Deadlock Avoidance in Torus NoC Applying Controlled Move via Wraparound Channels

Surajit Das and Chandan Karfa

Abstract Wraparound channels in Torus Network-on-Chip (NoC) help in reducing overall hop counts of a given traffic. The cyclic paths formed by wraparound channels make Torus NoC vulnerable to deadlock. Virtual channels (VC) or dedicated buffers are used in existing routing algorithms for Torus NoC to make them deadlock free. In this work, we present a deadlock avoidance approach in Torus without using VC or dedicated buffer. An *Arc Model* is proposed by restricting some movements via wraparound channels. Using different combinations from *Arc Model*, variants of algorithms are possible. As an application of *Arc Model*, a deadlock-free routing algorithm for Torus NoC is presented in this work without using any dedicated buffer or VC. Experimental results of that algorithm support its deadlock freedom and its effectiveness in saving hop counts.

Keywords Network-on-chip · Torus NoC · Wraparound channel · Deadlock avoidance

1 Introduction

Torus or k-ary n-cube NoC is widely used due to its high path diversity and topological symmetry with uniform node degree. This topology is largely used over the past 30 years in commercial supercomputer and data centre [1]. A 5 × 5 Torus NoC with router numbers and co-ordinates are shown in Fig. 1a. It is convenient to visualise Torus with help of Mesh NoC by connecting all boundary routers of a Mesh NoC to the corresponding opposite boundary routers. These additional connections help in

https://www.iitg.ac.in/ckarfa.

S. Das (✉) · C. Karfa
Indian Institute of Technology Guwahati, Guwahati 781039, Assam, India
e-mail: d.surajit@iitg.ac.in

C. Karfa
e-mail: ckarfa@iitg.ac.in

B. Mishra et al. (eds.), *Artificial Intelligence Driven Circuits and Systems*,
Lecture Notes in Electrical Engineering 811,
https://doi.org/10.1007/978-981-16-6940-8_8

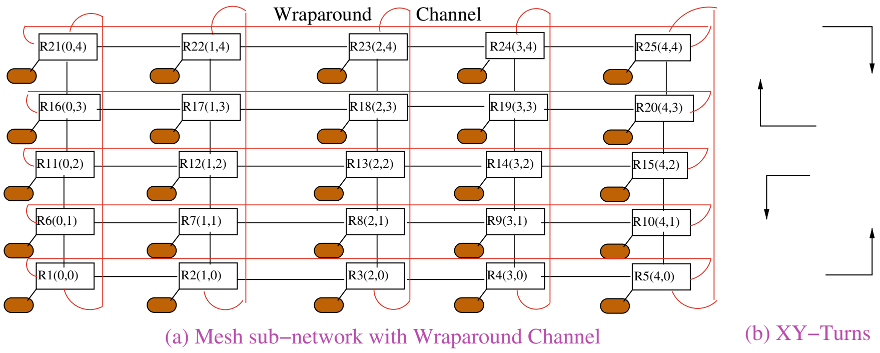

Fig. 1 **a** 5 × 5 Torus NoC, **b** Permitted turns for XY-routing

reducing hop counts and are termed as wraparound channels. Nevertheless, they also create circular paths which make Torus prone to deadlock. While in some kinds of traffic a Torus may have better performance, simultaneously it is harder to achieve deadlock freedom and verify it in hardware. Therefore, research on deadlock detection and avoidance in Torus has importance. Deadlock arises due to circular dependency for resources that ultimately halts the system. For avoiding deadlock in Torus, different techniques like bubble flow control [2, 3] or virtual channels (VC) with datelines [4] are used. All these methods have overhead in terms of extra buffer or VCs. This work is aimed at avoiding deadlock in Torus without using extra buffer or extra channels inspired from two classical approaches: Turn Model [5] and Channel Dependency Graph (CDG) [6].

In this work, we visualise Torus NoC by logically subdividing it into Mesh subnetwork and wraparound channels. Though wraparound channels are similar to other normal connections, we put a restricted move on wraparound channels. Marking some connections as wraparound channels are like dateline approach that is used for switching between VCs. In the dateline approach of deadlock avoidance, two VC classes VC0 and VC1 are used [4]. Certain co-ordinates are fixed for each X and Y-dimension for considering them as the dateline. If a packet is inserted into the network, it uses VC0. If a packet crosses the predefined dateline, for breaking the cyclic dependency, the packet is put from VC0 to another class VC1. Similarly, after crossing datelines (boundary of a Mesh NoC in our work), we put certain restrictions while forwarding a packet.

One approach of avoiding deadlock in Torus NoC or in ring network is to use extra buffers for implementing bubble flow control [2, 3]. Another approach is to use VCs [6–8] to overcome cyclic dependencies. However, these methods have overheads in terms of buffer usages, managing VCs and power consumption. Turn model (Fig. 1b) is an approach that does not use VCs or extra buffer for avoiding deadlock in Mesh NoC [5]. This motivates us to look into Turn model approach for deadlock avoidance in Torus. Deadlock verification works in [9] demonstrates that deadlock occurs by XY-routing in Torus NoC. Abstractly, the deadlock in Torus occurs because of the backward movement of a packet after taking wraparound channels in Torus. The question that motivates us: *Can we avoid deadlock in Torus without using VC and extra buffer and improve the utilisation of wraparound channels with certain restrictions in routing?* In this work, we, therefore, logically split a wraparound channel into *two arcs* to avoid a backward turn. We apply Turn model and take help of communicating finite state machine (CFSM)-based verification framework [9] for detecting deadlock while using those logically split wraparound channels. Since, the purpose of this work is to present a new deadlock avoidance approach only, rigorous performance evaluation and comparison of algorithms yielded from this approach is not presented in this work.

However, the *Turn model* is not applicable for deadlock detection for routing algorithms in Torus NoC. As per Turn Model, there is an impression that XY-routing is deadlock free in Torus NoC as well. On the other hand, XY-routing is deadlock prone in Torus NoC as shown in [9]. Therefore, it is desirable to have an accurate model to detect deadlock in Torus NoC. To the best of our knowledge, there is no such specific model available for accurate detection of deadlock in Torus NoC. Prime contributions of this work are

– An *Arc Model* is proposed for avoiding deadlock in Torus NoC without using extra buffer and VCs.
– As an utility of *Arc Model*, a deadlock-free deterministic routing algorithm for Torus NoC is demonstrated. More such algorithms are possible using *Arc Model*.
– The algorithm is implemented in CFSM based deadlock detection framework and compared with First Hop algorithm [5]. Experimental results show its effectiveness in saving hop counts.

The rest of the paper is organised as follows. Related works are presented in Sect. 2. Basic terminologies and analysis of deadlock for XY-routing in Torus NoC are presented in Sect. 3. The proposed *Arc Model* for deadlock avoidance in Torus NoC and its application is presented in Sect. 4. Experimental results are presented in Sect. 5. Finally, we conclude with the future direction of work in Sect. 6.

2 Related Work

Research on deadlock detection and avoidance for interconnection network and NoC is a more than three decades old problem with further scopes for improvements.

In bubble flow control, it uses dedicated buffer for avoiding deadlock in Torus NoC [2, 3]. It ensures at least one empty buffer slot in the ring that prevents deadlock. This approach has overhead in terms of maintaining an empty buffer. Dally et al. [6] introduce CDG for detecting the deadlock cycle. A necessary and sufficient condition for deadlock-free routing is the absence of a cycle in a channel dependency graph. The cycles in the channel dependency graph are removed by splitting physical channels into a group of VCs [6]. For eliminating the cycles, VCs are ordered and the routing algorithm is restricted to route packets in decreasing order. In dateline approach [4], two classes of VCs are used. A packet is forced to use another VC class after crossing the dateline to prevent cyclic dependency. Theorem on necessary and sufficient conditions for deadlock freedom of adaptive routing algorithms with VCs are presented in [8]. It provides an alternate escape path for messages that are involved in cyclic dependency.

Glass et al. [5] propose an alternate approach of designing deadlock-free routing algorithms called Turn model. Instead of adding extra buffer or VCs, the model is based on analysing the changes of direction by packets and the resultant cycles formed by them. All possible deadlock scenarios in Mesh NoC due to forbidden turns are well presented in these works [5]. For Torus NoC, use of wraparound channels are permitted only at the first hop (First Hop Algorithm) [5]. Otherwise, they result in deadlock. The restriction for using wraparound channel only at its first hop is eliminated in our work. For enhancing performance while using turn restrictions, research on effective turn distributions is presented in [10, 11]. Another deadlock avoidance approach partitions channels into disjoint sets without containing any cyclic dependency [12]. In that approach, VC is required for avoiding deadlock via wraparound channel. Since avoiding deadlock is expensive, in recent deadlock recovery research, different approaches are proposed for relieving from deadlock once it occurs [13–15].

To summarise, bubble flow control methods have overhead in terms of extra buffer [2, 3]. Similarly, VC methods have overhead for maintaining VCs and separate buffer corresponding to each VC [6, 8]. Turn model approach has no such overhead but wraparound channels are applicable only for limited packets that are injected only from boundary routers. This work intends for avoiding deadlock in Torus NoC with better utilisation of wraparound channel without using additional VC or buffer.

3 Deadlock in the Torus NoC and Dependency Graph

3.1 Deadlock in Torus NoC

Let us consider five packets p1(1, 3), p2(2, 4), p3(3, 5), p4(4, 1) and p5(5, 2) in a 5×5 Torus NoC in Fig. 2a. For each packet, the source and destination router numbers are put in the bracket. The current location of each packet is shown in Fig. 2a. For example, the packet p1(1, 3) has started from the router R1 is moving towards destination router R3 and is currently in the West port buffer of router R2. For the packet p4(4, 1), the shortest path $R4 \rightarrow R5 \rightarrow R1$ with the wraparound channel is going to be followed. In a similar way, the path for p2(2, 4) is $R2 \rightarrow R3 \rightarrow R4$, for p3(3, 5) is $R3 \rightarrow R4 \rightarrow R5$, and for p5(5, 2) is $R5 \rightarrow R1 \rightarrow R2$. Let us assume, each router has a buffer with a capacity of one packet, and packets p1, p2, p3, p4, and p5 are transmitted at the same time from their source routers. They have reached the input buffer of their next router, R2, R3, R4, R5, R1, respectively, as shown in

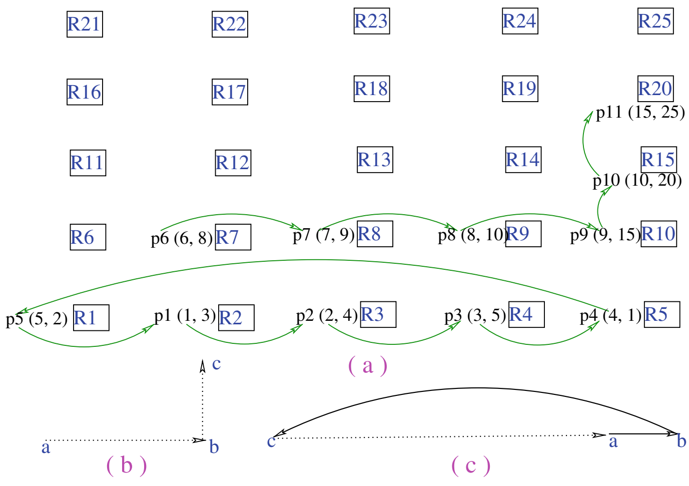

Fig. 2 **a** Buffer Dependency. **b** No Deadlock. **c** Deadlock

Fig. 2a. Therefore, the West input port Buffer of each router in Fig. 2a are full at the same moment. Packet p1(1, 3) is occupying the West buffer of router R2, waiting for the release of West port buffer of R3. Packet p2(2, 4) is occupying the same R3 buffer and is requesting the R4 buffer. Packet p3(3, 5) is occupying the R4 buffer and is requesting an R5 buffer. Packet p4(4, 1) is occupying the R5 buffer and is requesting an R1 buffer. Lastly, packet p5(5, 2) is occupying the R1 buffer and requesting an R2 buffer, currently occupied by packet p1. Thus, all five packets are waiting for each other for the release of buffer in a cyclic manner. This is an example of deadlock in Torus NoC. Though we have considered buffer capacity for only one packet in this example, deadlock is possible even if buffer capacity is for more than one packet but not unlimited.

3.2 Dependency Graph for Deadlock Representation

It is possible to spread buffer dependency in a direction between consecutive routers that have limited buffer, provided that each router generates packets destined in a specific direction, and destination nodes are at least two routers away from the source nodes. The exact path in which packets are waiting for the buffer is shown with a green arrow in Fig. 2a. *Buffer dependency graph represents only the direction in which dependency spreads without giving the router-wise detail path. It is like a CDG* [6], *as each vertex in the graph represents a buffer and its connecting input channel. One minor difference with CDG is, for representation convenience, all intermediate vertices are not put in the graph unless there is a change in direction.* The corresponding dependency graph for packet p6, p7, p8, p9, p10, is shown in Fig. 2b.

The corresponding dependency graph for packets p1, p2, p3, p4 and p5 is shown in Fig. 2c. Here, *dotted line* with arrow indicates the spreading of dependency without using a wraparound channel. *Solid* and *curved line* with arrow indicates the spreading of dependency due to the use of wraparound channels. Since the dependency graph forms a cycle in Fig. 2c, corresponding packets p1, p2, p3, p4 and p5 result in a deadlock. There is no deadlock for packet p6, p7, p8, p9, p10, as there is no cycle in Fig. 2b.

4 Deadlock Avoidance in Torus NoC

In a Torus NoC, direction wise there are four types of wraparound channels, namely, EW (from East to West boundary), WE (West to East), NS (North to South) and SN (South to North). Dependency graphs for wraparound channels are shown in Fig. 3. For the NS wraparound channel, there could be a sequence of packets waiting for each other's buffer in the cyclic path shown in Fig. 3a. The same case is applicable for the other three wraparound channels (Fig. 3b–d). Therefore, these wraparound channels are the primary cause of deadlock in Torus NoC.

As per the dependency graphs in Fig. 3, *backward movement immediately after taking any wraparound channel must be avoided to prevent deadlock.* After taking an X-wraparound channel EW, immediate movement *bd* towards the East has the potential for creating a deadlock in Fig. 3c. Similarly, after taking a Y-wraparound channel NS/SN, immediate movement in Y-direction towards the North/South has the potential for creating a deadlock. Therefore, restrictions need to be imposed for breaking the cycle created by wraparound channels. To achieve this, we propose an *Arc Model* in this work. *Arc Model* is applicable only to topologies like Torus where inherent cyclic paths (many rings) are present. Therefore, *Arc Model* is not applicable for NoC topologies like Mesh or Butterfly. Since no VC or additional buffer is used in *Arc Model*, there is no energy and area overhead in this approach as compared to the VC [7, 8] and bubble flow control [2, 3].

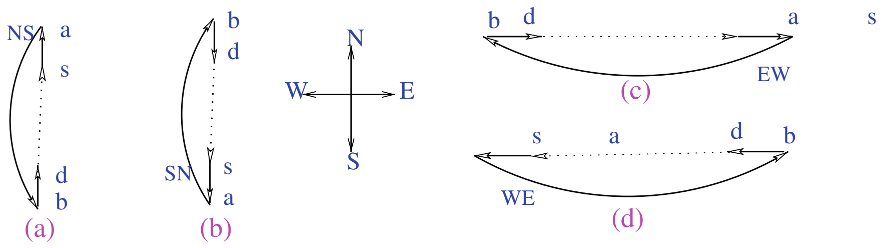

Fig. 3 Dependency for **a** NS, **b** SN, **c** EW and **d** WE wraparound channels

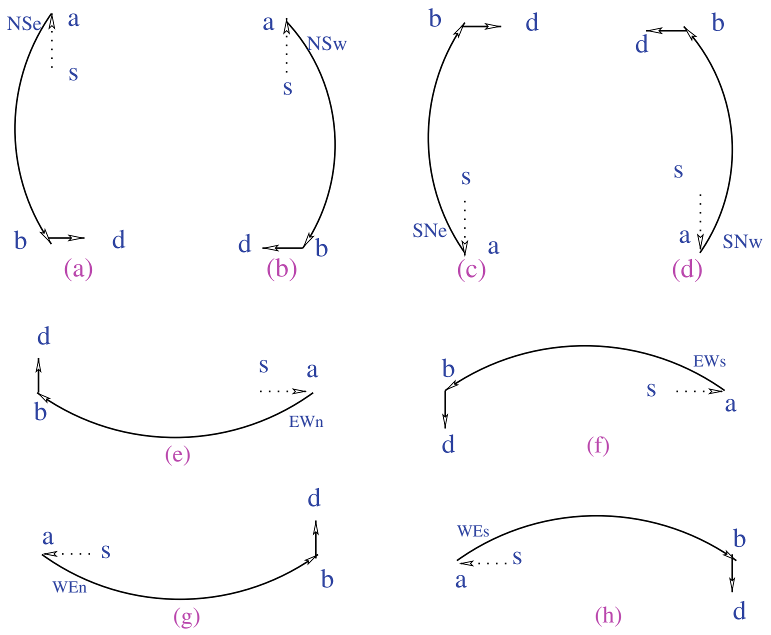

Fig. 4 Arc Model: **a** NSe, **b** NSw, **c** SNe, **d** SNw, **e** EWn, **f** EWs, **g** WEn, **h** WEs Arcs

4.1 Arc Model for Torus NoC

In our proposed Arc model, each wraparound channel is divided into two arcs by imposing restrictions on backward movement after the wraparound channel is taken. For example, the NS channel in Fig. 3a is divided into NSe (Fig. 4a) and NSw *arcs* (Fig. 4b). Essentially, after taking the NS channel, the packet is forced to move either in the East or West direction. Here, NSe /(NSw) *arc* is applicable if NS wraparound channel reduces hop counts and destination is in South-East/(South-West) direction. In a similar way, for breaking the cycle for all wraparound channels in Fig. 3, they are categorised into eight *arcs*. The eight possible *arcs* in the proposed *Arc Model* are shown in Fig. 4.

4.2 Application of Arc Model

For deadlock-free routing algorithm in Torus, instead of unrestricted wraparound channels, we use *arc(s)* from the *Arc Model* along with the XY-routing in the Mesh sub-network in such a way that they do not create cyclic dependency. Even though all the arcs are individually deadlock free, some of their combinations are deadlock prone. In this work, we have not systematically categorised all such arcs. More research with intensive verification is needed to categorise deadlock-free *arc* combinations, deadlock prone arc combinations and how many *arcs* can be used with a routing algorithm with deadlock freedom. In this work, we demonstrate one deadlock-free routing algorithm using two *arcs*. Variants of such algorithms using more *arcs* and more turns applying turn distribution [10, 11] are possible.

Algorithm 1: NE-SE Algorithm

```
Data: The source(S) and destination(D) co-ordinates of a packet in an NxN Torus NoC are
      S($x_s$, $y_s$) and D($x_d$, $y_d$), respectively. After each move, S is updated. Between a
      source and destination pair, the X-distance, $\Delta_x = |x_d - x_s|$ and the Y-distance,
      $\Delta_y = |y_d - y_s|$.
Result: The packet reaches destination when S == D.
1 while ( $(x_s \neq x_d) \vee (y_s \neq y_d)$ ) do
2     if ( $(x_s < x_d) \wedge (y_s > y_d) \wedge (\Delta_y > N/2)$ ) then
3         NSe arc is applicable. Keeps on moving in the North direction. Once the North
          boundary is reached, move using the NS channel. Just after taking the NS
          wraparound channel, move one step East for the NSe arc. Follow XY-routing.
4     else if $(x_s < x_d) \wedge (y_s < y_d) \wedge (\Delta_y > N/2)$ then
5         SNe arc is applicable. Keeps on moving in the South direction. Once the South
          boundary is reached, move using the SN channel. Just after taking the SN
          wraparound channel, move one step East for the SNe arc. Follow XY-routing.
6     else
7         Follow XY-routing.
```

The routing algorithm with (SNe + NSe) *arcs* with XY-routing for Torus NoC is presented as Algorithm 1. While comparing source and destination and calculating X-distance and Y-distance, we consider the co-ordinates in the same order as shown in Fig. 1. Since the packets that are destined towards North-East or South-East directions get advantage from using wraparound channels, we term this algorithm as *NE-SE algorithm*. Since all wraparound channels are not utilised, it is not a minimal routing algorithm for Torus. For the packet that uses *arc*, hop count is less than XY-routing and for the packet where *arc* is not applicable, hop count is equal to XY-routing. If an *arc* is applicable for a packet, first the corresponding wraparound channel is traversed. After that, XY-routing is used. The detailed steps are shown in Algorithm 1.

5 Experimental Results and Deadlock-Freedom Analysis

A communicating finite state machine (CFSM)-based simulation framework is used for all experiments in this work [9]. It detects confirmed deadlock with an exact deadlock scenario and reports overall hop counts saved. We consider three routing algorithms for experiments, (1) NSe + SNe + XY-turns (Algorithm 1), (2) EWs + WEn + XY-turns and (3) First Hop with XY-turns [5]. Algorithmic steps for the second algorithm are similar to Algorithm 1. Three types of traffic patterns are

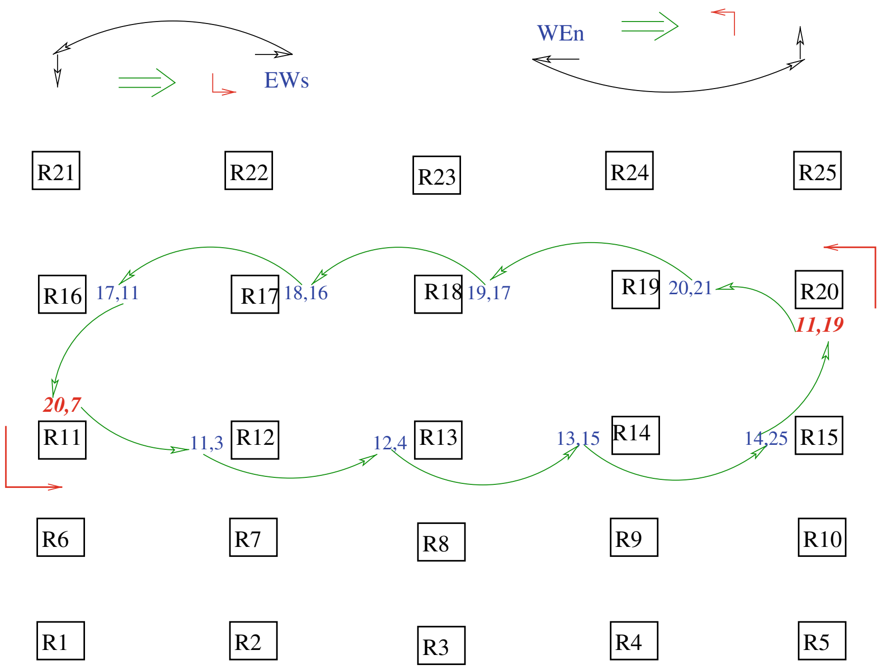

Fig. 5 Experimental Deadlock Snapshot: (EWs + WEn)

used in this work. Uniform traffic and Random permutation traffic are generated using Booksim simulator [16]. Another traffic pattern is generated using the random number function. In Experiment I, deadlock verification is performed. In Experiment II, saving of hop count is compared. All the experiments are performed in a single Intel Core i5 3.20 GHz, 8 GB RAM machine.

In Experiment I, deadlock is reported for the second algorithm (EWs + WEn + XY-turns) with all three types of traffic patterns. No deadlock is detected for the other two algorithms in any experiments performed with Torus NoC of gird size up to 12 × 12. An experimental deadlock snapshot for the second algorithm is depicted in Fig. 5. The CFSM based deadlock detector takes 2641 s to detect the deadlock in a 5 × 5 Torus NoC with random number traffic. An anti-clockwise cycle is formed using SE and NW turns added by (EWs + WEn) *arcs* along with XY-turns. Though SE and NW turns are YX-turns, they are unavoidable in boundary routers after using EWs or WEn *arcs*. The current position of a packet is represented using a number pair that indicates the source and destination of the packet. For example, a packet (17, 11) is currently arrived at East port of router R16 whose source and destination router is R17 and R11, respectively. The deadlock snapshot of the buffer dependency cycle is shown with a green arrow in Fig. 5. From Experiment I, it shows that some *arc* combinations are deadlock prone. A deadlock-free routing algorithms

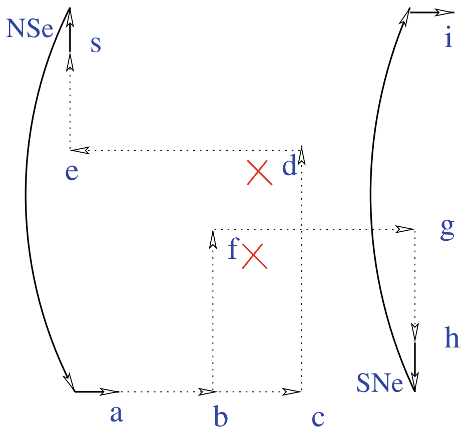

Fig. 6 Deadlock Freedom: (SNe + NSe + XY-Turns)

for Torus without using the dedicated buffer and VC are possible by careful use of *arc* combinations.

Deadlock-Freedom Analysis: CFSM-based framework detects confirmed deadlock. From a theoretical point of view, it does not guarantee the deadlock freedom if the deadlock is not found in a given traffic. Since no deadlock is detected in any experiment, it is likely that Algorithm 1 and First Hop Algorithm [5] are deadlock free. CFSM-based framework takes different times based on the size of the traffic pattern and NoC size to deliver all the packets at their destination router when deadlock is not detected. For the deadlock free (NSe + SNe + XY-Turns) it takes 4093 s to deliver a random traffic of 100000 packets in a 5×5 Torus NoC. An analysis for deadlock freedom is shown with help of Fig. 6. In Algorithm 1, SNe and NSe *arcs* do not create any extra turns besides XY-turns. Therefore, deadlock is not possible as per Turn model. For checking other possible cyclic dependencies involving *arcs*, a dependency graph is shown in Fig. 6. For an anticlockwise dependency cycle formation by NSe *arc*, a NW turn (not permitted) is needed at some vertex d. Similarly, for a clockwise dependency cycle formation by SNe *arc*, a SW turn is needed at some vertex, which is also not permitted. For spreading the dependency from NSe to SNe *arc*, a NE turn is needed at some vertex f, which is not permitted. Therefore, the dependency graph analysis shows that Algorithm 1 is deadlock free.

In Experiment II, Uniform (unf) and Random permutation (rpm) traffic with injection rate 0.05 and 0.08 are used for deadlock-free Algorithm 1 and First Hop Algorithm with XY-turns [5]. No deadlock is detected for any cases. For a packet with given source and destination, *Hop counts saved = (Manhattan distance between source and destination)* - (Actual distance traversed using an algorithm). Percentage of total hop counts saved in uniform and random permutation traffic are shown in Figs. 7 and 8, respectively. For First Hop Algorithm, hop count saving decreases with the increase of NoC size. Whereas, it does not very drastically for Algorithm 1. It shows better savings beyond 10×10 NoC (unf traffic) and 9×9 NoC (rpm traffic) in comparison to First Hop Algorithm.

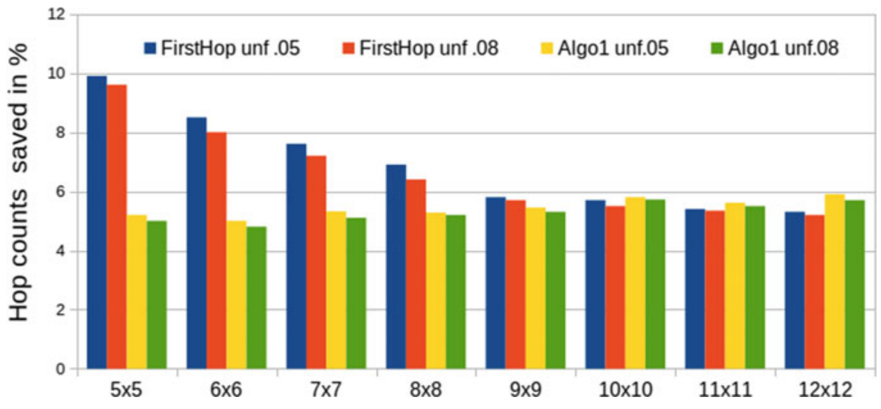

Fig. 7 First Hop Algorithm and Algorithm 1: Uniform Traffic

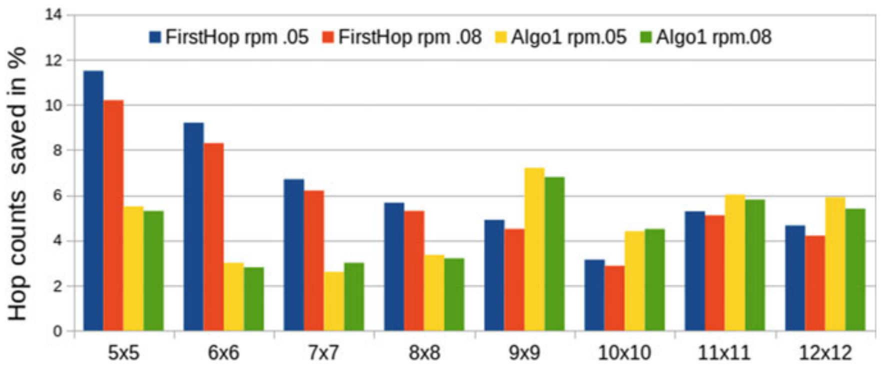

Fig. 8 First Hop Algorithm and Algorithm 1: Random Permutation Traffic

The Effect of the Ration r**:** The ratio r = (*Number of routers present in boundaries/Total routers in a Torus NoC*), decreases as the NoC size grows. Therefore, the percentage of packets injected from boundary routers also decreases as the NoC size grows. In First Hop, wraparound channels are useful only for packets that are injected from boundary routers [5]. Therefore, the effectiveness of the First Hop Algorithm decreases as the NoC size grows. Whereas the r factor does not have adverse effects on *Arc model*-based algorithm. Algorithm 1 uses only two *Arcs*. Therefore, there is a possibility for more effective algorithms by applying more number of mutually deadlock-free *arcs* and turns for Torus NoC, irrespective of NoC size.

6 Conclusion and Future Work

Avoiding deadlock in Torus NoC without using dedicated buffer and VC is presented with *Arc Model* in this work. As an application, a routing algorithm is demonstrated. Deadlock freedom of that algorithm is shown using a buffer dependency graph. Experimental results detect no deadlock and show savings in hop counts. The algorithm is compared with a First Hop algorithm [5] that does not use any additional buffer or VC for deadlock avoidance as well. All eight *arcs* in *Arc Model* are individually deadlock free with XY-turns. Whereas, some *arcs* combinations are deadlock prone. Deadlock is detected for such an *arc* pair and an experimental deadlock snapshot is given. One future direction of our work is to systematically categorise all deadlock-prone *arcs* and deadlock-free *arcs* combinations with respect to given sets of turns besides XY-turns. It would help to develop enhanced algorithms using a different combination of *arcs* and turns. For better traffic distribution and better performance, applying turn distribution concept [10, 11] with *Arc Model* seems promising.

References

1. Z. Yu, D. Xiang, X. Wang, Vcbr: Virtual channel balanced routing in torus networks, in *2013 IEEE HPCC* (2013), pp. 1359–1365
2. V. Puente, C. Izu, R. Beivide, J.A. Gregorio, F. Vallejo, J.M. Prellezo, The adaptive bubble router. JPDC **61**(9), 111 (2001)
3. V. Puente, R. Beivide, J.A. Gregorio, J.M. Prellezo, J. Duato, C. Izu, Adaptive bubble router: a design to improve performance in torus networks, in *ICPP* (1999), pp. 58–67
4. W.J. Dally, B. Towles, *Principles and Practices of Interconnection Networks*, 1st edn. (Morgan Kaufmann, 2003)
5. C.J. Glass, L.M. Ni, The turn model for adaptive routing, in *[JACM*, vol. 41 (1994), pp. 874–902
6. Dally, Seitz, Deadlock-free message routing in multiprocessor interconnection networks. IEEE TC **C–36**(5), 547–553 (1987)
7. J. Duato, A new theory of deadlock-free adaptive multicast routing in wormhole networks, in *IPDPS* (1993), pp. 64–71
8. J. Duato, A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks. *IEEE TPDS* **6**(10) (1995)
9. S. Das, C. Karfa, S. Biswas, Formal modeling of network-on-chip using cfsm and its application in detecting deadlock. IEEE TVLSI **28**(4), 1016–1029 (2020)
10. Ge-Ming. Chiu, The odd-even turn model for adaptive routing. IEEE Trans. Parallel Distrib. Syst. **11**(7), 729–738 (2000)
11. B. Fu, Y. Han, J. Ma, H. Li, X. Li, An abacus turn model for time/space-efficient reconfigurable routing, in *ISCA* (2011), pp. 259–270
12. M. Ebrahimi, M. Daneshtalab, Ebda: a new theory on design and verification of deadlock-free interconnection networks, in *2017 ACM/IEEE 44th ISCA* (2017), pp. 703–715
13. M. Parasar, H. Farrokhbakht, N. Enright Jerger, P.V. Gratz, T. Krishna, J. San Miguel, Drain: deadlock removal for arbitrary irregular networks, in *HPCA* (2020), pp. 447–460
14. A. Ramrakhyani, P.V. Gratz, T. Krishna, Synchronized progress in interconnection networks (spin): a new theory for deadlock freedom, in *ISCA* (2018), pp. 699–711

15. M. Parasar, A. Sinha, T. Krishna, Brownian bubble router: enabling deadlock freedom via guaranteed forward progress, in *NOCS* (2018), pp. 1–8
16. N. Jiang, D.U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D.E. Shaw, J. Kim, W.J. Dally, A detailed and flexible cycle-accurate network-on-chip simulator, in *2013 IEEE ISPASS* (2013), pp. 86–96