Received: April 26, 2022. Revised: June 15, 2022.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Interval Based Transaction Record Keeping Mechanism for Adaptive 3D Network-on-Chip Routing

Muhammad Kaleem[1,2]* **Ismail Fauzi Bin Isnin[1]**

[1]*School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia, Johor Bahru, Malaysia*
[2]*Department of Computer Science & Information Technology, University of Sargodha, Sargodha, Pakistan*
* Corresponding author's Email: kaleem.muhammad@graduate.utm.my

Abstract: Due to technology scaling, network-on-chip (NoC) become the viable solution for on-chip many-core systems. The most critical concern of NoC is congestion management caused due to heavy communication traffic between nodes. Without an appropriate congestion resolution strategy for reducing heavy in-network traffic, the efficiency of the entire network is damaged severely. In this paper, an interval based record-keeping mechanism is presented to reduce network traffic and congestion by maintaining a history table and previous packet transaction records at each node. Proposed method performs certain validity checks before allowing using previous transaction record from history table. The performance of the technique is investigated in terms of average delay and compared to the state-of-the-art routing algorithms using the Access Noxim simulator. The simulation results demonstrate that the proposed method has outperformed in terms of global average delay, with 8-12% improvement, the average number of hits is 26-61% greater than misses under different synthetic traffic. The proposed algorithm has been tested under various topological configurations for efficiency evaluations.

Keywords: Congestion-aware, Network-on-chip, Routing algorithms, Adaptive routing algorithms.

1. Introduction

System-on-chip (SoC) is an emerging technology containing many processing elements on a single chip. Most SoC bus system contains dedicated signal wires for communication. A new paradigm that helps to facilitate SoC limitations is network-on-chip (NoC). Hundreds of microprocessors can communicate by using NoC infrastructure [1]. NoC contains a common set of wires used to carry similar signals. Packets can simultaneously use different links of NoC to achieve a high level of parallelism hence, the complexity of integrated circuits increases. NoC also enhances performance in the form of throughput, latency and scalability compared to traditional interconnection architectures. However, thermal aggravation [2], transient and permanent defects [3], aging [4], high power density [5], and congestion [6] are some of the variables that contribute to NoC performance deterioration.

In order to utilize NoC's potential benefits and achieve a high level of parallelism, a route discovery algorithm must be devised and implemented. Unlike traditional computer communication networks, packets in NoC are of great significance and they are not permitted to be dropped even in the presence of congested regions [7]. This characteristic is known as lossless. As a result, the most critical concern in the NoC is congestion management; lacking appropriate solutions for congestion situations, the efficacy of the portion or entire network may be severely damaged [8]. Congestion on the path between source and destination is also known as In-network congestion [9]. In-network various solutions have been proposed such as regulating packet arrival or departure and by keeping the record of previous iterations and predictions [10]. To address in-network congestion, several approaches such as altering the design of routers and buffers [11], isolating traffic flows [12], employing adaptive congestion-aware routing algorithms, and so on have been proposed.

Congestion-aware adaptive routing can determine the allowed output port for each packet. Path selection approaches are utilized at each hop to pick the appropriate output port depending on network status, as well as ways for spreading congestion information to each router to keep each router fully informed about the current condition of the network.

In this work proposed strategy addresses in-network congestion along with ability to maintain status of nearby routers for future use. A record keeping mechanism is presented to maintain localized history tables after each transaction at every node. Four parameters are recorded in the history table: incoming packet direction, target packet direction, calculated cost and sampling time. If an entry is found in the table and it is valid in terms of time lapsed, and also the latest cost of the neighbour is within the acceptable range, then a packet will be sent to that neighbour and updating the history table for future transactions.

Following are the contributions of this work.

1. An interval based record keeping mechanism (IBRKM) addresses in-network congestion along with the ability to reduce communication overhead for transfer of vital stats of nearby routers for future use is proposed.
2. IBRKM works on a history table record of transactions carried through the node under validity conditions.
3. Extensive simulations were performed with different synthetic traffics to compare the results with existing techniques.

The organization of the paper is as follows. Related work on history based congestion-aware routing techniques is presented in Section 2. Methodology for the proposed routing approach has been explained in detailed Section 3. Simulation setup along with results and discussion has been expressed in Section 4. In Section 5, this work has been concluded.

2. Related work

Recent history-based congestion-aware NoC routing algorithms are briefly explained and critically analysed in this section. During idle cycles, congestion information propagates on conventional data links in the link-sharing method [13]. Compared to its counterparts, the link-sharing method provides a fast and more comprehensive network status overview to each router. This allows making more intelligent routing decisions to avoid congestion regions. Better routing decisions may be made when more exact and detailed network congestion information is obtained. To discover less congested paths, the routing method in [6] employs two routing tables. One table records directions depending on propagation delay, while the other records queuing delays of each router port. Whereas, the presence of a bottleneck is indicated by the queuing delay.

To balance traffic distribution and meet efficiency criteria, [14] provides a fully adaptive routing technique with energy and buffer awareness.

A model feature network state is suggested to alleviate network congestion. It considers both historical and present network situations for decision making. Fully adaptive routing improves performance by reducing network traffic congestion and fulfilling standards of high priority packet performance criteria. Last-level cache (LLC) [15] is a congestion control approach in which the NoC router is equipped with modest memory space to hold instances of heavily used cache blocks. However, non-negligible hardware overhead is observed.

An adaptive algorithm for reducing the transferring packet size by sending the calculated differences between packets [16]. On the bases of the data localization and error tolerance, this technique decreases traffic volume in NoCs without sacrificing substantial quality in the application output. It can only propagate limited congestion information. When a packet is generated, the algorithm presented in [10], stores a recommended route in the header of the packet based on betweenness centrality, prior packet history routes, and adaptively degree. Packets can proceed independently on the path until it hits a severe congestion situation. The congestion-aware adaptive routing algorithm will then execute and define the packet's next neighbour. The route can be altered as a result of adaptive routing. However route can be altered for the fixed number of times. At reaching a severe congestion situation, there may not be too many possibilities to overcome congestion.

2D adaptive odd-even routing algorithm extended to 3D adaptive odd-even routing presented in [17]. According to the 3D OE (three-dimensional Odd-Even) routing method, few turns such as Up-xy are not permitted in every even layer, whereas Xy-Down are not permitted in every odd layer. This feature, on the other side, limits total adaptability and affects system performance even further.

The Q-learning mechanism presented in [11], suggested a feedback-based proactive thermal management method. An agent learns from its own action during system activity in a simulation environment. The reward values for agents are recorded and updated in the table located in the router also known as Q-table. Hence it does not require learning packets to distribute data over the chip. Packets are detoured from high-risk zones as

rapidly as possible based on Q-table values. Q-learning-based routing, on the other hand, is based on average temperatures, which are always lower than the router's peak temperature, resulting in poor routing options.

To improve overall node utilization Q-learning based adaptive routing is presented in [18], focusing on balancing inter-layer traffic distribution. It also offers an extensive congestion investigation to reduce performance degradation. It discovers regional congestion and thermal hotspots by distributing traffic of overheated regions in a layer and continually learning as networks expand. In order to make an effective routing decision, it employs a Q learning-based selection of routes for the packets. To keep record of the status Q-table is employed in each router. Propagation of information about congestion situation is slow along with high update traffic.

A collaborative thermal-aware adaptive routing (CTTAR) strategy [19] is presented for synchronizing network traffic and temperature information. Due to unnecessary packet switching in routers, hotspots are produced. Dynamic buffer change is employed in CTTAR. To decrease the pace of temperature rise, routing restrictions are applied based on expected thermal information around potential overheated zones. Heat production and dispersion will be limited by the dynamic buffer update. Thermal regions are converted to congestion areas by CTTAR. However, in a high-congestion condition, it is ineffective.

TADWR [20], stands for a thermal-aware dynamic weighted routing technique which allows packets to dynamically adjust the weight of the cost model according to meet requirements of that particular area where node is situated and adaptively select next node. A dynamic model can work in both thermal and congestion scenarios. However, if thermal issues and congestion issues occur concurrently, TADWR is capable of providing a balanced approach to dealing with both of these issues. The cost model requires vital stats of neighbouring nodes such as temperature, workload and buffer status, to choose an appropriate neighbour. At every node, fetching stats from all neighbours increase communicating overhead.

Adaptive thermal-aware routing ATAR [21] traverses packets in 3D NoC based on the weighted cost model computation. Temperature is assigned fixed highest weight, subsequently decreasing weight to parameters such as path length, next neighbour queue length, and workload. Queue length and workload are solely responsible for providing congestion information. To choose the best neighbour for forwarding the intended packet, each router calculates the cost. Change in values of parameters depends on temperature change or change in network condition which will require a certain amount of time. All decisions were made at an individual node level by ATAR. Communication overhead for transfer of vital stats from neighbour nodes is extremely high in order to make decisions each time. The same procedure is repeated even in a very small span of time. Sometimes this repetition interval is so small that fetched stats are barely changed. Hence, this results in exhausting resources extensively and increasing delays in decision making. Both ATAR and TADWR are thermal-aware as well as congestion-aware routing algorithms.

In 3D NoC, it is challenging to reduce traffic in NoC, it is even harder in the presence of communication overhead for transfer of vital stats from neighbouring nodes, which is supposed to help and improve routing algorithms. Keeping record of all successful transactions not only reduces communication overhead for transfer of vital stats from neighbouring nodes in future but it also helps to take better decisions under specified supervision. As multiple paths exit between source and destination choosing a next neighbour is even more critical.

3. Methodology

The goal of the presented methodology is to reduce communication overhead to transfer vital stats from neighbour nodes in order to make decisions each time. This work presents a record keeping mechanism for maintaining localized history tables after each transaction at every node for future use. Consider if a packet is generated from the parent node, now it has to decide which way the packet should be sent to reach its destination. Each node has a small history table that records all previous activities. Only four parameters are recorded during each successful transfer i.e. incoming packet direction, target packet direction, calculated cost and sampling time. Incoming packet direction is the direction from which the packet is arriving. Target direction means after completing its process, the packet is sent in which direction. Time stamping is the time at which an arriving packet is sent towards its destination.

Abstract level router architecture for the proposed scheme is presented in Fig. 1. The routing and arbitration unit controls the crossbar in order to forward packets from incoming channels to output channels. This unit forwards the incoming packet information to IBRKM unit for decision making. IBRKM checks the history table to retrieve record of any previous related decision. Initially, there will be

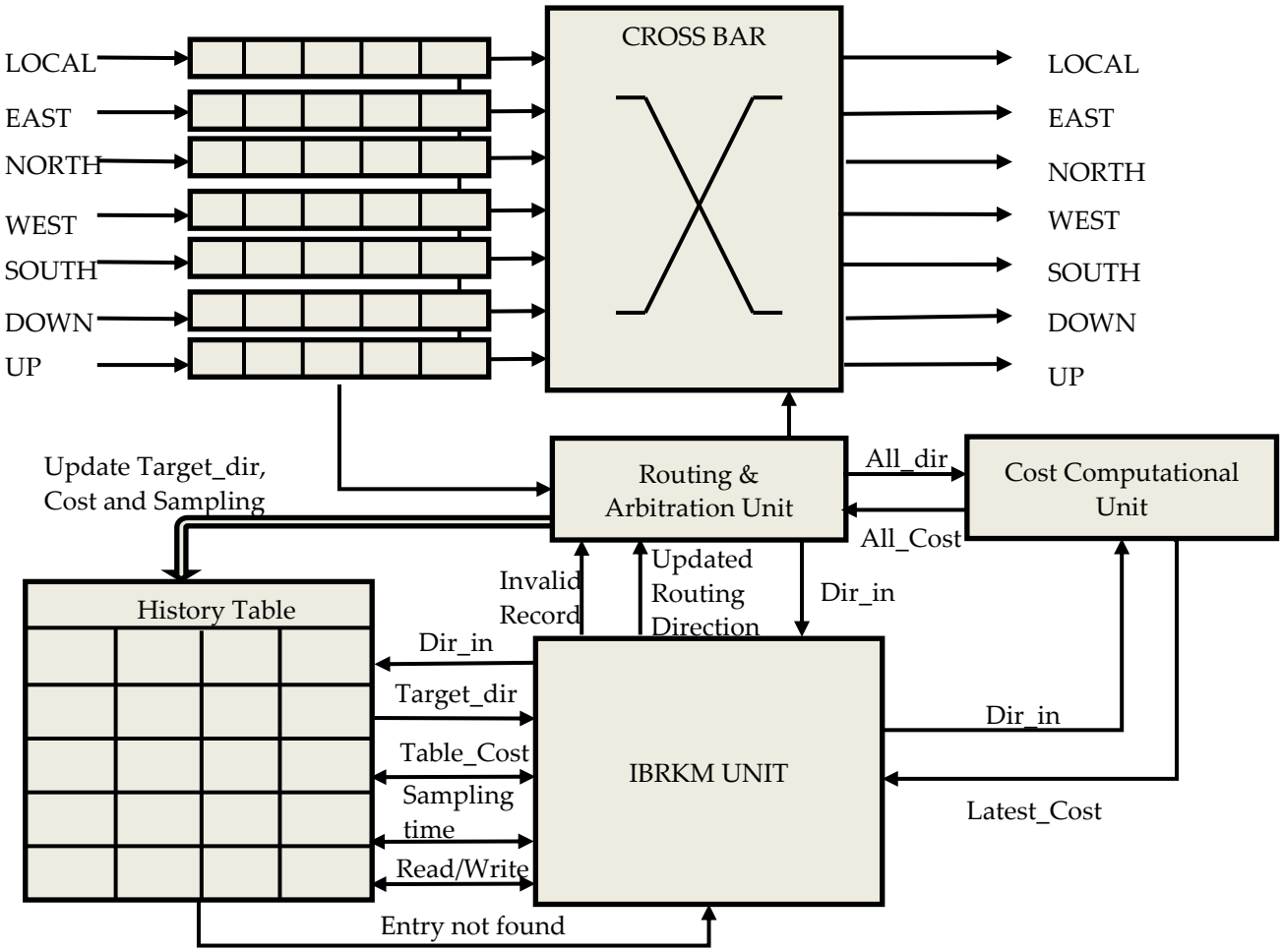


Figure. 1 Abstract level router architecture for IBRKM

no record hence it will return invalid to routing unit and routing unit will access cost of all directions, forward packet to minimum cost node and update local history table for future occurrences. If a record exists in the history table, IBRKM fetch record from the table and fetches the latest cost of the targeted node. Along with some validation checks discussed later. If the IBRKM is able to find the next neighbour, it will inform the routing unit by sending the target direction and updating the history table for the future.

Considering a packet arriving from any of the incoming directions Fig. 2, at first algorithm will check the record of previous entry for that particular incoming direction otherwise it will calculate cost of all neighbours and calculate best node and update table entry for future. If a record is found, now it is time to check its validity of the record. Entries in the table may be old and may not be valid any more. In this work, a mechanism for checking the validity of the entry is devised. It is observed that lower packet injection rates due to less traffic and less congestion table entries can be considered valid for a large amount of time whereas validity time at a higher packet injection rate should be considerably shorter.

$$\text{validity period} = \frac{1}{\text{PIR}} + \text{table sampling time} \quad (1)$$

Packet injection rate (PIR) is defined as flits/cycle/node and usually varies from 0.02 to 0.22. In order to calculate the difference of time in cycles where table entry should remain valid is proposed in Eq. (1). For lower PIR validity period should be greater and vice versa for higher PIR due to less traffic generation in lower PIR than higher PIR. A validity period is a time in which a particular entry will be considered valid as long as it is greater than the current simulation time. If the entry is invalid it will simply calculate the cost of all neighbours and calculate the best node, and update the table entry for future instances. But if the entry is valid it only fetches parameters of the previous best neighbour present in the table entry and recalculates its new cost. There is a possibility that entry is valid according to the time period but may experience an increase or decrease in cost due to changes in

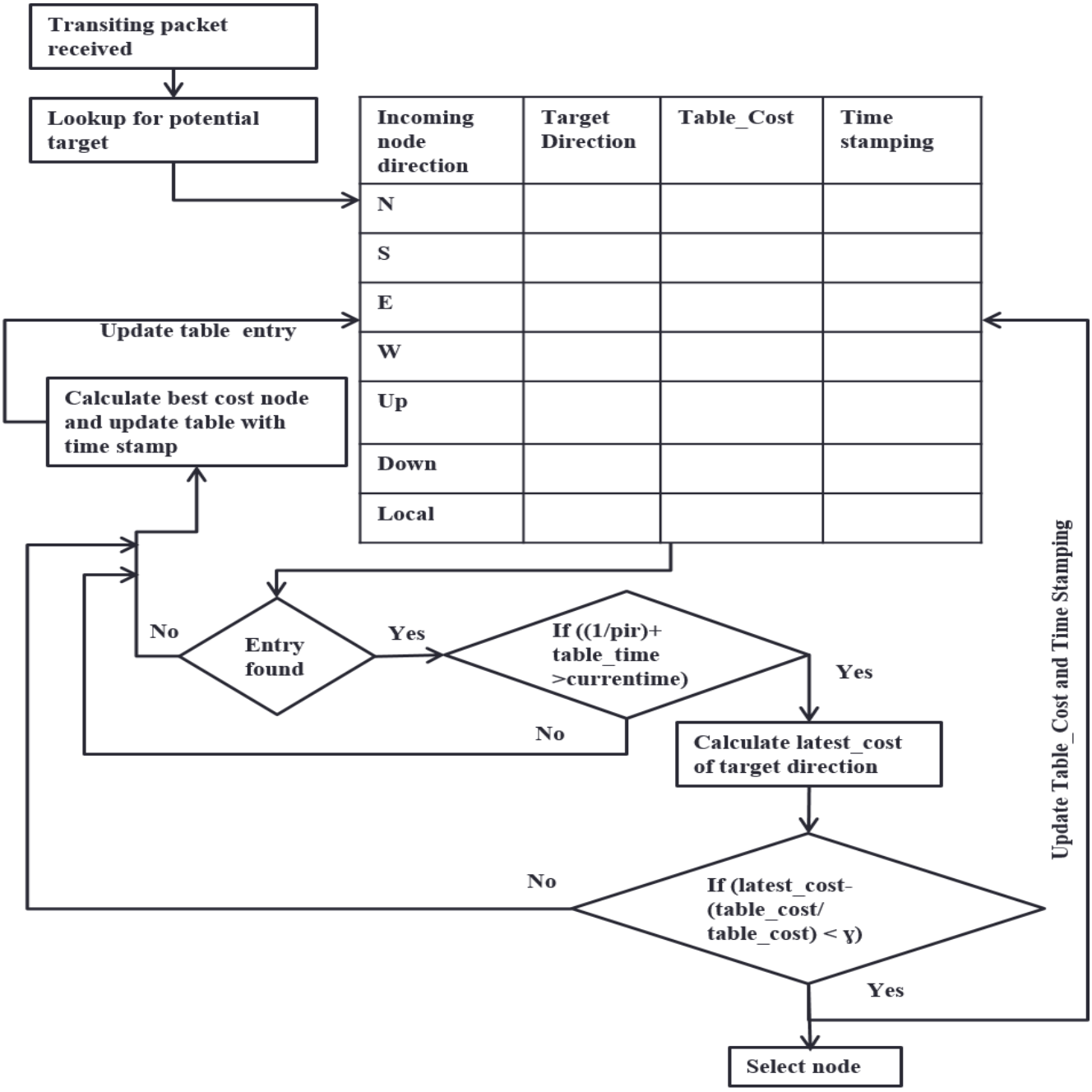


Figure. 2 IBRKM strategy flow chart

network conditions. Tiny fractional changes up to a certain limit (ɣ) can be considered valid but higher changes may lead to disastrous decisions. To calculate the change in percentage cost is proposed in Eq. (2).

$$\Delta \text{ cost percentage} = \left|\frac{\text{new cost}-\text{table\_cost}}{\text{table\_cost}}\right| \times 100 \quad (2)$$

Where new cost is the current cost and table_cost is the previously calculated cost and fetched from the history table. In order to validate a change in cost percentage it should be less than a certain limit (ɣ). In this work the validity limit has been defined by performing a set of simulations in section IV. Suppose the changes are greater than a certain limit (ɣ) it will simply calculate the cost of all neighbours and calculate the best node and update the table entry for the future. In that case, it will simply calculate the cost of all neighbours, calculate the best node, update the table entry for the future, and consider hit in the history table. Hit is considered only if entry is found in the table; it is valid in terms of time and also the cost variation is also in the acceptable range. Failing to satisfy any conditions will be considered as a miss from the history table. If the variation is within the prescribed limit, the next direction is returned and updates the history table for future transactions.

Algorithm 1. IBRKM

```
Input: destination node, source node, T, L, Q, W, dir_in
Output: path
1: function IBRKM( route_data, d_node , s_node)
2:  if s_node= d_node then
3:    directions ← direction_local
4:  else
5:     set i_node ← s_node
6:     while i_node ≠ d_node  do
7:        if( Node[i_node].dir_in[2] != 0)
8:        if ((1/PIR)+ Node[i_node].dir_in[3] >
Currentcyclenum)
9:          latest_cost ← α1.e.T + α2.e. L + α3.e. Q + α4.e.
W
10:        if (((latest_cost – table_cost)/table_cost)*100) <
ɣ)
11:              set Node[i_node].dir_in[3] =
Currentcyclenum
12:              set Node[i_node].dir_in[2] = latest_cost
13:              set direction ← Node[i_node].dir_in[1]
14:              P=push.back.direction
15:              break
16:       set dir ← getAvailableDirections(i_node)
17:          for k Є dir do
18:             set e ← i_node
19:             set cost ← α1.e.T + α2.e. L + α3.e. Q + α4.e.
W
20:             if V[s_node][i_node]+cost
<V[s_node][e.next] then
21:                  set V[s_node][e.next]← cost
22:       end for
23:       set i_node ← mincostNode(V, s_node,
directions)
24:       set direction ← direction_mincostnode( i_node)
25:             set Node[i_node].dir_in[3] =
Currentcyclenum
26:             set Node[i_node].dir_in[2] = mincost
27:             set Node[i_node].dir_in[1] = direction
28:             P= push.back.direction
29:     end while
30:    end else
31: return directions
```

IBRKM algorithm has three sections. The first section, checks the history table and performs validity checks. If validity is compromised or entry is not found, will calculate all neighbours' cost and calculate the new best node in the second section. The third section, will update the record of the history table according to the finding of section two for future references.

Algorithm 1 accepts arguments like the route data, the destination node and the source node. Whereas, route data contains parameters of node such as path length, Temperature, next router buffer, and workload where path length is donated by L, temperature is donated by T, next router buffer is donated by Q, and workload is donated by W. The source node (s_node) is the node from which the packets are generated. The destination node (d_node) is the node where the packets will be terminated. The incoming direction of the packet which can be from South, North, West, East, Up, Down, or Local, is represented by dir_in. route_data contains T, L, Q, W and dir_in. The algorithm will first examine the position of the destination and source nodes. If destination node and source node is same, will return after pushing direction Local in [line 2-3]. Now it will search in the history table to explore the previous record. If cost in the table is zero, it is accessing this record for the first time. So, the algorithm will calculate all possible neighbours and calculate their cost [line 17-21]; after calculating least cost node information is updated in the history table for future references. If the record is found for the same incoming direction, it will calculate the new cost for that neighbour only and if inspection of time validity and cost are validated. If entry is not found or invalid, it will look for all possible neighbour directions form the current node in [line 7-9]. All available directions for the node can be determined by getAvailableDirections() function. Fetch values for the parameters T, L, Q, and W for the intermediate node represented by i_node. The sum of weighted cost is then computed. mincostNode() is used to determine the node number of minimum cost node among all the neighbours. If the computed cost is less than the existing cost, the cost matrix is updated at [line 10]. The lowest cost node will now become the new i_node. Update record of minimum cost, target direction and current cycle number in the history table for future occurrences [line 11-14], whereas the current cycle number is determined by Currentcyclenum() function. The direction of the minimum cost node is pushed in directions by push.back.directions and returned. If any of the validity conditions are false, the algorithm will calculate all possible neighbours and calculate their cost [line 17-21], the cost model considered in this work is similar to [21] in [line 19], after calculating least cost node information is updated in the history table for future references [line 25-28]. The minimum cost node direction is pushed in a direction array and repeated until the destination is arrived.

4. Results and discussion

A cycle-accurate Access Noxim [22] simulator has been used to simulate IBRKM. The Access Noxim simulator works in conjunction with Noxim [23] and HotSpot [24]. Noxim was built with the System C library. A command-based interface is used to define and set NoC settings, such as the

Table 1. Simulation parameters

Parameters	Value
Simulation Time (Cycles)	200,000
Network Dimension	8 x 8 x 4
Packet size (Flits)	2~10
Buffer Size (Flits)	16
Warm-up Time (Cycles)	10,000
Synthetic Traffic Pattern	Bit-Reversal, Random, Shuffle
Packet injection interval	0.02
Packet injection rate (PIR) (flits/cycle/node)	0.02-0.22

buffer capacity, routing algorithm, network size and dimensions, traffic distribution system, number of network nodes, packet injection rate, etc. The simulator is capable of evaluating NoC capacity in terms of thermal profile, power consumption, throughput, and global average delay. For deep investigation of a single transaction Access-Noxim also provides a transaction level analysis mode.

4.1 Simulation setup

The proposed routing algorithm's performance is evaluated using an 8 x 8 x 4 3D NoC. Table 1 lists the parameters considered in the simulations. Different synthetic traffics utilized to analyse the capabilities of IBRKM compared to its adversaries. IBRKM is compared with state-of-the-art TADWR, ATAR, OE 3D, and fully-adaptive routing. Different PIR (Packet Injection Rate) simulations are made to run for 200,000 cycles. PIR (flits/cycle/node) is ranging from 0.02 to 0.22 with a 0.02 interval. The time interval distribution determines the occasion at which the packet is injected.

4.2 Performance evaluation

Bit-reversal, random, and shuffle traffic patterns are used in this section. Graphs for global average delay (cycle) are exhibited for each synthetic traffic. Fig. 3(a) depicts the global average delay of 3D OE, fully-adaptive routing, ATAR, TADWR, and IBRKM in Bit-Reversal traffic. As it can be seen that injection rates ranging from 0.02 to 0.06 for ATAR and from 0.02 to 0.10 for TADWR, both algorithms produce similar results compared to IBRKM; however, as the injection rate increases, the results begin to diverge. In comparison to state-of-the-art routing algorithms, the graph demonstrates that after 0.10, IBRKM with a history-based record mechanism has sustained. However, due to lack of ability to handling heavy traffic loads both fully adaptive and 3D OE routing rocketed sharply even at lower PIR. Percentage improvement in global average delay is 12.07 % over TADWR and 49.75 % over ATAR in Bit-Reversal traffic.

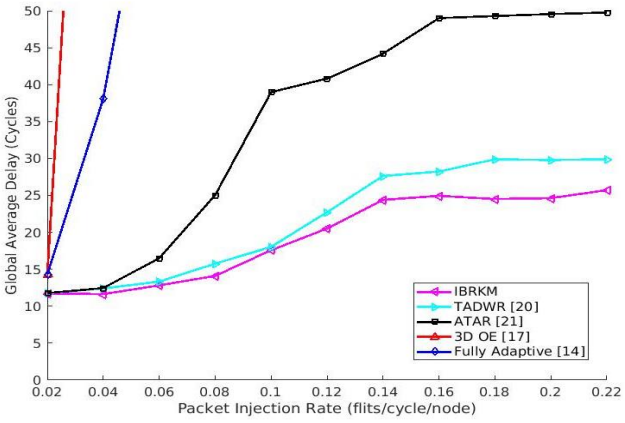
Fig. 3(b) depicts the global average delay of 3D OE, fully-adaptive routing, ATAR, TADWR and IBRKM under random traffic. It can be seen that injection rates ranging from 0.02 to 0.10 for ATAR and from 0.02 to 0.12 for TADWR, both algorithms produce similar results compared with IBRKM, but when injection rates rise beyond 0.12, the difference becomes more pronounced. Analysing the graph it is apparent that after 0.12, IBRKM with a history-based record mechanism sustained as compared to ATAR and TADWR. However, due to lack of ability to handling heavy traffic loads both fully adaptive and 3D OE routing flew vigorously even at lower PIR. Percentage improvement in global average delay is 11.17 % over TADWR and 32.24 % over ATAR.

The global average delay of 3D OE, fully-adaptive routing, ATAR, TADWR and IBRKM under Shuffle traffic is presented in Fig. 3(c). It can be seen that while the injection rate is between 0.02 and 0.04 for ATAR and from 0.02 to 0.12 for TADWR, both algorithms produce identical results compared to IBRKM, but when the injection rate exceeds PIR 0.12, the difference between TADWR, ATAR and IBRKM becomes evident. Analysing the graph, it is noticeable that after the PIR 0.12, IBRKM with a history-based record keeping mechanism performed better than TADWR and ATAR. However, due to lack of ability to handling heavy traffic loads both fully adaptive and 3D OE routing shot severely even at lower PIR. Percentage improvement in global average delay is 8.12% over TADWR and 36.43 % over ATAR.

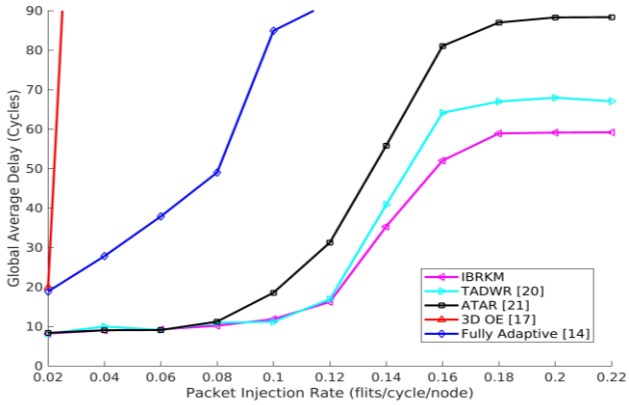
Considerable improvement in terms of hits can be observed due to the history-based record keeping mechanism observed in Fig. 4. It can be seen that at lower packet injection rates history table hits are greater than misses; it is because the number of hits depends on validity conditions. If validity conditions are relaxed, especially if it is allowed always to keep the historical data valid, hits will increase many folds but authenticity will be compromised. Hence a balanced approach is required to keep routing significant.

In Fig. 4, different values for ɣ such as 5 %, 10 %, and 20 % are applied respectively on Bit-Reversal traffic to observe their impact on the number of hits at various packet injection rates. The hit ratio can be calculated using Eq. (3) given below.

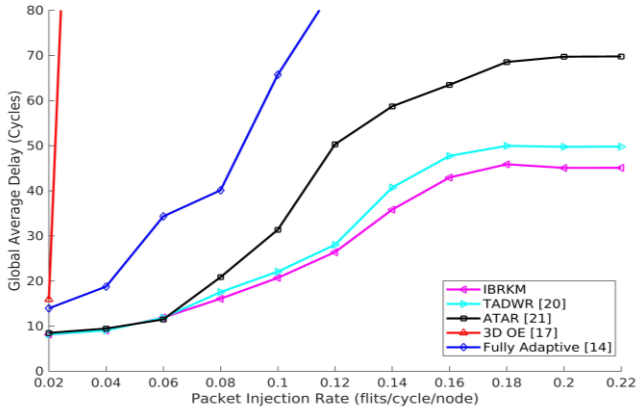
$$Hit\ Ratio\ (\%) = \frac{\text{Number of Hits}}{\text{Number of Hits+Number of Misses}} \text{ x } 100 \quad (3)$$



(a)

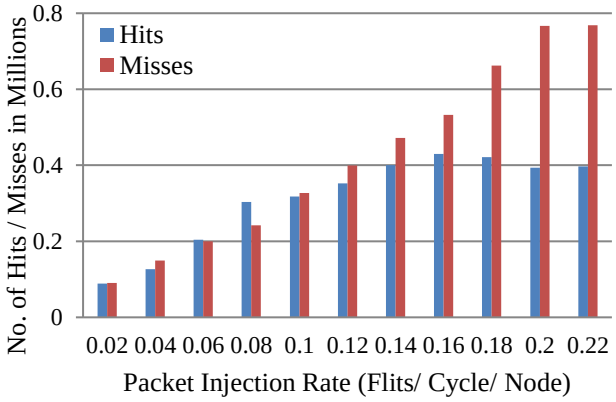


(b)

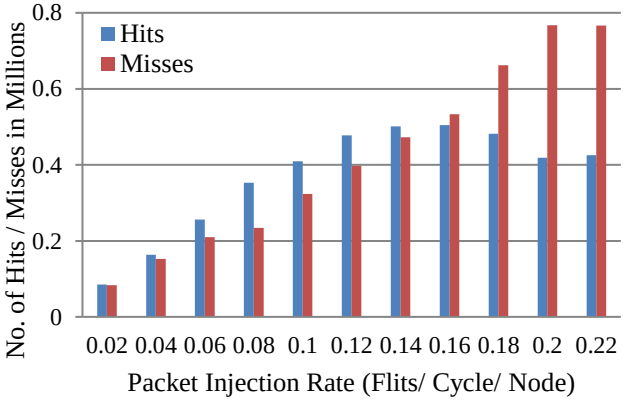


(c)

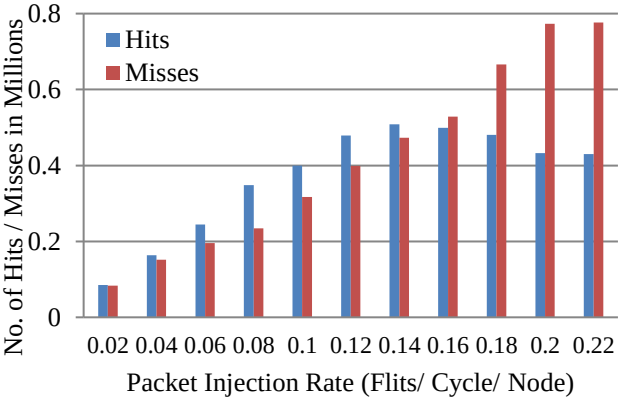
Figure. 3 Comparison of global average delay under various traffic patterns: (a) Bit-reversal traffic, (b) Random traffic, and (c) Shuffle traffic



(a)



(b)



(c)

Figure. 4 Number of hits vs misses in millions when: (a) ɣ=5, (b) ɣ=10, and (c) ɣ=20

Increasing the validity difference for cost from 5 % to 20 % for Bit-Reversal traffic it can be seen that at 5 % Fig. 4(a) the number of hits as compared to misses are less than 10 % Fig. 4(b) and 20 % Fig. 4(c). On close examination it is observed that 20 % of hits are similar to 10 %. This means that the value for ɣ should not be greater than 10 %. Hence during our experiments ɣ is assigned 10 % and various traffic synthesis are applied, such as Random, Bit-Reversal and Shuffle. It is observed that 26.34 % more hits in Random traffic, 46.97 % more hits in Bit-Reversal and 60.24 % more hits in Shuffle traffic.

In this work, IBRKM has been tested with different configurations of 3D mesh topology namely: 2x2x4, 4x4x4, 8x8x4. Where, each of the

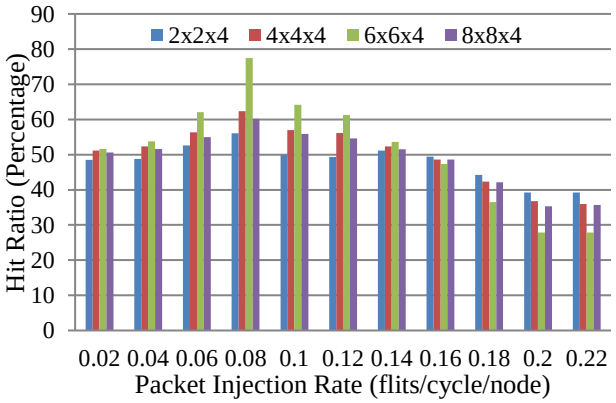


Figure. 5 IBRKM hit ratio under bit-reversal traffic

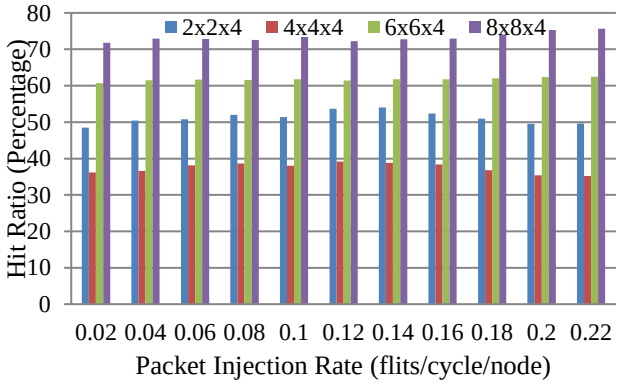


Figure. 6 IBRKM hit ratio under random traffic

configurations has different number of nodes and links. IBRKM has been assessed under synthetic traffic such as Bit-Reversal, Random and Shuffle traffic.

In general as PIR increases, the hit ratio suffers due to increasing traffic and the expiry of validity checks. In Bit-Reversal traffic initially at lower PIR, 6x6x4 performed better, and as PIR is high it performs poorly in Fig. 5. Whereas, 8x8x4 and 4x4x4 shows similar but stable response to Bit Reversal traffic.

Random traffic is also applied to the various topologies Fig. 6. 8x8x4 stood up and performed far better than other counterparts. Simultaneously, 4x4x4 is amongst the worst. As 8x8x4 is the biggest topology and 2x2x4 is the smallest.

Shuffle traffic is applied in Fig. 7. All configurations have performed well initially and as the PIR increases hit ratio started to reduce. 8x8x4 performs slightly better even in higher PIR. Overall in all traffics, 8x8x4 has been performing slightly better than other configurations, which helps the argument that IBRKM is scalable and better performance is observed in bigger configurations.

The hardware overhead of IBRKM over TADWR and ATAR is substantially 25% higher. Besides higher area, it presents approximately 52% better

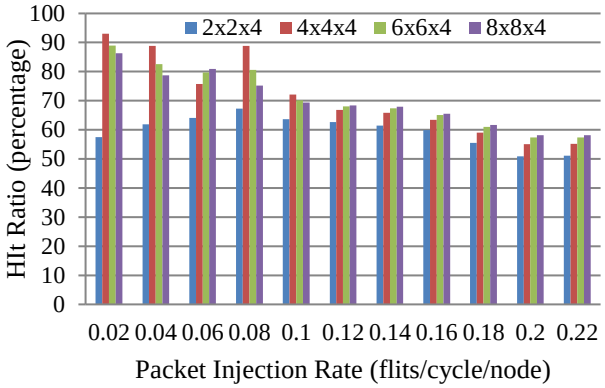


Figure. 7 IBRKM hit ratio under shuffle traffic

Table 2. Area efficiency (T/H) comparison after normalization

	ATAR	TADWR	IBRKM
Throughput Improvement (T)	1	1.41	1.90
Hardware Cost Overhead (H)	1	1.12	1.25
Area Efficiency (T/H)	1	1.26	1.52

efficiency over ATAR and 25% over TADWR. A normalized comparison of throughput and area is given in Table 2.

5. Conclusion

Congestion in NoC due to heavy traffic is a concern that needs to be addressed. Congestion is capable of generating more concerns such as thermal difficulties, thermal and traffic hotspots and increased delay in the network. Congestion can be removed by improving congestion strategy or by reducing excessive in-network traffic load. This work proposes IBRKM routing algorithm that addresses in-network congestion and reduce communication overhead for transfer of vital stats of nearby routers for future use. IBRKM works on a history table record of previous transactions carried in the node under validity conditions. The Proposed algorithm performs certain validity checks before using recorded iteration from the history table. With the help of proposed technique up to 26-61 % of valid records were found from the history table which is a significant improvement. As far as global average delay is concerned, 8 to 12 % improvement is found under different synthetic traffic compared to the state-of-the-art. IBRKM is scalable and better performance is observed in bigger configurations.

Conflicts of Interest

There are no conflicts of interest declared by the authors.

Author Contributions

The visualization, writing-original draft presentation, writing-review, paper conceptualization, investigation, methodology, validation formal analysis, software, data curation, investigation, formal analysis has been performed by 1st author. The project administration and supervision has been performed by 2nd author.

Acknowledgments

This research is supported by Ministry of Higher Education Malaysia (MOHE) and conducted in collaboration with Research Management Center (RMC) at the Universiti Teknologi Malaysia (UTM) under Fundamental Research Grant Scheme with grant number: R.J130000.7851.5F029. The authors appreciate greatly for the support.

References

[1] T. Boraten and A. K. Kodi, "Runtime techniques to mitigate soft errors in Network-on-Chip (NoC) architectures", *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, Vol. 37, No. 3, pp. 682-695, 2018.

[2] R. Salamat, M. Khayambashi, M. Ebrahimi, and N. Bagherzadeh, "LEAD: An Adaptive 3D-NoC Routing Algorithm with Queuing-theory Based Analytical Verification", *IEEE Trans. Comput.*, No. 1, p. 1, 2018.

[3] E. Fusella and A. Cilardo, "Lattice-Based Turn Model for Adaptive Routing", *IEEE Trans. Parallel Distrib. Syst.*, No. 1, p. 1, 2018.

[4] Z. Ghaderi, A. Alqahtani, and N. Bagherzadeh, "AROMa: Aging-Aware Deadlock-Free Adaptive Routing Algorithm and Online Monitoring in 3D NoCs", *IEEE Trans. Parallel Distrib. Syst.*, Vol. 29, No. 4, pp. 772-788, 2018.

[5] Y. Y. Chen, E. J. Chang, H. K. Hsin, K. C. J. Chen, and A. Y. A. Wu, "Path-diversity-aware fault-tolerant routing algorithm for network-on-chip systems", *IEEE Trans. Parallel Distrib. Syst.*, Vol. 28, No. 3, pp. 838-849, 2017.

[6] S. T. Muhammad, M. Saad, A. A. E. Moursy, M. A. E. Moursy, and H. F. A. Hamed, "CFPA: Congestion aware, fault tolerant and process variation aware adaptive routing algorithm for asynchronous Networks-on-Chip", *J. Parallel Distrib. Comput.*, 2019.

[7] D. Kouzapas, "Towards fault adaptive routing in metasurface controller networks", *J. Syst. Archit.*, Vol. 106, No. December 2019, p. 101703, 2020.

[8] M. Kaleem and I. F. B. Isnin, "A Survey on Network on Chip Routing Algorithms Criteria", *Adv. Intell. Syst. Comput.*, Vol. 1188, pp. 455-466, 2021.

[9] F. Bahman, A. Reza, M. Reshadi, and S. Vazifedan, "CACBR: Congestion Aware Cluster Buffer base routing algorithm with minimal cost on NOC", *CCF Trans. High Perform. Comput.*, 2020.

[10] R. Akbar and F. Safaei, "A novel congestion-aware routing algorithm with prediction in mesh-based networks-on-chip", *Nano Commun. Netw.*, Vol. 26, p. 100322, 2020.

[11] N. Shahabinejad and H. Beitollahi, "Q-Thermal: A Q-Learning-Based Thermal-Aware Routing Algorithm for 3-D Network On-Chips", *IEEE Trans. Components, Packag. Manuf. Technol.*, Vol. 10, No. 9, pp. 1482-1490, 2020.

[12] J. Huang, W. Zhong, Z. Li, and S. Chen, "Lagrangian relaxation-based routing path allocation for application-specific network-on-chips", *Integration*, Vol. 61, No. June 2017, pp. 20-28, 2018.

[13] C. Chen, Q. Li, N. Li, H. Liu, and Y. Dai, "Link-Sharing: Regional Congestion Aware Routing in 2D NoC by Propagating Congestion Information on Idle Links", In: *Proc. of 2018 IEEE 3rd International Conference on Integrated Circuits and Microsystems*, pp. 291-297, 2018.

[14] J. Wang, H. Gu, Y. Yang, and K. Wang, "An energy-and buffer-aware fully adaptive routing algorithm for Network-on-Chip", *Microelectronics J.*, Vol. 44, No. 2, pp. 137-144, 2013.

[15] J. Augustine, K. Raghavendra, J. Jose, and M. Mutyam, "Router Buffer Caching for Managing Shared Cache Blocks in Tiled Multi-Core Processors", In: *Proc. of 2020 IEEE 38th International Conference on Computer Design*, pp. 239-246, 2020.

[16] M. Momeni and A. J. Pozveh, "An Adaptive Approximation Method for Traffic Reduction in Network on Chip", In: *Proc. of 2020 6th Iranian Conference on Signal Processing and Intelligent Systems*, pp. 1-5, 2020.

[17] N. Dahir, T. Mak, R. A. Dujaily, and A. Yakovlev, "Highly adaptive and deadlock-free routing for three-dimensional networks-on-chip", *IET Comput. Digit. Tech.*, Vol. 7, No. 6, pp. 255-263, 2013.

[18] S. C. Lee and T. H. Han, "Q-Function-Based Traffic-and Thermal-Aware Adaptive Routing for 3D Network-on-Chip", *Electronics*, Vol. 9, No. 3, p. 392, 2020.

[19] L. Shen, N. Wu, G. Yan, and F. Ge,

"Collaborative thermal-and traffic-aware adaptive routing scheme for 3D Network-on-Chip systems", *IEICE Electron. Express*, pp. 18-20200425, 2021.

[20] M. Kaleem and I. F. B. Isnin, "Thermal-aware Dynamic Weighted Adaptive Routing Algorithm for 3D Network-on-Chip", *Int. J. Adv. Comput. Sci. Appl.*, Vol. 12, No. 11, pp. 342-348, 2021.

[21] R. Dash, A. Majumdar, V. Pangracious, A. K. Turuk, and J. L. R. Martín, "ATAR: An Adaptive Thermal-Aware Routing Algorithm for 3-D Network-on-Chip Systems", *IEEE Trans. Components, Packag. Manuf. Technol.*, No. 99, pp. 1-8, 2018.

[22] K. Y. Jheng, C. H. Chao, H. Y. Wang, and A. Y. Wu, "Traffic-thermal mutual-coupling co-simulation platform for three-dimensional network-on-chip", In: *Proc. of 2010 International Symposium on VLSI Design, Automation and Test*, pp. 135-138, 2010.

[23] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Noxim: An open, extensible and cycle-accurate network on chip simulator", In: *Proc. of 2015 IEEE 26th International Conference on Application-specific Systems, Architectures and Processors*, pp. 162-163, 2015.

[24] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "HotSpot: A compact thermal modeling methodology for early-stage VLSI design", *IEEE Trans. Very Large Scale Integr. Syst.*, Vol. 14, No. 5, pp. 501-513, 2006.