Journal of Scientific & Industrial Research
Vol. 81, May 2022, pp. 495-504

Load Based Dynamic Priority Arbiter for NoC Architecture

Parvathi S* and Umamaheswari S

Department of Information Technology, Anna University, MIT Campus, Chennai, 600 044, India

Received 11 May 2021; revised 31 March 2022; accepted 01 April 2022

The evolution of Very Large Scale Integration (VLSI) and the semiconductor industry have led to the focus on multicore architectures. Network on Chip (NoC) is one of such arrangement which is an interconnection framework comprised of cores, routers, and links. The output port for each request from the input port must be computed, and the output channel must be reserved for the next router. However, the same output port can be requested by more than one input port, but only one request can be granted at a time. Multiple requests for a single output channel will lead to congestion of the packets, thereby increasing the network latency and leading to packet losses. The arbiter selects any one of the input ports and grants permission to use the requested output port while putting the other input port requests to wait. For a congestion-free traversal of packets and to avoid dropping of packets, a Load based Dynamic Priority Arbiter (LDPA) with dynamically changing priorities during run time based on the input port load has been proposed. The proposed customized arbiter LDPA works based on the updates made in the reservations of each input port. The priority of each input port is given according to the average load. More weight is allotted to the highly loaded input ports. By randomization, the chance is given to the lower priority input ports to reduce starvation and hence latency. With the use of the proposed LDPA, the average network latency is reduced by about 15.98% when compared to that of baseline FIFO arbiter, without any compromise in power and throughput.

Keywords: Buffers, Dynamic arbiter, FIFO arbiter, Fixed priority arbiter, Routing, Virtual channels

Introduction

Network on Chip (NoC) is the interconnection scheme for the multi core architectures which replaces the bus-based communication for the connectivity of the cores and the routers. NoC is a packet-based communication framework. A typical NoC structure is comprised of cores, routers, links, and the Network Interfaces (NI).[1] The cores and the routers are connected via the NI. The performance of an NoC depends on the topology and the routing algorithms. There are two kinds of topologies: regular (homogenous) and irregular (heterogeneous) which are categorized based on the size and placement of the cores. Also, there are different kinds of NoC topology structures namely: mesh, torus, folded torus, ring, and fat-free. Regular topologies have cores of the same size, and they are fault tolerant. Irregular topologies have cores of variable sizes, and they are application specific.

The efficient selection of the routing algorithm and the topology will highly help in the improved performance of the multicore processor.

*Author for Correspondence
E-mail: parvathi.sivakumar@gmail.com

In packet-based communication, more than one packet would be requested for the same output port. So, the process of arbitration comes into the picture because of the contention created between the input ports requesting the same output port.[2] Arbitration is the process of selecting one of the contending input ports based on certain criteria and granting permission to get access to the output port. There are many kinds of arbitration techniques. The simple and baseline technique is First In First Out (FIFO) arbitration. The other techniques depend on the traffic pattern of the network, congestion levels of the packets during its traversal, buffers involved, and the virtual channels allocated to the input and output ports.[3]

In priority-based arbitration, priorities are assigned to the contending input ports to access the output port. The two kinds of priority-based arbiters are: Static and Dynamic. In static priority arbiters, fixed priority is assigned to each input port, and it would not change during runtime.

In dynamic arbiters, the priority of the packets would be changing according to the traffic pattern and packet flow. The reservation or allocation of the output channels is done based on the priority mechanism.

Related Work

A dynamic weight arbiter for network-on-chip is designed and implemented based on the Lottery algorithm which is a static arbitration mechanism in this work.[4] According to the congestion experienced, turn weights are employed to dynamically adjust the weights in each direction. The dynamic weight arbiter decreases the overall network latency.

A dynamic priority arbiter, D-L arbiter has been proposed in this work, which can adjust the priority dynamically by detecting loads of input ports in each clock cycle and works based on the lottery mechanism.[5] Overall improvement in communication has been achieved along with the reduction of the buffer resources required under non-uniform traffic patterns.

In this work, Asynchronous Bypass Channels, (ABCs) which is a router-based micro architecture have been proposed. The ABCs provide synchronization at intermediate nodes between the source and the destination and thereby avoiding the synchronization delay.[6] Also a new network topology and routing algorithm has been proposed. Improvement in the performance of a regular synchronizing design has been achieved.

In an ideal network, a single cycle equals the low-load network latency between a source and destination. A router for network-on-chip called Bypass router has been proposed to create a single-cycle data path all the way from the source to the destination.[7] This custom-designed network along with the customized router and routing algorithm is compatible with all topologies and deterministic routing algorithms.

The authors of the work have proposed an arbitration mechanism for NoC to attain a reduction of latency and congestion delay.[8] This arbitration mechanism works for both bypass and baseline pipelines in routers. The simulations are performed for different sized mesh topologies and routing algorithms using Noxim.

An Advance Virtual Channel Reservation (AVCR) has been proposed to reduce contention delay.[9] AVCR is a smart communication service that provides a highway to target packets. AVCR works by predicting the destination of some packets ahead of their arrival at the network interface (NI). Reserving the Virtual Channel (VC) resources thereby achieving a reduction in VC allocation and switch arbitration delay is the process involved.

An analytical modeling technique for priority aware NoCs under bursty traffic has been presented.[10] Bursty traffic model is being frequently used in various applications in place of simple traffic patterns. So bursty traffic is used as a generalized geometric distribution here and the maximum entropy method has been applied to construct analytical models. A decreased error rate has been achieved for two different traffic patterns namely synthetic and bursty.

A new arbitration scheme is proposed for the crossbar switch in wireless routers for the fair allocation of the port priorities.[12] The allocation is performed based on the load. By this, port contention in the wireless routers is avoided and the utility of the output port is improved along with the reduction in network and packet latency.

Authors have proposed a routing method with a combination of two techniques namely: XY and VCT.[13] The performance upgradation of the NoC is achieved. The routers are designed in a way to adopt efficient arbitration schemes. The allocation of the packets is performed based on priority.

A distributed arbitration scheme is proposed by the authors for dynamic CDMA-bus-based systems.[14] The single arbitration unit is decomposed into many simple elements. The elements are connected via a ring. The ring executes an algorithm for solving the conflicts that arise in the destination. Code words are assigned to the processing elements.

Materials and Methods

Static and Dynamic Arbiters

Arbiter is used whenever there is a contention between the input ports that is when more than one input port is requesting for the same output port. The arbiter must give the grant based on the arbitration logic and the input port having the higher priority based on the packet injection rate. Static priority arbiter is also called a Fixed Priority Arbiter (FPA). In FPA, the priority assigned to the input port is fixed. Priority does not change from cycle to cycle.[11] The lower the number, the higher is the priority given to that port and the packet from the highest priority input port traverses first. Only one requester must be granted for the same resource.

An FPA is shown in Fig. 1, with the priorities of each port mentioned as values ranging from 0 to 4. The input port requests signals and the output port grant signals of all the ports of the NoC framework are indicated. The allocations are based on the routers. Each router has links in E, W, N, S, and Local. Here East port is given the highest priority: with the priority value, 0 and Local port has the least priority:

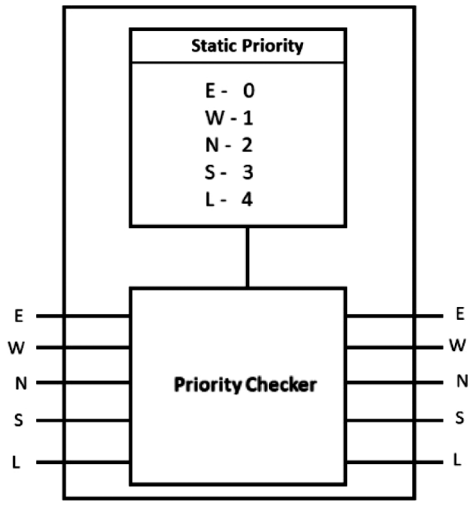

Fig. 1 — Fixed Priority Arbiter (FPA)

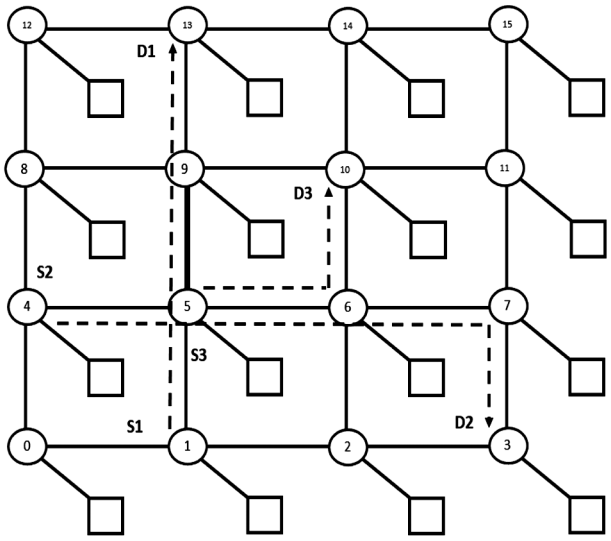

Fig. 2 — 4 x 4 NoC with contention at node 5

with value 4. A priority checker involves in the checking of the priority values assigned to the input ports and giving grants to the ports with the highest priority and allowing them to access the output ports.

An example flow diagram of packets in a 4 × 4 mesh topology is shown in Fig. 2, based on the data of source and destination given in Table 1. Three source-destination pairs are given: 1-13, 4-3, and 5-10. Here the packets moving from 5 to 10 and from 4 to 3 are moving through router 5. The priority values for all these pairs are mentioned in Table 1. These packets are requesting for the same east output port, thereby leading to the contention of packets. Hence the process of arbitration comes into the picture in this scenario. According to the priority-based arbitration,

Table 1 — Priority of input and output ports

Source ID	Destination ID	Input port	Output port	Priority
S1: 1	D1: 13	South	North	2
S2: 4	D2: 3	West	East	1
S3: 5	D3: 10	Local PE	East	3

the higher priority packet, 4 to 3 wins the arbitration and the packets can be allowed to the forwarding phase and the packet from 5 to 10 is buffered as it has low priority.[12,13]

In a dynamic arbiter, the priority can be changed for each flow. A one-hot priority signal p is used to select the highest priority request.[14] Dynamic arbiter works based on arbiter logic and a priority updater that generates the next priority. The next priority specifies the highest priority port in the next cycle.

There are two types of variable priority arbiter namely: oblivious arbiter and round-robin arbiter. In oblivious arbiter, the next priority chosen for the next cycle does not depend on the request and grant of the port in the current cycle. In a round-robin arbiter, the next priority chosen for the next cycle depends on the request and grant of the port in the current cycle.[15,16]

Load based Dynamic Priority Arbiter (LDPA)

Motivation for Customized Dynamic Priority Arbiter

The objective of this work is to improve the arbitration process involved in the network-on-chip architecture. This is performed by making changes in the way the reservations are made for each input port. The priority to each input port is given according to their average load. The round-robin arbiter provides an equal number of chances to every input port to provide strong fairness but that does not lead to better network performance in terms of latency and throughput. Hence, a Load based Dynamic Priority Arbiter (LDPA) has been proposed. In LDPA, the priority is assigned and varied dynamically according to the load at runtime. According to the priority logic, more weight is given to the higher load priority input ports. Due to randomization, the chance is not only given to higher priority ones, and this eventually reduces the waiting time and hence the network latency thereby improving the network performance.

Process Flow of the Entire Arbitration Process of the LDPA

The entire working flow of the LDPA's arbitration process is given in detail in this section. How a flit moves from one node to another, and the overall process flow from the input port requests till the grant for the output port access is explained. These processes happen in the various modules of the arbiter.

A packet or flit is received from the upward node to the downward node thereby the flow of incoming flits is monitored. To accept the flit into the downward node, there should be an incoming request and a free slot should be available in the input buffer.

The process of transmitting the flit that has been received by the input channel to the desired output channel of the crossbar switch is performed in two phases: reservation phase and forwarding phase. The process of arbitration is done in the reservation phase, and after arbitration, the movement of the flit to the switch traversal mode is carried out in the forwarding phase.

Each flit undergoes a push or pop operation. The transmitting phase pops the front flit one by one from every input port and every virtual channel from each input port and checks whether the popped flit is a head flit. The reservation phase uses the information in the head flit. Head flit contains the source and destination IDs. Output port is identified from the destination ID.

From the route function, the request from the input port to the specific output port is stored in the reservation table of the corresponding output port. The information of requesting the input port and its virtual channel is stored in the reservation table of that output port. If more than one input port request exists for the same output port, the function: Check Reservation of the proposed LDPA is called for.

The reservation table of the west output port is shown in Table 2. The reservation table consists of the information of requests from the other four directions input ports, namely N, E, S, and, local to access the west output port. The table also saves the information from which virtual channel (0, 1 or 2) the request is given. The request bit is set if a request is made from a particular input port. This request bit is reset when the requested input port is granted to access the west output port.

The percentage load of each buffer i.e., the buffer occupancy is identified by the tracking process of the total number of flits in the individual buffer. The number of flits gets updated by the push and pop operation of the flits. The load percentage updating is also performed in the Load identification module. This is calculated by the ratio of the total number of flits in the buffer at that instant to the total number of flits that the buffer can hold.

Table 2 — Reservation table for west output port request

Input port	Input virtual channel	Request bit
N (North)	00	1
E (East)	01	1
S (South)	xx	0
L (Local)	01	1

After every update of the buffer, the value is returned to the requesting input port. There are three virtual channels in each input and individual values of the percentage are summed up to arrive at the average buffer load for that input port. The same process is followed for the calculation of all the requesting input ports. This is sorted out in descending order: the higher one being first in the order in the Load identification module.

According to the sorted descending order of the average load of the requesting input ports, a ratio gets generated. For example, if there are three requests, the top one will be assigned the highest priority and the ratio as 3, the next one will be assigned 2 and the last one is assigned with a ratio of 1. This is the process happening in the Ratio generation module.

The total number of tickets is fixed. According to the ratio generated in the above module, the total number of tickets is divided and given to every requesting input port. For instance, let the total number of tickets ‘n’ be 100 and the number of requesting input ports is 3, then the one with the ratio of 3 will get 52 tickets, the next with the ratio of 2 will get 32 tickets and the last one with ratio 1 will get 16 tickets.

The interval is generated based on the values of the ratio generation. For the same example of 100 tickets in the ratio 3:2:1 with 52, 32, and 16 tickets respectively, in the interval would be generated as 0–51 for the first input port, 52–83 for the second input port, and 84–99 for the last input port.

Then, a random number ‘p’ that lies between 0 and 99 for 100 tickets is generated by the linear feedback shift register technique in Random Number Generation Module. Based on the interval in which the random number falls, that input port is granted access to the output port.

The flits from that input port are moved to the forwarding phase. If for an instant the random number ‘p’ is 87: then it lies in the interval 84–99 and it infers that the last input port has won the arbitration process and those flits are moved to the forwarding phase.

Algorithm for the Proposed LDPA

1. *Storing the number of flits as a variable.*
2. *Increment the variable with push operation, and decrement with pop operation to keep track of how much buffer is used.*

3. *The front flit is popped up one by one from every input port and checked whether it is a head flit. This is done in the transmitting phase.*
4. *Head flit contains information of source and destination IDs.*
5. *Reservation table of an output port stores the requesting input port and its virtual channel information. Request bit is set whenever there is a request.*
6. *The input port requests are identified from the information of the reservation table and the output port is identified from the destination ID.*
7. *Percentage load of each buffer is calculated by the ratio of total number of flits in the buffer at that instant to the total number of flits that the buffer can hold.*
8. *There are three virtual channels in each input and individual values of the percentage load are summed up to arrive at the average buffer load for that input port.*
9. *The updated load values are sorted out in descending order.*
10. *Ratio is generated based on average load and the highest priority is given to the input port with highest average load.*
11. *Interval generation for allotting the number of tickets to the requesting input ports is performed according to the generated ratios.*
12. *Random number is generated in the range of the intervals generated.*
13. *Based on the interval in which the random number falls, that input port is given access to the output port.*
14. *The request bit is reset once the input port is given the grant and flits from that input port are moved to the forwarding phase.*

The signal from the checkReservation function will be sent as output. If the input port calling the check Reservation function equals the input port that has won the arbitration inside the function, then the available (grant) signal is sent back to that input port and that input port flits are moved to the forwarding phase. Else a busy signal is sent back along with the details of the input port which has won the arbitration. The reservation table of a particular output port has the information of the input port which has won the arbitration for the output port.

Then these flits are traversed from the upward node to the downward node until the tail flit reaches the output port. Till this time, the links are occupied by the flits of the input port which has won the arbitration by the concept of the proposed LDPA.

Arbitration Process Flow for a West Output Port

The architecture of the Load based Dynamic Priority Arbiter (LDPA) is illustrated in Fig. 3, for the request of a west output port.

Based on the overall arbitration process flow explained above, the working flow for a specific output port, here west output port, is explained in this section.

In general, there would be 5 input ports: North (N), South (S), East (E), West (W), and Local in a node

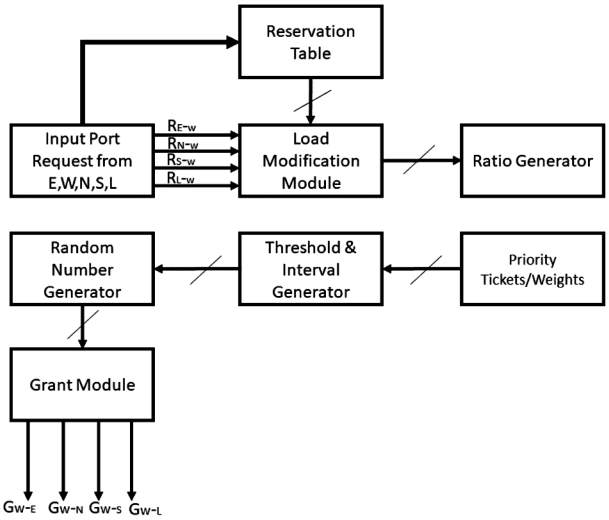

Fig. 3 — Architecture for the Load based Dynamic Priority Arbiter (LDPA) for west output port

and each of the input ports has 3 virtual channels. So, at a time any of the input ports can make a request to the output ports.[17] The input ports requesting for the west output port would be from the other four directions E, N, S, and Local and the request signals are $R_{E\text{-}W}$, $R_{N\text{-}W}$, $R_{S\text{-}W}$, and $R_{L\text{-}W}$ respectively as shown in Fig 3. The tracking process of the buffer details of all the input ports helps to identify the traffic or load in each input port. Based on the tracking process, the priority-based mechanism can be applied to attain reduced network latency without affecting the performance of the NoC.

The priority is given here according to the load in each requesting input port. The input port requests for the west output port are updated in a reservation table in the form of vector information. The priority can be assigned to the input port requests according to their average load. The larger the average load, the higher priority is assigned as the ticket or weight.

The ticket value is incremented or decremented according to the flit arrival by push and pop operation which is done dynamically during the run time according to the traffic pattern. This is termed as the individual buffer load.

At the end of this operation, the reservation table will contain information about the input port requests and input port virtual channel. It is given as input to the load identification module. Here average load of each input port is compared on a percentage basis and the priority is assigned on a ratio basis. The random number generator generates a random number p.

The grant module generates the grant signals to access the output port. The input port which has won the switch arbitration stage is moved to the forwarding stage. The procedure is performed in each node or router and priority is assigned and changed dynamically during runtime whenever there is an input port requesting for an output port.

Based on the load and priority of the input ports the grant signals are assigned to the winning input port to access the output port. The four grant signals from the west output port are GW-E, GW-N, GW-S, and GW-L. These are the grants of the west output port for requests from the east input port, north input port, south input port, and local, respectively.

Experimental Evaluation and Results

The work has been entirely carried out using an NoC simulator called Noxim.[18] Noxim is an open, configurable, extendible, cycle-accurate NoC simulator developed in System C which allows to analyze the performance and energy figures of various NoC architectures. The Noxim simulator is developed using System C, an extension of the system description library written in C++. An NoC instance is simulated using System C code in the Noxim Runtime Engine (NRE).

The NRE supports various kinds of NoC topologies, different buffer sizes, variable packet injection rates, and adaptive routing algorithms.

Upon execution of various tasks at the end of the Simulation, Noxim generates performance parameters in terms of latency, throughput, and energy consumption. The simulation is carried out for 10,000 cycles.

Results with Synthetic Traffic Pattern

Network latency, throughput, and energy consumption of NoC with the two static priority arbiters, namely FIFO and FPA, and two dynamic arbiters, namely RR and proposed LDPA are measured.

Average latency by varying the packet injection rate for 4 × 4 NoC with random traffic pattern is presented in Table 3.

The average latency is calculated using Eq. (1)

$$Lavg = \forall i \sum Li/Ni \quad \ldots (1)$$

where, L_{avg} is the average latency,

L_i is the latency when the packet injection rate is 'i'

N_i is the number of packet injection rate entries.

The average latency computed using Eq. (1) is presented in Table 3.

Average Latency Reduction (RiL_{avg}) for Arbiter 'X' in comparison with FIFO arbiter can be computed using Eq. (2).

$$RiLavg(Arbiter\ X) = \frac{Lavg\ (FIFO) - Lavg\ (Arbiter\ X)}{Lavg(FIFO)} \quad \ldots (2)$$

Average latency reduction in percentage for FPA, RR, and proposed LDPA in comparison with FIFO arbiter with random traffic pattern for a 4 × 4 NoC is calculated using Eq. (2).

The network latency in cycles is measured for various packet injection. The values are 3.03, 8.01, and 15.98 respectively. The average latency can be computed in the same way using Eq. (2) for all the traffic patterns with different topologies. The network latency in cycles is measured for various packet injection rates for 4 × 4

Table 3 — Average Latency of 4 × 4 NoC with random traffic pattern

Packet injection rate	FIFO	FPA	RR	LDPA
0.005	8.83	8.55	8.89	8.55
0.006	9.35	9.18	9.23	9.14
0.007	9.33	9.20	9.62	9.56
0.008	9.88	9.93	9.47	9.19
0.009	10.38	9.82	10.31	10.12
0.01	10.44	10.63	10.92	10.81
0.02	18.72	17.90	18.34	18.01
0.03	98.12	95.03	69.78	56.02
0.04	1003.37	1095.61	900.88	1068.98
0.05	2130.21	1981.33	1873.12	1808.53
0.06	2795.86	2628.59	2591.3	2331.61
0.07	3291.61	3195.49	3000	2818.63
0.08	3720.40	3620.13	3567.2	3059.14
0.09	4004.19	3876.16	3600.3	3334.51
0.1	4247.62	4154.22	3978.15	3400.45
Average latency	1424.55	1381.45	1310.51	1196.88

NoC, 8 × 8 NoC, and 16 × 16 NoC topologies for all the arbiters: FIFO, FPA, RR, and proposed LDPA with three synthetic traffic patterns: random, transpose, and butterfly. The network latency for a 4 × 4 NoC is presented in Fig. 4.

The percentage reduction of the average latency for FPA, RR, proposed LDPA in comparison with FIFO arbiter is presented in Table 4

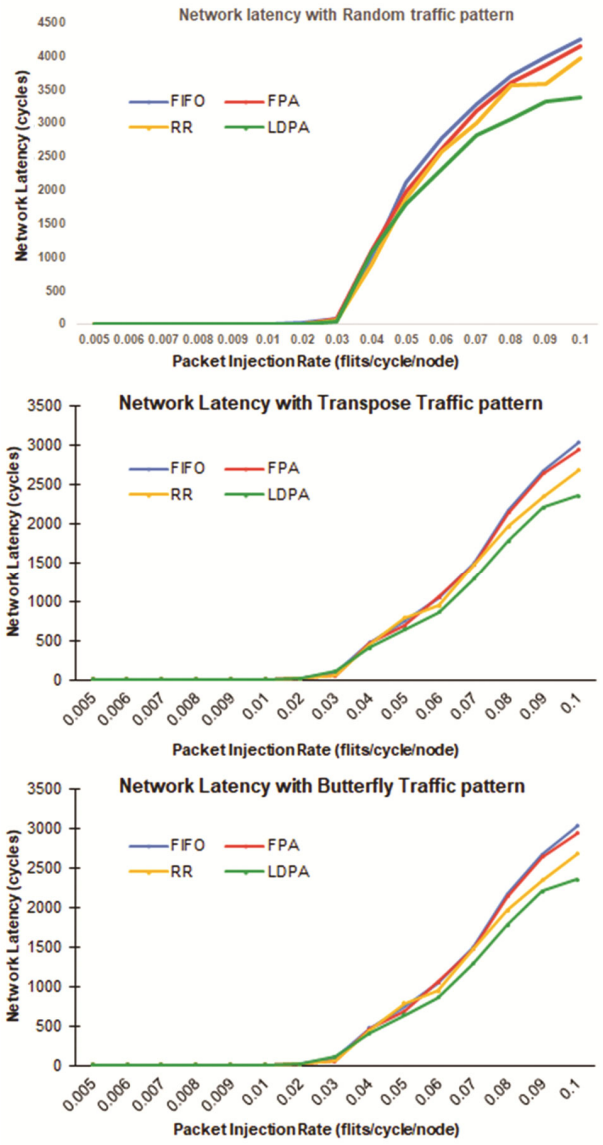

Fig. 4 — Network latency for 4 × 4 NoC

The results of network latency for all the mentioned traffic patterns, arbiters, for 4 × 4 NoC is given in Fig. 4.

The average throughput in flits per cycle has been presented for the 4 × 4, 8 × 8, and 16 × 16 NoC topologies in Fig. 5 for random, transpose, and butterfly traffic patterns for the four arbiters, respectively.

The average energy consumption in micro-joules is presented for all the arbiters with the three traffic patterns in Table 5.

Result for VOPD Application

VOPD – Video Object Plane Decoder is a real-time application. Our proposed LDPA can be used in the VOPD application. The objective of latency reduction

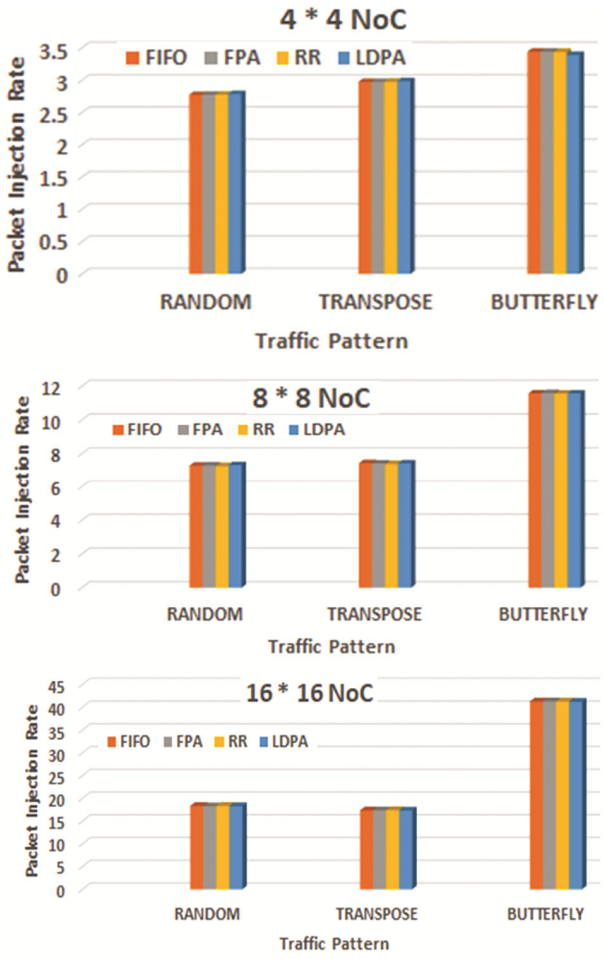

Fig. 5 — Average throughput

Table 4 —Average latency reduction in percentage for various arbiters in comparison with FIFO arbiter

	4 × 4 NoC			8 × 8 NoC			16 × 16 NoC		
	Random	Transpose	Butterfly	Random	Transpose	Butterfly	Random	Transpose	Butterfly
FPA	3.02	1.34	2.36	2.65	2.88	2.49	1.88	1.14	1.54
RR	8.00	1.91	8.88	8.5	5.98	7.72	9.45	4.64	8.62
LDPA	15.98	4.53	14.69	15.58	7.89	13.73	15.25	7.28	14.30

Table 5 — Average energy consumption (micro-joules) for various arbiters with different traffic patterns

	4 × 4 NoC			8 × 8 NoC			16 × 16 NoC		
	Random	Transpose	Butterfly	Random	Transpose	Butterfly	Random	Transpose	Butterfly
FIFO	2.392	2.961	2.328	5.929	5.899	5.845	9.666	9.408	9.325
FPA	2.393	2.360	2.329	5.930	5.898	5.840	9.652	9.412	9.325
RR	2.394	2.957	2.328	5.926	5.898	5.845	9.665	9.412	9.324
LDPA	2.386	2.355	2.327	5.930	5.902	5.843	9.661	9.405	9.324

Table 6 — Latency Results for VOPD application

ARBITERS	NETWORK LATENCY (cycles)
FIFO	3749.3
FPA	3771.4
RR	3815.3
Proposed LDPA	3625.4

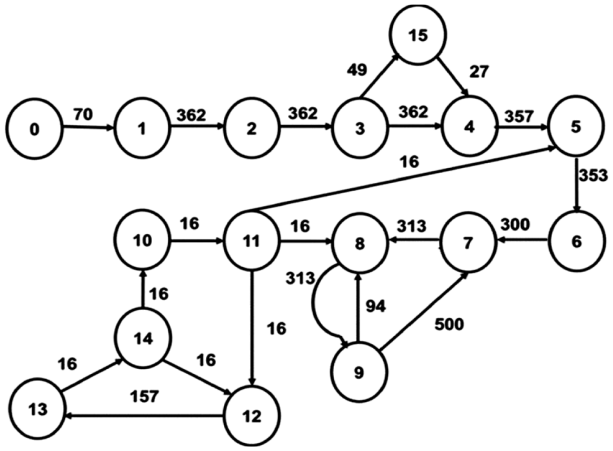

Fig. 6 — Core flow graph of VOPD application

using LDPA has been attained in the real-time application, VOPD, without much change in energy consumption and throughput. The Core Flow Graph (CFG) for the VOPD application is shown in Fig. 6. for which the latency reduction has been achieved.

The values of the network latency measured in 4 × 4 NoC mapped with VOPD application with static and dynamic arbiters in Noxim Simulator are given in Table 6.

Discussion

In this proposed work, the performance measures for latency, throughput, and energy consumption are done through the Noxim simulator for different mesh sizes.

The average energy consumption of the various arbiters is given in Table 5. There are not many changes in the values of the energy consumption in micro-joules for the various arbiters with the three traffic patterns.

The percentage reduction of the average latency for FPA, RR, LDPA in comparison with FIFO arbiter is presented in Table 4.

For the 4 × 4 NoC, the percentage reduction of latency of the proposed LDPA with respect to FIFO is 15.98% for random traffic, 4.3% for transpose traffic, and 14.69% for butterfly traffic pattern.

The percentage reduction of latency of the LDPA with respect to FIFO for 8 × 8 NoC is 15.58% for random traffic, 7.89% for transpose traffic, and 13.73% for butterfly traffic pattern.

In the case of 16 × 16 NoC, the latency reduction achieved by the LDPA compared to FIFO is 15.25 % for random traffic, 7.28% for transpose traffic, and 14.30% for butterfly traffic pattern.

From the above analysis, latency reduction has been attained with all three kinds of synthetic traffic patterns: random, transpose, and butterfly with the proposed arbiter, LDPA. At the same time, the maximum percentage reduction of latency for the LDPA has been achieved for the random traffic pattern with a value of 15.98%.

In FIFO arbiter, if a flow has a higher traffic rate and requests output port may starve for a long time. This may lead to congestion in the network.

In the case of the Fixed Priority Arbiter, the highest demanding flow may be assigned the least priority and it may starve. Hence that flow blocks the buffer slots and further increases the average network latency.

In Round Robin (RR) arbiter, the higher priority might be given to the path with less traffic flow or no traffic flow that is an idle path. Hence, the latency of the highest demanding flow increases, and in turn, average network latency increases.

In the cyclic pattern of arbitration, each port must be given access at least once in a cycle. This arbitration mechanism would be suitable for a uniform traffic pattern. It is not suitable for a non-uniform traffic pattern such as a random traffic pattern.

In the case of our proposed LDPA, the tracking process takes place to find the path having the

maximum traffic flow, which is the highest demanding flow, by computing the average buffer load. This load is updated for each buffer of each input port contesting for the same output port. These load values are compared, and a higher priority is given to the port having a higher load.

The ports having lower traffic load or flow would be waiting for a longer time. This also increases the starvation delay. To reduce starvation, a ticket system and randomization in selection is used in the proposed LDPA.

The total number of tickets is divided and allotted to each flow based on their priority. Higher priority flow gets a higher number of tickets.

Then a random number is generated to choose one of the tickets. The flow which is having that ticket is the winner of the arbitration process and the corresponding input port is given the grant to access the output port.

As a higher number of tickets is allotted to a higher priority flow, the chance of winning is also higher. At the same time, lower priority flows can also win due to randomization, and hence starvation is reduced.

In this way in our proposed arbiter LDPA, reduces average network latency by giving higher priority to higher traffic flows and reduces starvation when compared to the static arbiters: First In First Out (FIFO), Fixed Priority Arbiter (FPA), and the dynamic Round Robin (RR) arbiter.

Our proposed LDPA has given a good improvement in latency for three kinds of traffic patterns: random, transpose, and butterfly in three different topologies, namely, 4 × 4, 8 × 8, and 16 × 16 NoCs as shown in Table 4.

In many real-time applications, the traffic pattern is random in nature. The proposed customized arbiter, LDPA performs better for the random traffic pattern and hence suits better for real-time applications. From the results, it is also noted that the latency reduction is achieved for VOPD, a real-time application without any compromise in the throughput. The throughput of all the NoCs with the various traffic patterns has been maintained while attaining the latency reduction. In the same way, the energy consumption with all the arbiters: FIFO, FPA, RR, and the proposed arbiter LDPA, with all synthetic traffic patterns, have also been maintained.

Conclusions

In this work, a customized Load based Dynamic Priority Arbiter (LDPA) has been proposed and its performance parameters have been compared with three other kinds of arbitration processes namely, FIFO, FPA, and RR arbiters. The proposed LDPA gives better performance in terms of latency reduction of up to 15.98% with random traffic pattern while maintaining the energy consumption and throughput. This has been achieved by prioritizing the input ports dynamically based on the traffic load and the randomization process applied in the port selection for the traffic flow of the packets. Although the randomization process reduces the starvation of the low priority input ports, it may lead to higher priority flows to wait for a longer time. Our future work is to identify the port utilization in the high traffic flow pattern and to identify the starvation delay details based on the waiting cycles.

References

1 Hennessey J L & Patterson D A, *Computer Architecture–A Quantitative Approach* (Morgan Kaufmann Elsevier) 2012.

2 Fu Z & Ling X, The design and implementation of arbiters for Network-on-chips, *2nd Int Conf Ind Informat Syst*, 2010, 292–295.

3 Noghondar A F & Reshadi M, A low-cost and latency bypass channel based on-chipnet work, *J Super Comput*, **71** (2015) 3770–3786.

4 Xu Z, Zhang S, Ni W, Yang Y & Bu J, Design and Implementation of a Dynamic Weight Arbiter for Networks-on-Chip, *4th IEEE Int Conf Sci Technol ICST* (IEEE) 2014, 354–357.

5 Wang J, Li Y, Peng Q & Tan T, A Dynamic Priority Arbiter for Network-on-Chip, *Int Symp Ind Embed Syst* (IEEE) 2009, 253–256.

6 Jain T N, Gratz P V, Sprintson A & Choi G, Asynchronous bypass channels: improving performance for multi-synchronous NoCs, *Fourth Int Symp Netw-on-Chip* (ACM/IEEE) 2010, 51–58.

7 Noghondar A F, Reshadi M & Bagherzadeh N, Reducing bypass-based network-on-chip latency using priority mechanism, *IET Comput Digit Tech*, **12(1)** (2018) 1–8.

8 Wang B & Lu Z, Advance Virtual Channel Reservation, *Design*, *Automat Test Europe Conf Exhibit (DATE)* 2019, 1178–1183.

9 Mandal K M, Ayoub R, Kishinevsky M, Islam M M & Ogras U Y, Analytical performance modeling of NoC sunder priority arbitration and bursty traffic, *IEEE Embed Syst Lett*, **13(3)** (2021).

10 Chang K C, Liao I M & Shiu B Y, Design and implementation of a NoC Supporting priority-based communications for many-core SoCs, *Int Comput Symp* 2010, 483–488.

11 Surumi A & Suranya G, A modified NoC router architecture with fixed priority arbiter, *Int J Sci Res*, **4(10)** (2015).

12 Rad F, Reshadi M & Khademzadeh A, A novel arbitration mechanism for crossbar switch in wireless network-on-chip, *Cluster Comput*, **24** (2021) 1185–1198.

13 Prasannamariya L & Priyadarshini R, An efficient arbitration technique for NOC Using XY and VCT Routing, *Int J innov Res Technol*, **6(12)** 2020.

14 Nikolic T R, Nikolic G S & Djordjevic G L, "Distributed arbitration scheme for on-chip CDMA bus with dynamic codeword assignment, *ETRIJ*, **43(3) (**2021) 471–482.

15 Helal K A, Attia S, Ismail T & Mostafa H, Priority-select arbiter: An efficient round-robin arbiter, *Ann IEEE Northeast Workshop on Circuits and Systems (NEWCAS*) (IEEE) 2015, 1–4.

16 Chan C H, Tsai K L, Lai F & Tsai S H, A Priority based Output Arbiter for NoC Router, *IEEE Int Symp Circuits Syst (ISCAS)* (IEEE) 2011, 1928–1931.

17 Vinoth K M & Senthil K M, Design and Implementation of Router Arbitration in Network on Chip, *Int J Eng Res Technol*, **3(2)** 2014.

18 Catania V, Mineo A, Monteleone S, Palesi M & Patti D, Noxim: An open, extensible and cycle-accurate network on chip simulator, *IEEE 26th Int Conf Appl-Specif Syst Arch (ASAP)* (IEEE) 2015, 162–163