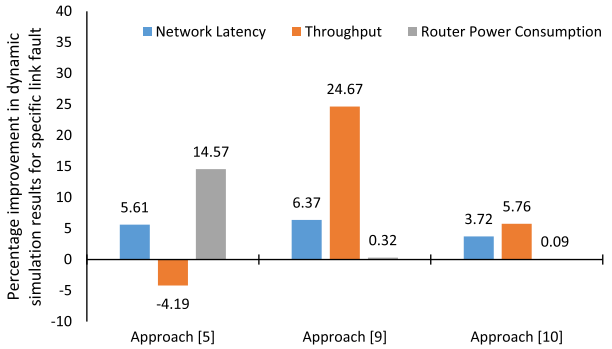


FIGURE 11. Percentage of improvements in dynamic performance metrics using our approach over the approaches [5], [9], [10] for the FTTGs generated for specific link fault.

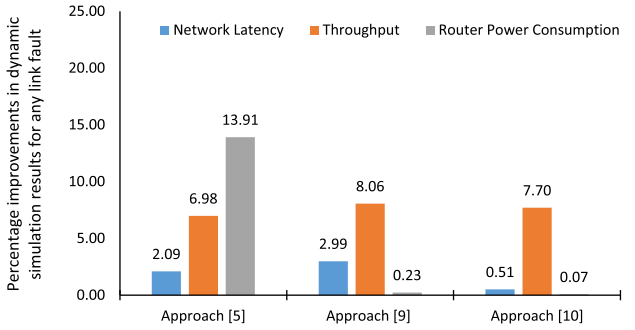


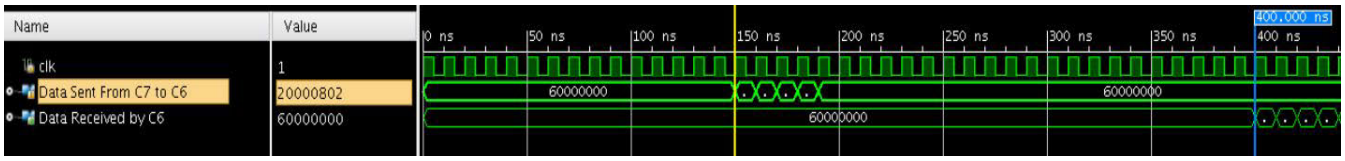
FIGURE 12. Percentage of improvements in dynamic performance metrics using our approach over the approaches [5], [9], [10] for the FTTGs generated for any link fault.

specific topologies using our approach. The results show that instead of generating the FTTGs by considering the Ring topology as base, custom topology generation has significant improvements in both static and dynamic performance metrics.

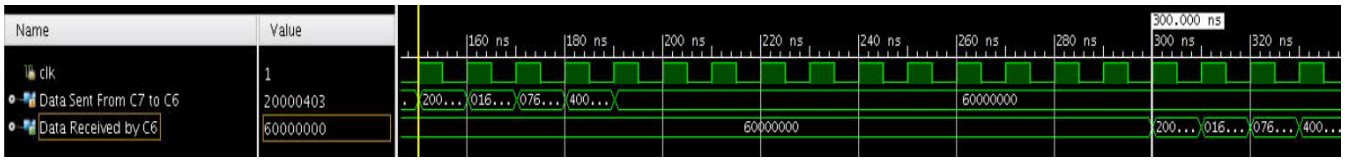
The similar trend can be seen in the case of any link fault in the FTTGs of different applications. As mentioned earlier, the results shown in Table 14 are obtained by taking the average of three link faults having high, medium, and low communication bandwidths. From Table 14, it can be seen that the network latency for different applications obtained using our approach is less than the approaches [5], [9], [10]. This is because for all the three types of link faults, the hop count in the FTTGs generated using our approach is less than its counterparts. The average percentage improvements in network latency for any link fault in different application FTTGs generated using our approach over the approaches [5], [9], [10] are 2.03%, 2.99%, and 0.51%, respectively. For throughput and router power consumption, there are significant improvements using our approach over the approaches [5], [9], [10]. Fig. 12 shows the percentage improvements in dynamic simulation results using our approach over its counterparts. As it can be seen that there is an improvement of 6.98%, 8.06%, and 7.70% in throughput using our approach over the approaches [5], [9], [10], respectively. As discussed earlier, compared to our approach and the approaches [9], [10], the approach [5] has used one additional router to route the data packets which resulted in high router power consumption. Therefore, there is an improvement of 13.91% in router power consumption using our approach over the approach [5]. On contrary, there are very little improvements in power consumption using our approach over the approaches [9], [10]. It is due to the fact that the number of routers used by our approach and the approaches [9], [10] are same.

C. FPGA IMPLEMENTATION

One of the novel contributions of this work is FPGA implementation of the fault-tolerant ASNoC design. We have implemented the FTTGs generated using our approach and the approach [5] on an FPGA. Among the approaches [5], [9], [10], the approach [5] has implemented the FTTG of an application on an FPGA, while the approaches [9], [10] have



(a) Simulation waveform of the data transferred from C7 to C6 using approach [5]



(b) Simulation waveform of the data transferred from C7 to C6 using our approach

FIGURE 13. FPGA simulation waveform for the FTTGs shown in Fig. 1.

limited to software implementation only. Therefore, we have considered the approach [5] for the comparison with our approach. However, the similar process can be applied to the other approaches [9], [10]. The NoC router architecture that has been implemented on a Kintex KC705 based FPGA is described in Section V.C. The major motivation for the FPGA implementation is to analyze the system performance and behavior in run-time. The NoC router architecture implemented on an FPGA works at 100 MHz clock frequency while providing link fault-tolerance to the design. Please note that the NoC router architecture has embedded with the fault-tolerant routing algorithm (Algorithm 1) discussed in Section V.D. Fault-detection and fault-localization in this work are done based on the MTTF analysis, whereas fault simulation can be considered as one of the possible extensions to this work. According to the MTTF analysis, based on the link failure the FTTGs are implemented on an FPGA. From the software implementation results, it has been seen that our approach has shown significant improvement over the approaches [5], [9], [10]. However, to understand the practical behavior of an application it is necessary to know the communication latency (in seconds), and number of data packets sent/received between the cores. The communication latency on an FPGA is defined as the time taken for one flit to either send or receive from source to destination routers. With this information, we have implemented the FTTGs generated using the approaches [5], [9], [10] and our approach by considering the link faults shown in Fig. 1. From the experimental results it has been observed that the time taken to run the applications (PiP, MPEG-4, MP3Encoder, and VOPD) on an FPGA using our approach is less when compared to the approaches [5], [9], [10]. This can be illustrated by considering MP3Encoder application as a case study.

Fig. 13 shows the screen-shot of FPGA simulation (Post-implementation) waveform results for the edge C6-C7 of MP3Encoder application. Please note that the results shown in Fig. 13 are for the FTTGs generated using the approach [5] and our approach by considering the link fault shown in Fig. 1(b) and 1(e), respectively. Here the link fault is considered between the routers r_2, r_4 and r_2, r_3 in the FTTGs generated using the approach [5] and our approach, respectively. As mentioned earlier in the software implementation section, the link fault is considered between the routers whose cores are communicating with high bandwidth. Since the cores mapped onto the routers are different for our approach and the approach [5], the link fault considered for the experiment would be different. However, the idea is to consider a link fault in the topology and analyse the network behavior on an FPGA. Fig. 13(a) represents the data sent from core C7 to C6 that has been received at C6 successfully. The yellow marker in the Fig. 13(a) indicates the time at which the data has sent from core C7, i.e., 150 ns, and the blue marker shows the time at which the data has received in the core C6, i.e., 400 ns. In the event of link fault, the cores C6 and C7 require four hops and two hops for the communication using the approach [5] and our approach, respectively. For four hops the time taken for the transfer of one flit using the approach [5] is 250 ns. Similarly for our approach, the data from core C7 - C6 has started at 150 ns of time (see Fig. 13(b)) and received at core C6 by 300 ns of time. The time taken for one flit to transfer from core C7 to C6 or C6 to C7 is 150 ns using our approach. The time taken for one flit to send/receive using our approach is less compared to approach [5].

Table 15 shows the total time taken (seconds) for the transfer of MP3Encoder application data using the FTTGs generated by the approach [5] and our approach, respectively. In Table 15, the first column represents the edge present in the MP3Encoder application. The second column represents the Bandwidth (in bits per second) of an edge, and the third column represents the number of flits required for each edge in the application, while each flit is of 32-bit size. The columns four to six and seven to nine represent the time taken for transferring the flits of each edge in the application using the approach [5] and our approach, respectively. On an FPGA, it has been noticed that the routers which are connected at one hop distance, has taken 100 ns for sending or receiving the data flits. However, for subsequent increment in the hop count, an additional delay of 50 ns is added to the design.

TABLE 15. Calculation of the total time taken for sending the application data from one core to the other on an FPGA.

Edge in MP3Encoder ACG	Bandwidth (bits per second)	No. of flits	Approach [5]			Our approach		
			Hop count	Time (ns)	Total time per flits (sec)	Hop count	Time (ns)	Total time per flits (sec)
C4-C3	524288	19421	2	150	0.002913	1	100	0.001942
C5-C4	1048576	38839	0	0	0	1	100	0.003884
C6-C8	188743.68	6993	5	300	0.002098	2	150	0.001049
C6-C7	157286.4	5828	4	250	0.001457	2	150	0.000874
C5-C6	912261.12	33790	1	100	0.003379	1	100	0.003379
C2-C5	1048576	38839	1	100	0.003884	0	0	0
C1-C2	2184183.808	80898	1	100	0.00809	2	150	0.012135
C1-C3	4257218.56	157677	0	0	0	0	0	0
C1-C9	26214.4	973	3	200	0.000195	1	100	9.73E-05
C9-C10	2184183.808	80898	0	0	0	0	0	0
C10-C13	10485.76	391	5	300	0.000117	2	150	5.87E-05
C12-C13	524288	19421	4	250	0.004855	1	100	0.001942
C12-C11	4262461.44	157871	0	0	0	0	0	0
Overall time taken for sending the data flits from source to destination					0.002076			0.001951

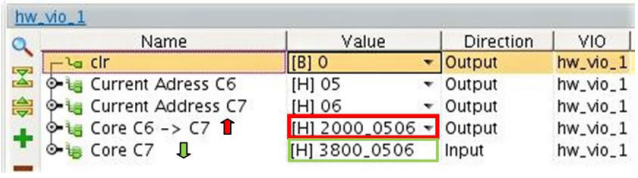


FIGURE 14. Screen-shot of the VIO IP core which shows the successful delivery of HF from source core C6 to destination core C7.

TABLE 16. Resource and on-chip power estimation for an MP3Encoder application implemented on an Kintex KC705 FPGA development board using our approach and the approach in [5].

Methodology	Estimation of Resources		On-chip power (in mW)
	LUTs	FFs	
Our approach (FTAS-SL)	19991	20979	0.279
Approach in [5]	22462	23848	0.285
Percentage Improvement	**11%**	**12.03%**	**2.10%**

Fig. 14 shows the screen-shot of the VIO IP core [42] window of the FTTG implemented on an FPGA using our approach and approach [5]. Since the flit is of 32-bit size, the inputs on the Kintex board may not suffice. Hence, we have used the VIO IP core for sending and receiving the data to the FPGA. This ensures that the data has successfully delivered from core C6 to the core C7 in an application. As we can observe from Fig. 14, there are current addresses to each core, data input, and output. Based on the router addresses and the fault-tolerant routing algorithm mentioned in Section V.D, the data flits will be routed to the destination. From Fig. 14 it can be seen that the data has been successfully delivered from core C6 to the destination core C7. Once the HF is delivered, then the payload and tailer flit will be sent. The tailer flit ensures that the complete application data has been transferred from the source core to the destination core. As we can observe from Table 15, the overall time taken to run the application using our approach is 0.001951 seconds, and the approach [5] is 0.002076 seconds. The time taken to run an application using our approach is less when compared to the approach [5]. This is because our approach has generated the fault-tolerant topologies with less hop count between the routers to communicate, while the approach [5] has used more number of hops to communicate between the routers. Therefore, compared to the approach [5], our approach has taken 6.02% less time to run MP3Encoder application. This shows the efficiency of our approach over the approach [5] in generating the fault-tolerant ASNoC design for any link fault. The estimation of resource utilization for the MP3Encoder application implemented on an FPGA using our approach and the approach [5] is shown in Table 16. The number of LUTs and FFs used by our approach in generating fault-tolerant application-specific NoC topology is less compared to the approach [5]. This is because the number of routers used for generating the topology using the approach [5] is high compared to our approach. Similarly, the on-chip power dissipation calculated using Xilinx Power Estimator (XPE) tool embedded in Vivado 2016.2 for our approach is less compared to the approach [5]. There are significant improvements using our approach over the approach [5] in terms of resources utilized on an FPGA and the on-chip power estimated using XPE tool. These improvements are due to the addition of spare link to the topology such that the design is optimized compared to the approach [5]. With the addition of spare link, it has been noticed that our approach has resulted in less hop count between the routers. The similar trend can be seen in other applications namely PiP, MPEG-4, and VOPD FTTGs generated by our approach and the approaches [5], [9], [10]. This completes the analysis of fault-tolerant application-specific topology implementation on an FPGA.

VII. CONCLUSION

We presented a fault-tolerant ASNoC design based on a PSO-based meta-heuristic algorithm and its FPGA implementation in this paper. Experiments were carried out for the multi-media applications PiP, MPEG-4, MP3Encoder, and VOPD while taking into account link failures in the generated topology. The results were compared to the approaches reported in the literature [5], [9], [10]. The results show that both software and FPGA implementations have improved the performance significantly over the existing approaches. Consideration of reconfiguration and scheduling policies for multiple applications that can be implemented on an FPGA are planned for the future.

VIII. LIMITATIONS OF THE STUDY

This work proposes a PSO algorithm for the spare link placement in the ASNoC topology. The fine-tuning of the PSO algorithm for the high number of cores in the application is a little challenging. Therefore, there is a scope of work in addressing the limitation of the PSO algorithm with other meta-heuristic algorithms.

REFERENCES

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

[2] M. Majer, C. Bobda, A. Ahmadinia, and J. Teich, "Packet routing in dynamically changing networks on chip," in *Proc. 19th IEEE Int. Parallel Distrib. Process. Symp.*, Apr. 2005, pp. 1–8.

[3] L. Benini, "Application specific NoC design," in *Proc. Design Autom. Test Eur. Conf.*, vol. 1, Mar. 2006, pp. 1–5.

[4] M. Radetzki, C. Feng, X. Zhao, and A. Jantsch, "Methods for fault tolerance in networks-on-chip," *ACM Comput. Surveys*, vol. 46, no. 1, pp. 1–38, Oct. 2013.

[5] S. Yesil, S. Tosun, and O. Ozturk, "FPGA implementation of a fault-tolerant application-specific NoC design," in *Proc. Int. Conf. Design Technol. Integr. Syst. Nanosc. Era (DTIS)*, Apr. 2016, pp. 1–6.

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

[7] N. Jiang, G. Michelogiannakis, D. Becker, B. Towles, and W. J. Dally, "BookSim 2.0 user's guide," Standford Univ., Standford, CA, USA, Tech. Rep. 1, 2010.

[8] R. Al-Badi, M. Al-Riyami, and N. Alzeidi, "A parameterized NoC simulator using OMNet++," in *Proc. Int. Conf. Ultra Modern Telecommun. Workshops*, Oct. 2009, pp. 1–7.

[9] S. Tosun, V. B. Ajabshir, O. Mercanoglu, and O. Ozturk, "Fault-tolerant topology generation method for application-specific network-on-chips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 9, pp. 1495–1508, Sep. 2015.

[10] P. Kullu and S. Tosun, "Energy-aware and fault-tolerant custom topology design method for network-on-chips," *Nano Commun. Netw.*, vol. 19, pp. 54–66, Mar. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1878778918300863

[11] S. Tosun, V. B. Ajabshir, O. Mercanoglu, and O. Ozturk, "Fault-tolerant irregular topology design method for network-on-chips," in *Proc. 17th Euromicro Conf. Digit. Syst. Design*, Aug. 2014, pp. 631–634.

[12] Y.-X. Zheng, P.-P. Kan, L.-B. Chen, K.-Y. Hsieh, B.-C. Cheng, and K. S.-M. Li, "Fault tolerant application-specific NoC topology synthesis for three-dimensional integrated circuits," in *Proc. IEEE Int. SOC Conf.*, Sep. 2011, pp. 296–301.

[13] Z. Li, J. Huang, Q. Xu, and S. Chen, "Integer linear programming based fault-tolerant topology synthesis for application-specific NoC," in *Proc. IEEE 12th Int. Conf. ASIC (ASICON)*, Oct. 2017, pp. 96–99.

[14] P. Shah, A. Kanniganti, and J. Soumya, "Fault-tolerant application specific network-on-chip design," in *Proc. 7th Int. Symp. Embedded Comput. Syst. Design (ISED)*, Dec. 2017, pp. 1–5.

[15] N. Venkataraman and R. Kumar, "Design and analysis of application specific network on chip for reliable custom topology," *Comput. Netw.*, vol. 158, pp. 69–76, Jul. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128618307254

[16] P. Narayanasamy, S. Gopalakrishnan, and S. Muthurathinam, "Custom NoC topology generation using discrete antlion trapping mechanism," *Integration*, vol. 76, pp. 76–86, Jan. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167926020302728

[17] J.-A. Carballo, W.-T.-J. Chan, P. A. Gargini, A. B. Kahng, and S. Nath, "ITRS 2.0: Toward a re-framing of the semiconductor technology roadmap," in *Proc. IEEE 32nd Int. Conf. Comput. Design (ICCD)*, Oct. 2014, pp. 139–146.

[18] P. Gargini, "Roadmap evolution: From NTRS to ITRS, from ITRS 2.0 to IRDS," in *Proc. 5th Berkeley Symp. Energy Efficient Electron. Syst. Steep Transistors Workshop (E S)*, 2017, pp. 1–62.

[19] F. M. Sajjade, N. K. Goyal, and B. K. S. V. L. Varaprasad, "Single event transient (SET) mitigation circuits with immune leaf nodes," *IEEE Trans. Device Mater. Rel.*, vol. 21, no. 1, pp. 70–78, Mar. 2021.

[20] H. Cha, E. M. Rudnick, J. H. Patel, R. K. Iyer, and G. S. Choi, "A gate-level simulation environment for alpha-particle-induced transient faults," *IEEE Trans. Comput.*, vol. 45, no. 11, pp. 1248–1256, Nov. 1996.

[21] N. Miskov-Zivanov, K.-C. Wu, and D. Marculescu, "Process variability-aware transient fault modeling and analysis," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2008, pp. 685–690.

[22] S. Bhunia, S. Mukhopadhyay, and K. Roy, "Process variations and process-tolerant design," in *Proc. 20th Int. Conf. VLSI Design Held Jointly 6th Int. Conf. Embedded Syst. (VLSID)*, 2007, pp. 699–704.

[23] J. Gracia, L. J. Saiz, J. C. Baraza, D. Gil, and P. J. Gil, "Analysis of the influence of intermittent faults in a microcontroller," in *Proc. 11th IEEE Workshop Design Diag. Electron. Circuits Syst.*, Apr. 2008, pp. 1–6.

[24] P. S. Ho and T. Kwok, "Electromigration in metals," *Rep. Prog. Phys.*, vol. 52, no. 3, p. 301, Mar. 1989.

[25] C. E. Blat, E. H. Nicollian, and E. H. Poindexter, "Mechanism of negative-bias-temperature instability," *J. Appl. Phys.*, vol. 69, no. 3, pp. 1712–1720 1991.

[26] D. Boudreaux, F. Williams, and A. Nozik, "Hot carrier injection at semiconductor-electrolyte junctions," *J. Appl. Phys.*, vol. 51, no. 4, pp. 2158–2163, 1980.

[27] G. R. Wilkinson, "Digital circuit wear-out due to electromigration in semiconductor metal lines," M.S. thesis, Dept. Elect. Eng., California Polytech. State Univ., San Luis Obispo, CA, USA, 2009, p. 196.

[28] N. Baptistat, K. Abouda, G. Duchamp, and T. Dubois, "Effects of process-voltage-temperature (PVT) variations on low-side MOSFET circuit conducted emission," in *Proc. 12th Int. Workshop Electromagn. Compat. Integr. Circuits (EMC Compo)*, Oct. 2019, pp. 213–215.

[29] T. Sudo, H. Sasaki, N. Masuda, and J. L. Drewniak, "Electromagnetic interference (EMI) of system-on-package (SOP)," *IEEE Trans. Adv. Packag.*, vol. 27, no. 2, pp. 304–314, May 2004.

[30] P. Maheshwari, T. Li, J.-S. Lee, B.-S. Seol, S. Sedigh, and D. Pommerenke, "Software-based analysis of the effects of electrostatic discharge on embedded systems," in *Proc. IEEE 35th Annu. Comput. Softw. Appl. Conf.*, Jul. 2011, pp. 436–441.

[31] A. Khare, C. Patil, and S. Chattopadhyay, "Task mapping and flow priority assignment of real-time industrial applications for network-on-chip based design," *Microprocessors Microsyst.*, vol. 77, Sep. 2020, Art. no. 103175.

[32] P. V. Bhanu, C. Vudadha, and J. Soumya, "FILA: Fault-model for interconnection links in application-specific network-on-chip design," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, pp. 1–5.

[33] 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.

[34] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw. (ICNN)*, vol. 4. Nov. 1995, pp. 1942–1948.

[35] 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.

[36] 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.

[37] D. B. Johnson, "A note on Dijkstra's shortest path algorithm," *J. ACM*, vol. 20, no. 3, pp. 385–388, Jul. 1973.
[38] Dally and Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, vols. COM-36, no. 5, pp. 547–553, May 1987, doi: 10.1109/TC.1987.1676939.
[39] 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.
[40] X. Kintex. *FPGA KC705 Evaluation Kit*. Accessed: May 21, 2021. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/ek-k7-kc705-g.html
[41] P. V. Bhanu and S. J., "Fault-tolerant application mapping on mesh-of-tree based network-on-chip," *J. Syst. Archit.*, vol. 116, Jun. 2021, Art. no. 102026.
[42] *Virtual Input/Output V3.0 Logicore IP Product Guide*, Xilinx, San Jose, CA, USA, 2018, p. 24.

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

From 2016 to 2017, he was an Electronic Design Intern at Panacea Medical Technologies, Bengaluru, India. From 2017 to 2020, he was a Junior Research Fellow with the Department of EEE, BITS-Pilani, Hyderabad Campus, India, 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 Department of Electrical and Electronics, BITS-Pilani, Hyderabad Campus, India, in 2020. His research interests include digital design, computer architecture, VLSI design, and FPGA implementation.

RAJAT KUMAR received the bachelor's degree in engineering from the Department of Electrical and Electronics, BITS-Pilani, Hyderabad Campus, India, in 2020. His research interests include computer architecture and VLSI design.

VISHAL SINGH received the bachelor's degree in engineering from the Department of Electrical and Electronics, BITS-Pilani, Hyderabad Campus, India, in 2020. His research interests include computer architecture, VLSI design, and different verification methodologies.

J. SOUMYA (Member, IEEE) received the bachelor's degree in electronics and communication engineering from Jawaharlal Nehru Technological University Hyderabad, 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" at Indian Space Research Organization (ISRO), Bengaluru, India. From 2014 to 2015, she was a Faculty Member at 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, 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 Delhi, 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 in mixed signal circuit design at Texas Instruments, 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, Grimstad, Norway. His main research interests include cyber-physical systems, autonomous systems, and wireless embedded systems.