Hindawi
Journal of Sensors
Volume 2023, Article ID 9753534, 1 page
https://doi.org/10.1155/2023/9753534

Retraction

Retracted: Integrated Sensory Throughput and Traffic-Aware Arbiter for High Productive Multicore Architectures

Journal of Sensors

Received 12 December 2023; Accepted 12 December 2023; Published 13 December 2023

This article has been retracted by Hindawi, as publisher, following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of systematic manipulation of the publication and peer-review process. We cannot, therefore, vouch for the reliability or integrity of this article.

Please note that this notice is intended solely to alert readers that the peer-review process of this article has been compromised.

Wiley and Hindawi regret that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] T. Venkata Sridhar and G. C. Krishnaiah, "Integrated Sensory Throughput and Traffic-Aware Arbiter for High Productive Multicore Architectures," *Journal of Sensors*, vol. 2022, Article ID 2911777, 14 pages, 2022.

Hindawi
Journal of Sensors
Volume 2022, Article ID 2911777, 14 pages
https://doi.org/10.1155/2022/2911777

Research Article

Integrated Sensory Throughput and Traffic-Aware Arbiter for High Productive Multicore Architectures

T. Venkata Sridhar [1,2] **and G. Chenchu Krishnaiah** [3]

[1]*Dept. of E&C, VTU, Belgaum 5900018, India*
[2]*Dept. of ETC, IIIT-BH, Bhubaneswar 751003, India*
[3]*Dept. of ECE, ASCET, Gudur 524101, India*

Correspondence should be addressed to T. Venkata Sridhar; venkatasridhar@iiit-bh.ac.in

Received 27 June 2022; Accepted 11 August 2022; Published 5 September 2022

Academic Editor: C. Venkatesan

The increasing demand for network and high-performance devices requires large data throughputs with minimal loss or repetition. Network on chips (NoC) provides excellent connectivity among multiple on-chip communicating devices with minimal loss compared with old bus systems. The motivation is to improve the throughput of the NoC that integrated on multicores for communication among cores by reducing the communication latency. The design of the arbiter in the crossbars switch of an NoC's router has a vital role in judging the system's speed and performance. Low latency and high-speed switching are possible with high performance and good switching equipment at the network level. One of the significant components in NoC under SoC design is the arbiter, which governs the system's performance. Proper arbitrations can avoid network or traffic congestions like livelock and buffer waiting. The proposed work in this paper is to design an efficient and high productive arbiter for multicore chips, especially SoCs and CMPs. The proposed arbiter is showing good improvement in the throughput at higher data rates; an average of more than 10% throughput improvement is noticed at higher flit injection rates independent of the VCs implemented. Further, the critical delays are reduced to 15.84% with greater throughputs.

1. Introduction

High-performance computing devices are the most regular devices at a compact level in the day-to-day lifestyle of humans in the present era. Recently, even large computing equipment becoming handy and lightweight requires the integration of larger hardware components at a large scale into compact modules resulting in an SoC (system on a chip). The design of an SoC [1, 2] is a common phenomenon in the modern manufacturing of major electronics. To meet the QoS (quality of service) of these manufacturing technologies at a scaling of nanometer level, factors like latency, power dissipation, error rate, and software error rates [3, 4] need to be appropriately managed [5]. Embedded SoC is application-oriented and requires an uncompromised communication interface for different environments on the chip, like processor(s), memory, control module, sometimes form firmware to the software. Due to the great demand in processing speed, the manufacturers had increased the number of processors integrated on a single chip considerably, starting from dual core to many cores [6]. MPSoC [7] are the new-generation manufacturing methods from the past one and a half-decade, giving a reasonable throughput for multitasking. Figure 1 shows a job run of a typical application-oriented MPSoC. Smart MPSoC contains integrated hardware [8].

Before mapping the data to be handled by multiple processors, it is essential to configure and evaluate the requirements as per the specific task. To map different cores on a single chip, like processor(s), DRAM(s)/SRAM(s), DSP(s), video processors, and DMA, the established bus technology has many compromises.

NoC is a simple solution for such compromises to overcome. An NoC reduces the burden of calculation from the

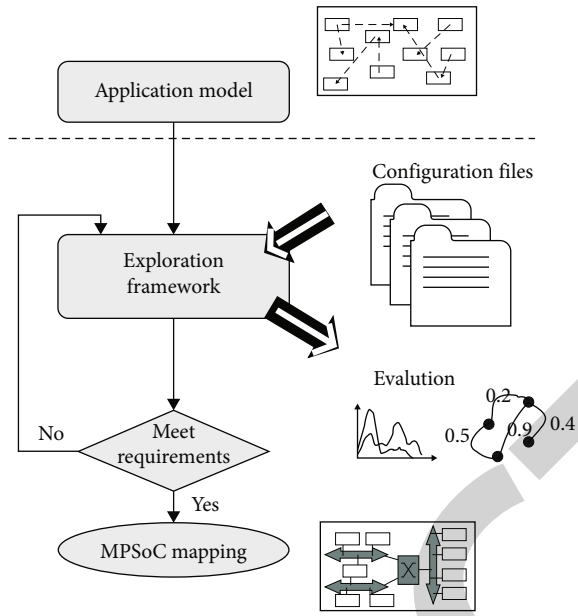

Figure 1: Job run of MPSoC.

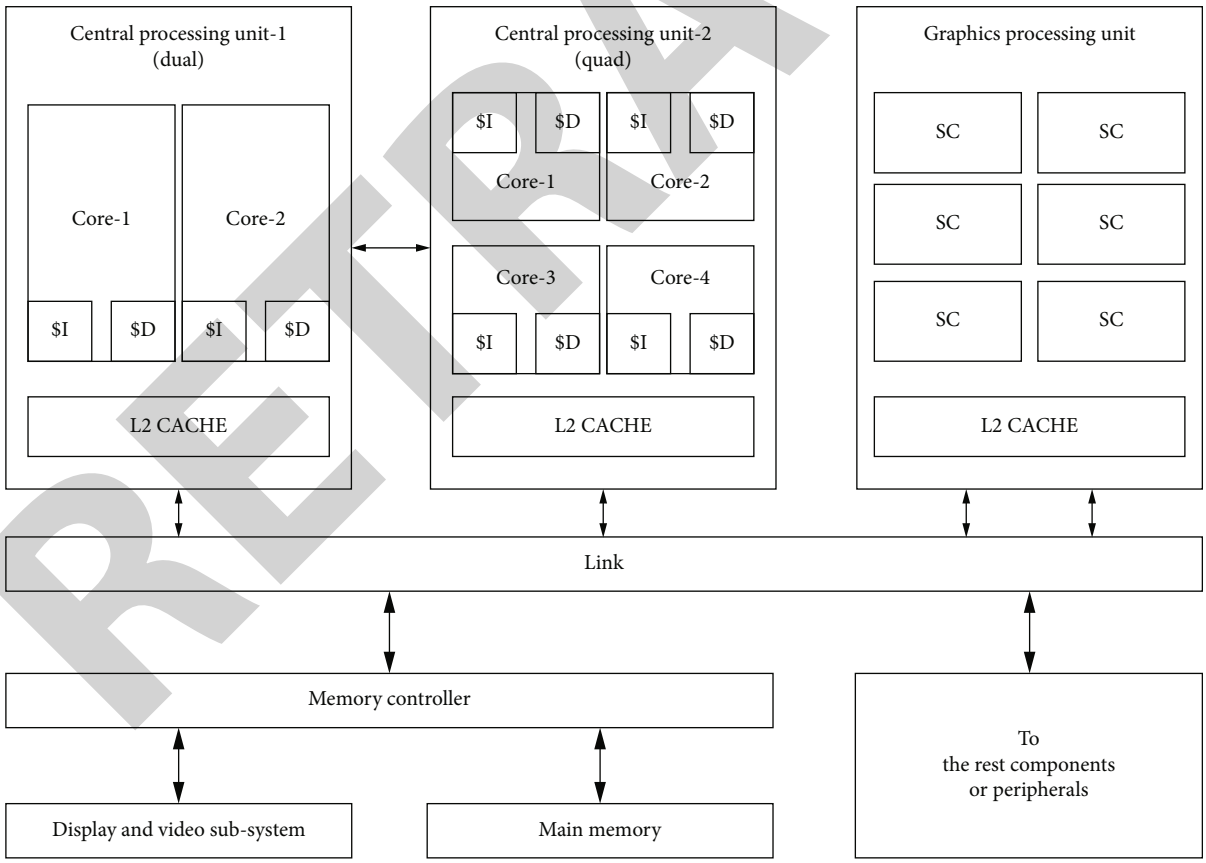

Figure 2: On-chip blocks of a multicore architecture.

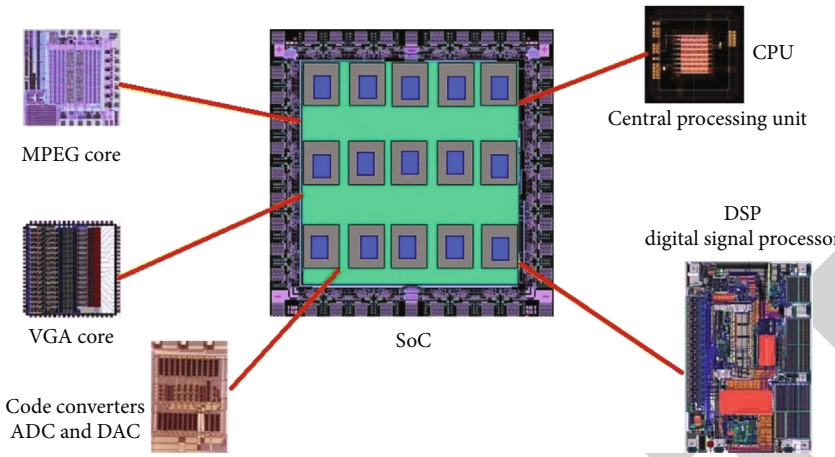

Figure 3: Device-level visualization of SoC.

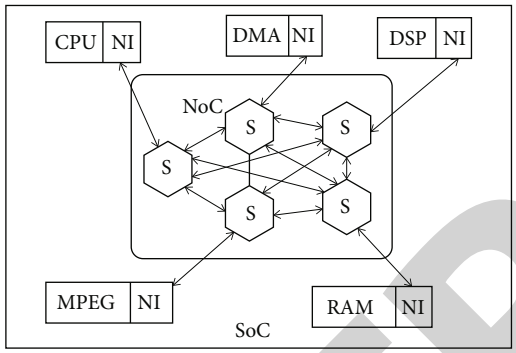

Figure 4: Fundamental NoC structure.

transmission. An efficient NoC requires good routing, network interfacing, and switching. A crossbar with a well-structured arbiter design is inevitable. The major contribution of the work it to improve the throughput of the arbiter in critical traffic timings that was achieved through a flexible priority resolver. The proposed arbiter is tested under synthetic traffic (ST) and uniform random traffic (URT) conditions. For both conditions, improvement in the throughput was observed. If the communication latency between the cores is reduced, then the time taken for data traversal is reduced; hence, this method can produce more throughput in a small time.

The major challenge in the existing models is handling the data when a network congestion occur, like a deadlock or a livelock. The existing model arbiter has fixed arbitrations for all the traffic conditions and is suffering if traffic conditions are unpredictable. The proposed model has a flexible priority resolver that can adjust the arbitrations according to the traffic. The proposed arbiter is showing good improvement in the throughput at higher data rates; an average of more than 10% throughput improvement is noticed at higher flit injection rates independent of the VCs implemented. Further, the critical delays are reduced to 15.84% with greater throughputs.

The process of mapping is information exchange between multicores or processing elements in the specified architecture. An efficient rouging on the NoC will execute the same without any deadlock or livelock. In the proposed model, the NoC with a flexible priority arbiter is enabling efficient information exchange between all cores in MPSoC. Many deep learning and AI platforms are in urge for high data processing systems; hence, this paper aims for such design [9–11].

Section 2 presents multicore architecture design and NoC interconnections and problems in design. Section 3 covers NoC crossbar switching with a well-organized arbiter design. Section 4 includes simulations of the proposed work and analysis. Section 5 is the conclusion containing limitations and plans of the work.

2. Multicore Systems

Multicore architectures are the best inevitable design for high-performance computers to handle large and complex data. Its architecture and design provide all the rising performance needs. Figure 2 illustrates the basic building blocks of heterogeneous SoC. And Figure 3 depicts device-level visualization.

2.1. Multiprocessor Architecture. The basic design of all multicore processors is to integrate the required data memories and instruction memories, i.e., L1 cache, secondary memories, i.e., L2 cache, on the chip itself. That is shown in Figure 2.

A proper link between different modules is required so that it should not degrade the performance of SoC. This is not achievable with the age-old bus systems. [3]. Either an SoC is general or application-specific; the NoC provides an exceptional solution for on-chip communication [12, 13].

2.2. The Communication Interface: NoC. As NoC is emerging as the leading solution to interface all on-chip devices in an SoC [14, 15], the design elements should be keenly followed. Like, choosing topology for implementation,

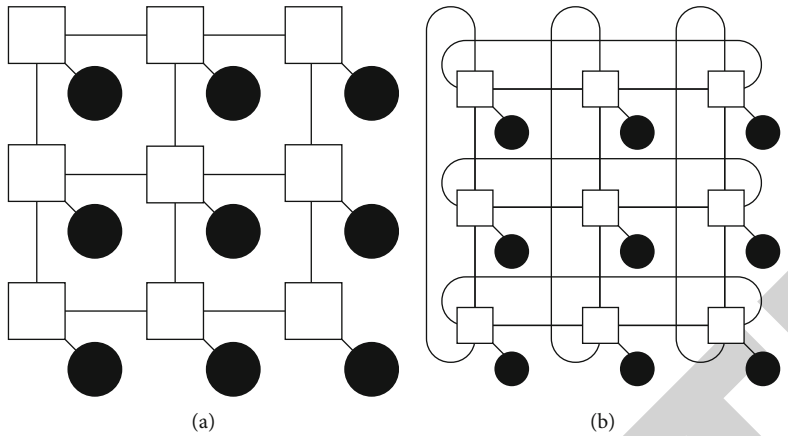

Figure 5: (a) A 3×3 mesh in 2D with one core connected to one router. (b) A 3×3 torus in 2D with one core connected to one router.

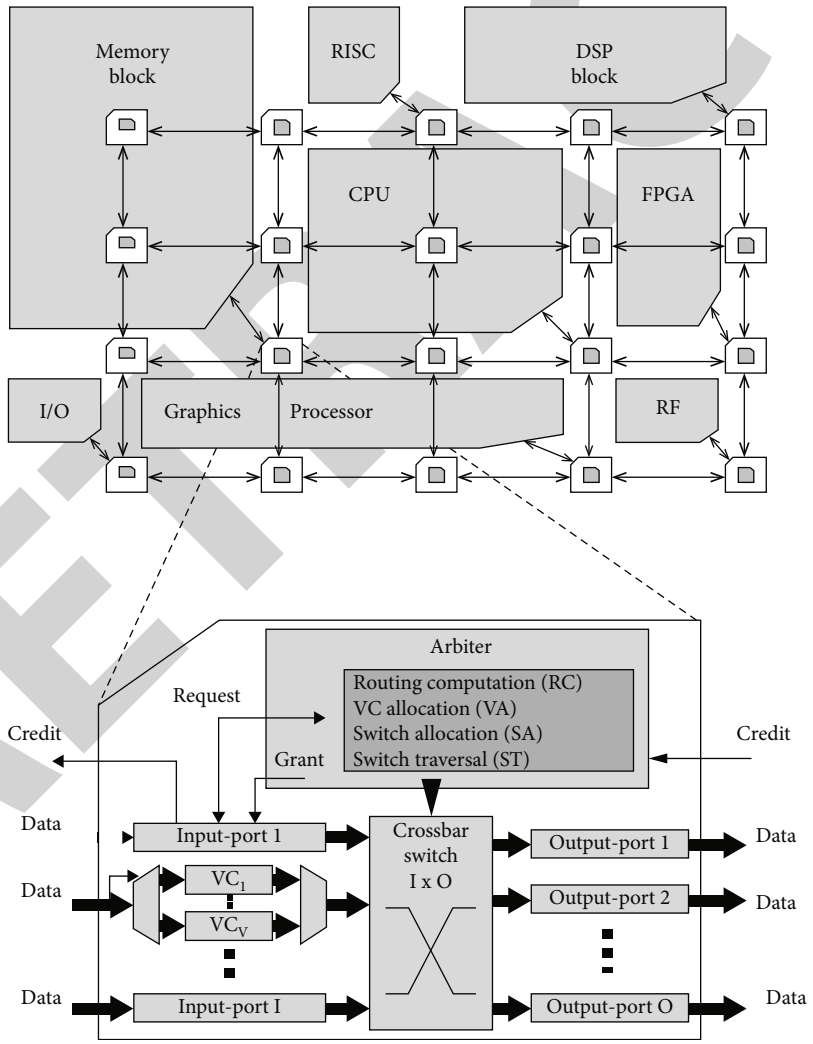

Figure 6: Different cores of SoC connected using NoC.

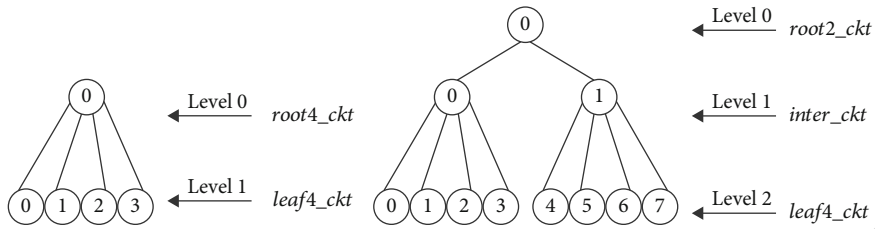

Figure 7: Decentralization tree of RRA.

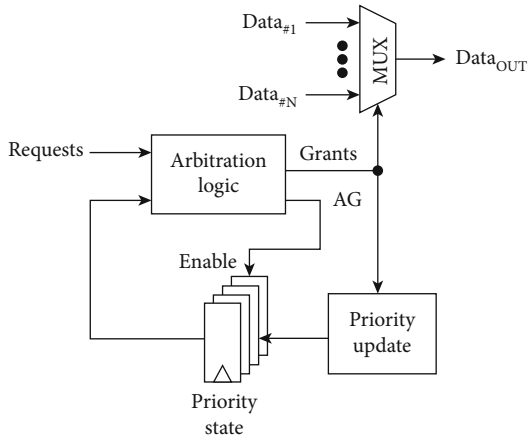

Figure 8: MARX structure.

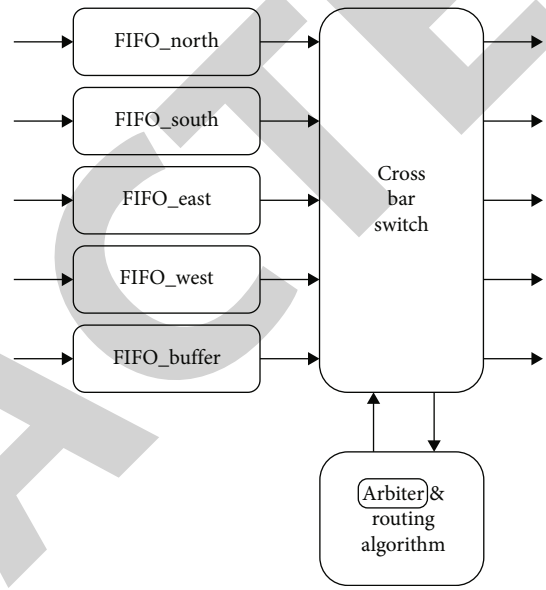

Figure 9: Data switching blocks.

maximum sustainable and error-free routing, and flow-control system [16, 17] are the vital elements in the design of an NoC. Figure 4 presents the basic block level understanding of the network on chip. Application-specific ONoCs (Optical NoCs) are even embedding the optics in microelectronics [18].

The aim of an NoC as the heart of SoC is to provide an efficient, mostly stand-off free, power, and throughput aware communication amidst different cores of the system on chip [19]. Modern SoC designs are coming with FinFETs, which can improve communication speed [20].

2.3. The NoC Architectonic. Network on chip architectonic contains many building blocks; among most important are *topology*, *interface*, and *routing* selection, for an efficient communication establishment.

2.3.1. Selection of Topology. The selection of the topology decides the area, power consumption, and speed of communication between the connected cores; it should be picked according to the need. That is as Application Specific (AS) or non-AS designs of SoC. Topologies like "Star," "Mesh," "Torus," "Octagon," "Spidergon," and "Tree" in 2D and 3D [21–23]. Mesh and Torus (sometimes folded) are the regular in power considerations, shown in Figure 5. The average distance between cores should be minimum concerning a hop count.

2.3.2. Interface Design. An interface provides communication between the core and the network with an assured throughput. An NI (network interface) will do assemble and disassemble packets and communicate them with the core. Proper choosing of network topology and routing techniques will improve the efficiency of the network interface.

2.3.3. Routing. The communication path from the source core to the destination core through NoC, without any congestions or blockages, is the aim of routing [24]. Defining a proper routing algorithm is needed to decide the latency in communication. Figure 6 illustrates a block diagram of the complete NoC architecture.

2.4. Crossbar Switch. After finalizing the routing and packing, it is required to switch the data according to the algorithm defined by NoC. Each data packet is Muxed thorough $I \times O$ crossbar (Figure 6). Per one cycle, one defined data packet can go through.

Figure 6 is a heterogeneous MPSoC architecture in which each tile contains different processing elements like memory block, DSP block, reduced instruction set computer, CPU, and graphic processor. An efficient networking architecture requires communicating data sharing among all the modules/processing elements. A single block of network interface/router is expanded in dotted line, which consists of a mechanism for data exchange in all four directions of connections. This is explained in section II-C: 2-3, D-E,

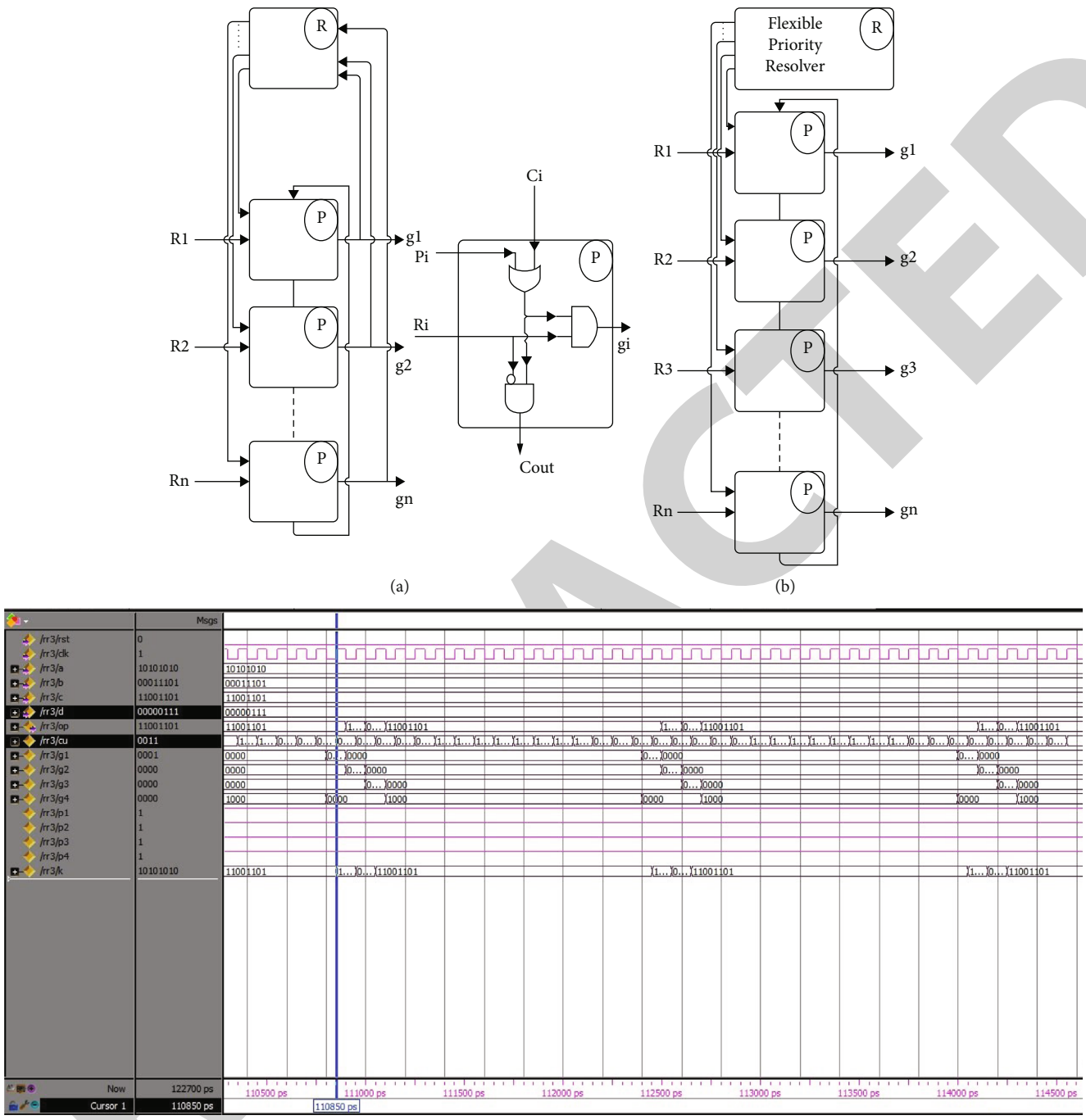

Figure 10: (a) Existing arbiter, (b) proposed flexible arbiter, and (c) arbitration simulation result.

respectively. The performance of the router directly depends on the efficiency of the crossbar switch and arbiter design. Hence, to achieve a low communication latency, these two are playing a very important role.

2.5. Arbiter. The arbitrations are dependent on the selection of virtual channels or the wormhole method. The arbiter is generally responsible for allowing the channel usage for all the inputs according to the routing strategies. Prioritization is more vital because it decides the speed of communication. In this research, we used a flexible priority such that livelock will be mostly avoided. Once all the input data reached the crossbar, the stages of the arbiter will be terminated accordingly.

Kameda et al. [25] had proposed an SFQ design for validating the crossbar. The idea is to increase the throughput

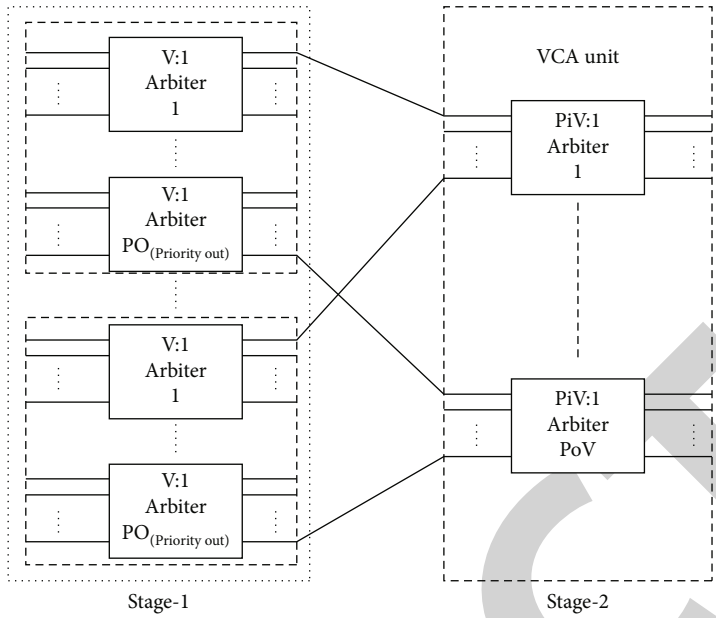

Figure 11: VCA arbitration.

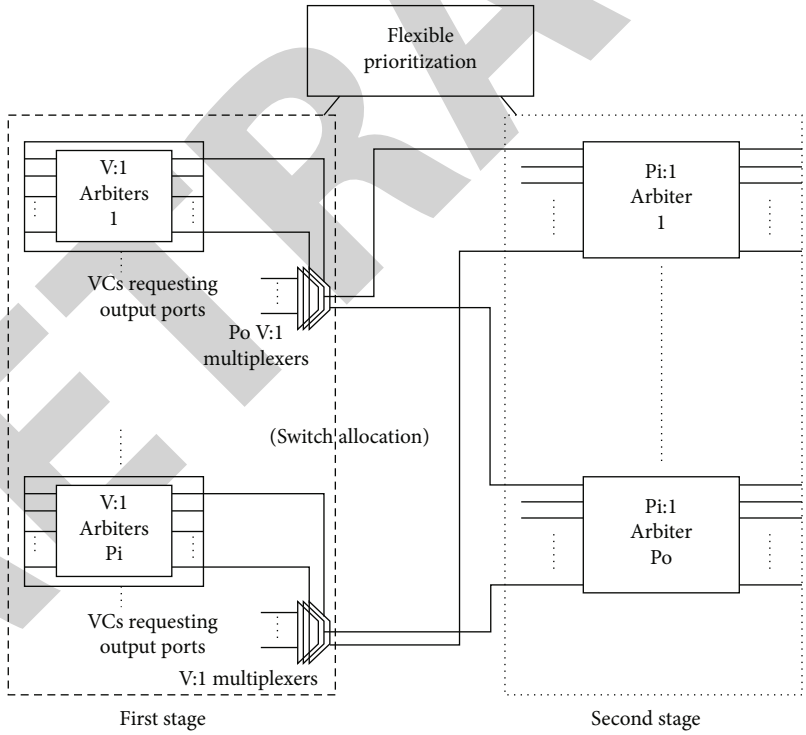

Figure 12: FP switch allocator.

to a considerable level by avoiding collisions. The experiment is limited to the usage of frequency and is more accurate at 40 GHz. Prioritization is a parallel processing type. The RRA design leads to high speed with frequency limitation.

Lee et al. [26] had reported a decentralized arbiter with high speed (HDRA). This design is aimed only at the most censorious path delays to minimize the production cost. The VOQs introduced in the design will reduce the blocking of the head of the line. HRDA is a derived version of PPA (O(log 4 N)structure), PRRA, and IPRRA as given in Figure 7.

When the integration of the cores in SoC increases, the design suffers from a large amount of complexity to find

Step 1: *check all input buffer requests.*
Step 2: *is the load/traffic more than regular transfer with respect to previous router data.*
Step 3: *chose to assign priority as rotation if traffic increases else, fixed.*
Step 4: *map i/p-port to o/p-port of arbiter for packet transfer.*
Step 5: *check for replication of the same packet assigned to various arbiters.*
Step 6: *check packets of o/p-port mapping to i/p-port according to the assignment.*
Step 7: *verify for the multiple iterations.*
Step 8: *issue grant.*

Algorithm 1

all the decentralized nodes by NoC and end assembly will become an issue.

Giorgos et al. [27] have suggested a simple design without much changes in the existing design, but with intelligent adoption. The approach is merging of the switching allocation, i.e., MARX which combines the multiplexer and arbiter as depicted in Figure 8. This design focuses majorly on the performance of the entire system, which increases the complexity in design. The Merged ARbiter and multipleXer (MARX) combines the architectures of an arbiter and multiplexer.

3. Arbiter Design

The significant component in NoC under SoC design is the arbiter, which governs the system's performance. Proper arbitrations can avoid network or traffic congestions like livelock and buffer waiting with both synchronous and asynchronous communication [28]. The proposed work in this paper is to design an efficient and high productive arbiter for multicore chips, especially SoCs and CMPs. The switching of the arbiter is shown in Figure 9. The design is a mesh topology and can communicate in four directions with a local transfer, which deserves a buffer count of five.

RR (round robin) arbiter model is the most commonly used method for NoC router design because of its ease and straightforwardness. Let us use RQ_n: request, GT_n: grant, and PR_n and $\widehat{PR_n}$: priorities of current and immediate future cycle. Kin_n and $Kout_n$ decide the priorities between arbiter buffer or cells. Then, according to RRA design, the grant will be issued only when PR_n is 1 as follows:

$$GT_n = RQ_n \cdot (PR_n + Kin_n), \tag{1}$$

$$\overline{Kout}_n = RQ_n \cdot (PR_n + Kin_n), \tag{2}$$

$$\widehat{PR_n} = GT_{n-1} + PR_n \cdot Kin_n. \tag{3}$$

Thus, the channel allocation for the next stage of traffic purely depends on the current running stage and followed by a subsequent request of existing channel using data. This is valid only if at least one present state exists. This drawback was modified with a priority resolver in our design, which estimates the density of the future traffic depending on the input requests to the router (possibly N:E:W:S direction) and load on previous router which directed the current transfer as shown in Figure 10. It has two efficient prioritizations chosen dynamically as fixed or rotating [29–32].

3.1. VCA Arbiter. Congestion avoidance, like a deadlock and livelock, can be handled by the arbiter. Flit bifurcation in routing is done based on the size of the data, the head flit may be one, but the body flits are packet size-dependent. The count is calculated before it reaches the VC (virtual channel). The allocation unit of the VC is done with a head flit. VCA unit allocates the channel for the data packets to travel through different routers. The mechanism of the two stages is illustrated in Figure 11.

3.2. Switching Arbiter. Once VC is allocated to the packets, the central part is to do switch allocation (SA), which leads to reaching the packet to the destination by competing with all the other VC allocated packets. Here, all the flits need to be allocated. To reduce the delay in communication to the destination, a flexible priority resolver is introduced in this proposed architecture. This reduced the computation complexity of allocation among all the VC allocations shown in Figure 12. Switch traverse (ST) and link traverse (LT) will lead the granted allocation packets to reach the specified destination.

3.3. Proposed Arbiter. Most of the existing arbiter designs follow a round robin which rotates the output grants received and precedes the rest Figure 10(a). This will increase the burden on the arbiter as the flits increase for large throughputs. The proposed arbiter has a flexible priority resolver which estimates the traffic/load on the arbiter about to come Figure 10(b). Unlike normal RRA, the specific rotation or fixed orientation is predefined here; hence, the time calculation of arbitrations can reduce, and throughput will increase.

The priority resolver has flexibility depending on the traffic, like fixed, rotating. If traffic is less and the expected latency in switching is less, then fixed priority can be chosen; if not, rotating priority can be chosen to distribute equal priority to all switching flits. The structure of the algorithm is as follows.

3.4. Proposed Arbiter Algorithm. The algorithm below specifies the arbiter's input request and grants according to flexible priority, depending on the load/traffic buffer channel rout calculation; VCA, SA, ST, and LT are the significant concerns in computing the travel delay. In the proposed arbiter, important computations are done in the stage of

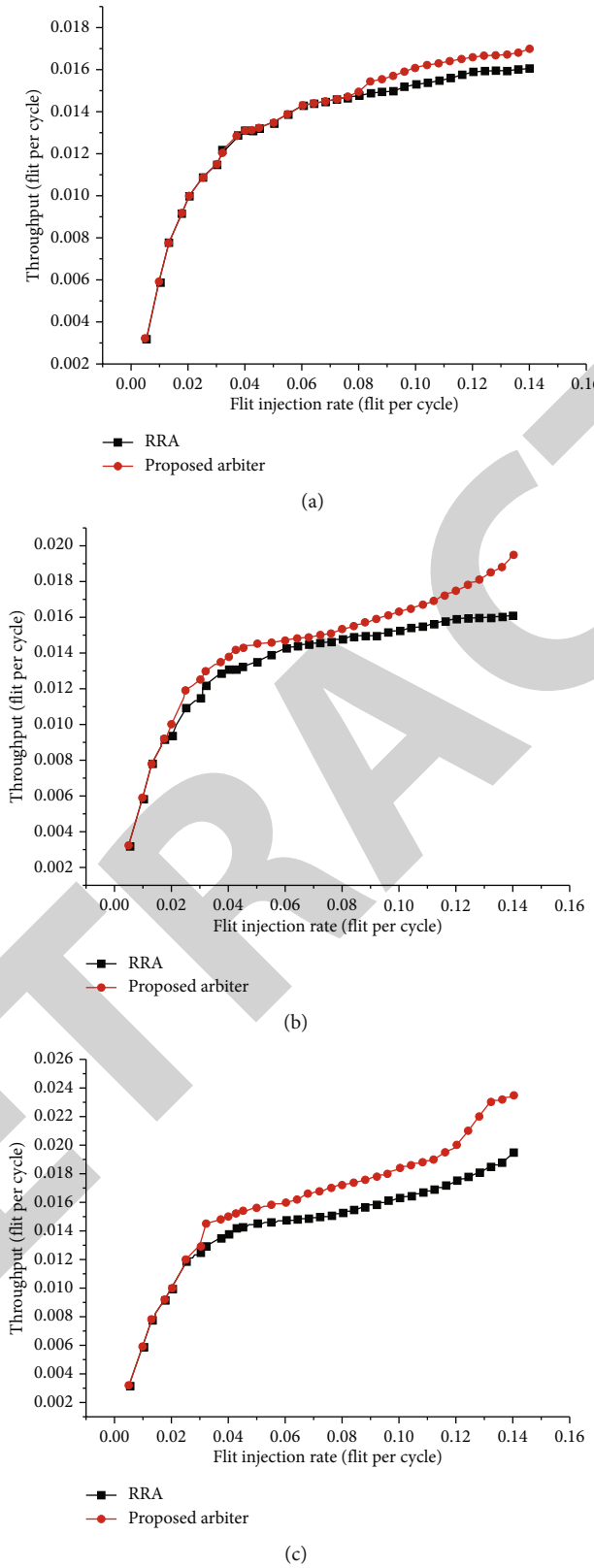

FIGURE 13: Throughput under URT. (a) FIR vs. throughput with two VCs, (b) FIR vs. throughput with four VCs, and (c) FIR vs. throughput with eight VCs, respectively.

priority resolving hence expected low latency with high throughput. First, it verifies all allocated buffers in step 1, finds the density of the traffic load and enables the priority resolver if found large loads in steps 2-3, and assigns

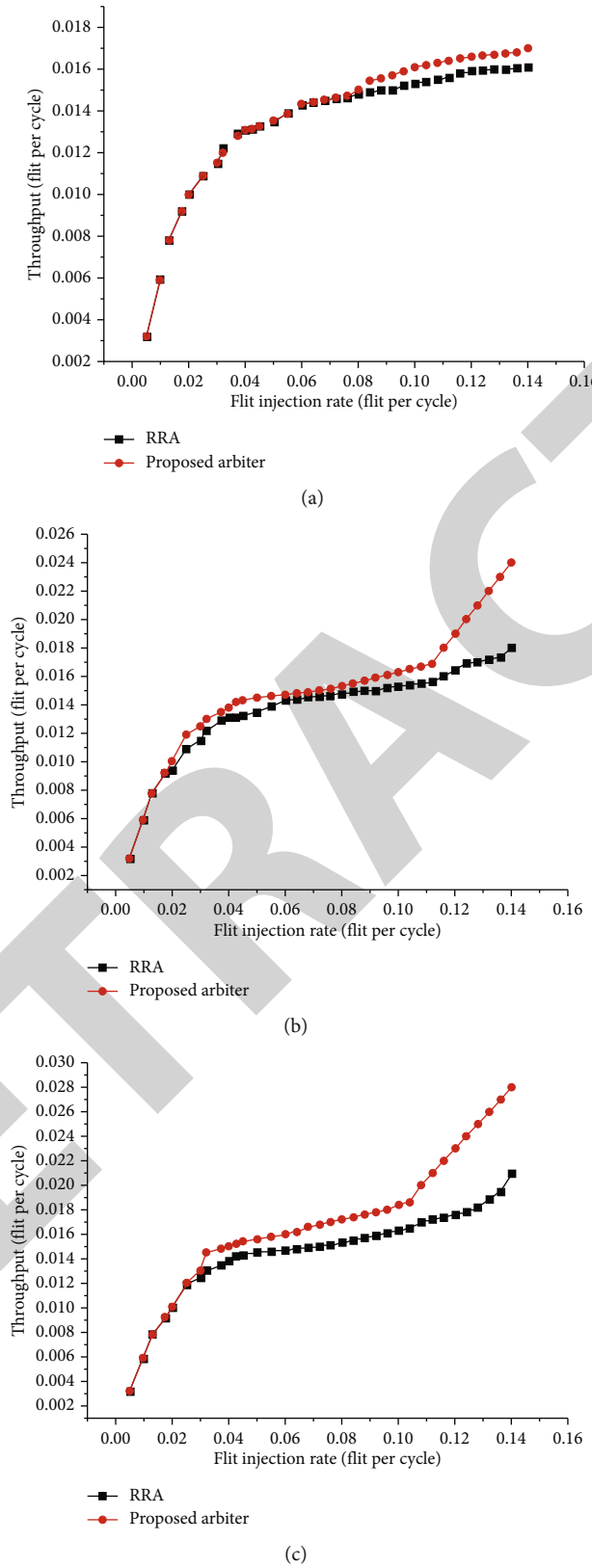

Figure 14: Throughput under synthetic traffic. (a) FIR vs. throughput with two VCs, (b) FIR vs. throughput with four VCs, and (c) FIR vs. throughput with eight VCs, respectively.

mapping of source and destinations by eliminating redundant allocations in steps 4-5. Check the transfer from lower end node to higher in the same process for finding multiple iterations; if all good, enable the transfer in steps 6–8.

Table 1: Flit injection ratio vs. throughput of existing model and proposed model for various test conditions (FIR and Tp are measured in flit per cycle).

URT												ST											
Existing RRA						Proposed arbiter						Existing RRA						Proposed arbiter					
2VCs		4VCs		8VCs		2VCs		4VCs		8VCs		2VCs		4VCs		8VCs		2VCs		4VCs		8VCs	
FIR	*Tp*	*FIR*	*Tp*	*FIR*	*Tp*	*FIR*	*Tp*	*FIR*	*Tp*	*FIR*	*Tp*	*FIR*	*Tp*	*FIR*	*Tp*	*FIR*	*Tp*	*FIR*	*Tp*	*FIR*	*Tp*	*FIR*	*Tp*
0.00	0.0000	0.00	0.0000	0.00	0.0000	0.00	0.0000	0.00	0.0000	0.00	0.0000	0.00	0.0000	0.00	0.0000	0.00	0.0000	0.00	0.0000	0.00	0.0000	0.00	0.0000
0.01	0.0048	0.01	0.0058	0.01	0.0058	0.01	0.0051	0.01	0.0061	0.01	0.0061	0.01	0.0049	0.01	0.0059	0.01	0.0058	0.01	0.0055	0.01	0.0061	0.01	0.0060
0.02	0.0089	0.02	0.0092	0.02	0.0095	0.02	0.0091	0.02	0.0121	0.02	0.0101	0.02	0.0088	0.02	0.0092	0.02	0.0089	0.02	0.0092	0.02	0.0101	0.02	0.0091
0.03	0.0109	0.03	0.0115	0.03	0.0125	0.03	0.0112	0.03	0.0132	0.03	0.0130	0.03	0.0110	0.03	0.0117	0.03	0.0124	0.03	0.0113	0.03	0.0129	0.03	0.0130
0.04	0.0118	0.04	0.0131	0.04	0.0138	0.04	0.0123	0.04	0.0145	0.04	0.0151	0.04	0.0119	0.04	0.0130	0.04	0.0139	0.04	0.0124	0.04	0.0140	0.04	0.0151
0.05	0.0129	0.05	0.0135	0.05	0.0143	0.05	0.0137	0.05	0.0148	0.05	0.0155	0.05	0.0131	0.05	0.0137	0.05	0.0147	0.05	0.0139	0.05	0.0146	0.05	0.0156
0.06	0.0142	0.06	0.0140	0.06	0.0148	0.06	0.0146	0.06	0.0150	0.06	0.0160	0.06	0.0143	0.06	0.0138	0.06	0.0149	0.06	0.0148	0.06	0.0148	0.06	0.0161
0.07	0.0145	0.07	0.0144	0.07	0.0150	0.07	0.0149	0.07	0.0152	0.07	0.0167	0.07	0.0145	0.07	0.0141	0.07	0.0151	0.07	0.0150	0.07	0.0150	0.07	0.0163
0.08	0.0146	0.08	0.0148	0.08	0.0152	0.08	0.0152	0.08	0.0157	0.08	0.0171	0.08	0.0146	0.08	0.0148	0.08	0.0153	0.08	0.0153	0.08	0.0155	0.08	0.0171
0.09	0.0150	0.09	0.0151	0.09	0.0159	0.09	0.0159	0.09	0.0162	0.09	0.0178	0.09	0.0151	0.09	0.0150	0.09	0.0159	0.09	0.0160	0.09	0.0159	0.09	0.0179
0.10	0.0152	0.10	0.0157	0.10	0.0162	0.10	0.0162	0.10	0.0171	0.10	0.0183	0.10	0.0153	0.10	0.0151	0.10	0.0163	0.10	0.0163	0.10	0.0165	0.10	0.0183
0.11	0.0155	0.11	0.0161	0.11	0.0169	0.11	0.0165	0.11	0.0179	0.11	0.0190	0.11	0.0156	0.11	0.0155	0.11	0.0170	0.11	0.0167	0.11	0.0169	0.11	0.0205
0.12	0.0156	0.12	0.0163	0.12	0.0175	0.12	0.0167	0.12	0.0182	0.12	0.0201	0.12	0.0157	0.12	0.0161	0.12	0.0178	0.12	0.0169	0.12	0.0189	0.12	0.0232
0.13	0.0158	0.13	0.0164	0.13	0.0178	0.13	0.0169	0.13	0.0189	0.13	0.0219	0.13	0.0158	0.13	0.0170	0.13	0.0182	0.13	0.0170	0.13	0.0215	0.13	0.0252
0.14	0.0160	0.14	0.0165	0.14	0.0192	0.14	0.0174	0.14	0.0195	0.14	0.0233	0.14	0.0161	0.14	0.0181	0.14	0.0210	0.14	0.0175	0.14	0.0239	0.14	0.0279

Table 2: Feature estimation of arbiter.

Design	Area occupation (μm^2)	Critical delay (ps)
Existing RRA	124.6	456.4
Proposed	109.5	384.2

Depending on the traffic conditions (load) of the preceding router, the flexible priority will be assigned. If a deadlock like conditions of multiple data packets path requests one channel then the priority resolver of arbiter decides the grant to which packets. So flexible priority-based dynamic arbitrations will be implemented. The initial conditions for the resolutions are to read all the requests of input ports that want grant for the same output port and then the arbiter encounters a $K \times K$ matrix developed by flexible priority resolver with bits arranged in a triangular array. The grant will be issued to the requested based on the highest priority resolved matrix to the same output port. The row in the matrix competes input requests and its priorities. The scheduling module will study the priorities of each input port request. The arbiter receives updated within the scheduling matrix when the maximum priority input is served by means of making the request which gets the earlier access. Row and column receive inversed for much less precedence for the next round of arbitration. The simulation end result for the arbitration is shown in Figure 10(c) which gives precedence with apropos to the grant generation. The arbiter is implemented using VHDL in Xilinx Vivado 2021.2.1 and simulated with Vivado-IDE.

4. Throughput Assessment

In this paper, the proposed arbiter is designed on a baseline router of NoC, where the RC, VC, and SA are not much modified. Prior prioritizations greatly reduce the burden of arbiter computation and competition for the allocations through a flexible priority resolver. To maximize the throughput, we used a 3×3 2D mesh. Input ports of all routers have four buffers of 32-bit length. The network is implemented using VHDL in Xilinx Vivado 2020.2 and simulated with Vivado-IDE. To compute the load calculations, we used a URT and synthetic traffic pattern. The throughput is VC number dependent; as the number of VCs increases, the throughput increases as illustrated in Figures 13 and 14 for URT and synthetic traffic, respectively. The prioritized circuit verifies the traffic load in all interfacing directions and preestimates the best suitable path for transmitting the flits without locking them more time at the buffer. First, the system is tested with low flit injection ratio to high flit injection ratio with uniform random traffic (URT) and then with the synthetic traffic, with more than 10% throughput improvement noticed. Table 1 shows the flit injection ratio versus throughput of existing model and the proposed model. Flexible priority resolver usage in the proposed model has larger throughput at higher flit injection ratios with various traffic conditions. It is observed that there is around 3% to 30% throughput increase with respect to low FIR to high FIR.

Table 2 shows the synthesis results of the arbiter in comparison with the existing RRA arbiters, which offers a 12% less occupation of the area and a 15.81% reduction in critical latency.

The future expansions in this model are to work for more throughputs at lower flit injection rates. Further, the area optimization techniques are expected to be implemented for better performance.

5. Conclusion

The main feature of the proposed model is to provide high throughput and efficient on-chip communications for multicore architectures. With the proposed arbiter design, it is observed that the area occupation of buffers and critical delays were considerably reduced; hence, the waiting time of the flits at the arbiter buffers will be less, and the total communication time will decrease. Further, as the priorities are resolved before the flits get granted to the arbiter, the load of the arbiter will significantly reduce, and communication will be efficient. Thus, the proposed model improves the throughput in multicore systems like SoC with an efficient arbiter at NoC. This work has a limitation, i.e., area occupation of the die and productivity (throughput) have no more remarkable improvement at low traffic/load in comparison with the existing models. The proposed arbiter is showing good improvement in the throughput at higher data rates; an average of more than 10% throughput improvement is noticed at higher flit injection rates independent of the VCs implemented. Further, the critical delays are reduced to 15.84% with greater throughputs. Besides, the design proves that at higher flit rates the throughput is increasing considerably. Hence, new hopes are increasing to do further research to accommodate more packets per cycle without compromising the system performance. The future design extensions of the proposed model are expected to handle the exceptions in data transmission over on-chip networks and reduce further latency.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

I thank my working place, IIIT-Bhubaneswar, which provided me with the flexibility and facilities to carry my research during my job as an Assistant Professor of the ETC department. Further, I thank my research bringing university VTU, Belgaum, for permitting me as a researcher. I specially acknowledge my guide Dr. G. Chenchu Krishnaiah and the doctoral committee for giving me timely suggestions.

References

[1] A. Rumyantsev, T. Krupkina, and V. Losev, "Development of a high-speed multi-target measurement system-on-chip," in *2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pp. 1601–1604, Saint Petersburg and Moscow, Russia, 2019.

[2] G. Martin and H. Chang, "System-on-chip design," in *ASICON 2001. 2001 4th International Conference on ASIC Proceedings (Cat. No.01TH8549)*, Shanghai, China, 2001.

[3] W. Liu, W. Zhang, X. Wang, and J. Xu, "Distributed sensor network-on-chip for performance optimization of soft-error-tolerant multiprocessor system-on-chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 4, pp. 1546–1559, 2016.

[4] U. Schlichtmann, "Tomorrow's high-quality SoCs require high-quality embedded memories today," in *Proceedings in Proc SQED*, p. 225, San Jose, CA, USA, 2002.

[5] F. Yazıcı, A. S. Yıldız, A. Yazar, and E. G. Schmidt, "A novel scalable on-chip switch architecture with quality of service support for hardware accelerated cloud data centers," in *2020 IEEE 9th International Conference on Cloud Networking (CloudNet),*, Piscataway, NJ, USA, 2020.

[6] W. S. Chu, C. S. Kim, H. T. Lee et al., "Hybrid manufacturing in micro/nano scale: a review," *International journal of precision engineering and manufacturing-green technology*, vol. 1, no. 1, pp. 75–92, 2014.

[7] W. Wolf, A. A. Jerraya, and G. Martin, "Multiprocessor system-on-chip (MPSoC) technology," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 10, pp. 1701–1713, 2008.

[8] H. I. Gaha and M. Balti, "Novel bi-UWB on-chip antenna for wireless NoC," *Micromachines*, vol. 13, no. 2, p. 231, 2022.

[9] K. Zou, Y. Wang, L. Cheng, S. Qu, H. Li, and X. Li, "CAP: communication-aware automated parallelization for deep learning inference on CMP architectures," *IEEE Transactions on Computers*, vol. 71, no. 7, pp. 1626–1639, 2022.

[10] P. Chen, W. Liu, H. Chen et al., "Reduced worst-case communication latency using single-cycle multihop traversal network-on-chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 7, pp. 1381–1394, 2021.

[11] Y. You, Y. Chang, W. Wu et al., "New paradigm of FPGA-based computational intelligence from surveying the implementation of DNN accelerators," *Design Automation for Embedded Systems*, vol. 26, no. 1, pp. 1–27, 2022.

[12] H. Li, Z. Tian, J. Xu, R. K. Maeda, Z. Wang, and Z. Wang, "Chip-specific power delivery and consumption co-management for process-variation-aware manycore systems using reinforcement learning," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 5, pp. 1150–1163, 2020.

[13] B. S. Feero and P. Pande, "Networks-on-chip in a three-dimensional environment: a performance evaluation," *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 32–45, 2009.

[14] D. Bertozzi, A. Jalabert, Srinivasan Murali et al., "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 2, pp. 113–129, 2005.

[15] A. Psarras, S. Moisidis, C. Nicopoulos, and G. Dimitrakopoulos, "Networks-on-chip with double-data-rate links," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 12, pp. 3103–3114, 2017.

[16] S. Kundu and S. Chattopadhyay, *Network on chip: the next generation of system-on-chip integration*, Proc CRC Press, T & F Group, 2015.

[17] S. D. Oliveira, B. M. Carvalho, and M. E. Kreutz, "Network-on-chip irregular topology optimization for real-time and non-real-time applications," *Micromachines*, vol. 12, no. 10, p. 1196, 2021.

[18] J. Trajkovic, S. Karimi, S. Hangsan, and W. Zhang, "Prediction modeling for application-specific communication architecture design of optical NoC," *ACM Transactions on Embedded Computing Systems*, vol. 21, no. 4, Article ID 35, pp. 1–29, 2022.

[19] A. Chandra and K. Chakrabarty, "A unified approach to reduce SOC test data volume, scan power and testing time," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 22, no. 3, pp. 352–362, 2003.

[20] M. Shrivastava, R. Mehta, S. Gupta et al., "Toward system on chip (SoC) development using FinFET technology: challenges, solutions, process co-development & optimization guidelines," *IEEE Transactions on Electron Devices*, vol. 58, no. 6, pp. 1597–1607, 2011.

[21] S. Kundu and S. Chattopadhyay, *Network on chip: the next generation of system-on-chip integration*, Proc CRC Press, T& F Group, 2015.

[22] B. K. Joardar, R. G. Kim, J. R. Doppa, P. P. Pande, D. Marculescu, and R. Marculescu, "Learning-based application-agnostic 3D NoC design for heterogeneous manycore systems," *IEEE Transactions on Computers*, vol. 68, no. 6, pp. 852–866, 2019.

[23] Y. Gan, H. Guo, and Z. Zhou, "3D NoC low-power mapping optimization based on improved genetic algorithm," *Micromachines*, vol. 12, no. 10, p. 1217, 2021.

[24] S. K. Jena, S. Biswas, and J. K. Deka, "Retesting defective circuits to allow acceptable faults for yield enhancement," *Journal of Electronic Testing*, vol. 37, no. 5-6, pp. 633–652, 2021.

[25] Y. Kameda, S. Yorozu, Y. Hashimoto, H. Terai, A. Fujimaki, and N. Yoshikawa, "Single-flux-quantum (SFQ) circuit design and test of crossbar switch scheduler," *IEEE transactions on applied superconductivity*, vol. 15, no. 2, pp. 423–426, 2005.

[26] Y. Lee, J. M. Jou, and Y. Chen, "A high-speed and decentralized arbiter design for NoC," in *2009 IEEE/ACS International Conference on Computer Systems and Applications*, pp. 350–353, Rabat, Morocco, 2009.

[27] G. Dimitrakopoulos, E. Kalligeros, and K. Galanopoulos, "Merged switch allocation and traversal in network-on-chip switches," *IEEE Transactions on Computers*, vol. 62, no. 10, pp. 2001–2012, 2013.

[28] G. A. Subbarao and P. D. Häfliger, "Design and comparison of synthesizable fair asynchronous arbiter," in *2020 18th IEEE International New Circuits and Systems Conference (NEWCAS)*, pp. 122–125, Montreal, QC, Canada, 2020.

[29] J. Wei, J. Zhang, X. Zhang et al., "An asynchronous AER circuits with rotation priority tree arbiter for neuromorphic hardware with analog neuron," in *2019 IEEE 13th International Conference on ASIC (ASICON)*, Chongqing, China, 2019.

[30] J. Kathuria and M. Sharma, "Data access resolver for IOT enabled SOC interconnections with dynamic programability feature," in *8th International Conference on Signal Processing and Integrated Networks (SPIN)*, pp. 333–338, Noida, India, 2021.

[31] T. Knorst, J. Vicenzi, M. G. Jordan et al., "An energy efficient multi-target binary translator for instruction and data level parallelism exploitation," *Design Automation for Embedded Systems*, vol. 26, no. 1, pp. 55–82, 2022.

[32] H. Zheng, K. Wang, and A. Louri, "Adapt-NoC: a flexible network-on-chip design for heterogeneous manycore architectures," in *IEEE International Symposium on HPCA*, pp. 723–735, Seoul, Korea (South), 2021.