Dynamic Stress Wormhole Routing for Spidergon NoC with effective fault tolerance and load distribution

Nishant Satya Lakshmikanth
Department of Computer Science
College of Engineering,Guindy
Anna University,Chennai-600025
sailtosatya@gmail.com

Krishna Kumaar N.I.
Department of Computer Science
College of Engineering,Guindy
Anna University,Chennai-600025
nikrishnaa@gmail.com

Sudha S (co-author), Senior Lecturer
Department of Computer Science
College of Engineering,Guindy
Anna University,Chennai-600025
sudha_s@cs.annauniv.edu

***Abstract*- The rudimentary state of progress of the Networks-on-Chip (NoC) paradigm is a great boon which in turn triggers many to involve in the process of innovation. The optimal Networks-on-Chip paradigm depends on multiple parameters like topology, routing, fault tolerance, load distribution, additional and apt usage of virtual channels, buffer allocation etc. We have designed a Dynamic Stress Wormhole Routing (DSWR) for Spidergon NoC which routes the packets along the shortest path. Incase of faulty nodes or link, our algorithm routes the packets without causing congestion in the entire network. We have compared our results with the other routing techniques and found that DSWR shows improved performance in terms of latency. Furthermore, it is shown that when the network size increases the DSWR for Spidergon architecture performs better than the Mesh architecture with xy routing.**

Keywords - Network-on-Chip(NoC) System-on-Chip(SoC), topology, Dynamic Stress Wormhole Routing, fault tolerance, load balancing, Topology Adjacency Matrix

I. INTRODUCTION

System-on-Chip(SoC) ascribes its basic fundamental building blocks on the efficient NoC switches, which connects SoC's memories and processing elements. NoC, when built to its best level can result in an infrastructure that is robust, scalable, high performing and minimizing area and power consumption. In this paper, a study is made at architectural design, flow control and routing algorithm to overcome permanent and transient fault of links and nodes. At architectural level, Spidergon NoC is selected to maximize the best access to every recess node which also prevents deadlock and channel dependencies [2]. After making many simulations over a wide variety of existing topologies, we finally select Spidergon architecture with the following concrete reasons that are explained at Architectural level.

The well known switching technologies which are in existence are virtual cut-through, store-and-forward and worm-hole switching [1]. Store-and-forward switching makes use of its buffer size to the maximum level. It stores and waits until the whole packet is received at each router before forwarding to the next level. In the case of virtual cut-through switching, the packet is immediately forwarded to the next router and packet delay is reduced. But still if the availability of the next level router fails, then the current router has to store the entire packet. Worm-hole switching divides the packets into small flits and routes in parallel manner thus reducing the packet delay and buffer size. Since our aim is to abridge the buffer size, it is quite easy to discord virtual cut-through and store-and-forward switching and select wormhole routing.

We have computed a novel routing algorithm namely Dynamic Stress Wormhole routing (DSWR),which is explained at Routing level. The architectural design and the routing algorithm are implemented in Nirgam Simulator (NoC Interconnect Routing and Application Modelling) under SystemC[9]. It allows to experiment with various options available at every stage of NoC design: topology, switching technique, virtual channels, buffer parameters, routing mechanism and applications.

II. RELATED WORKS

The non-adaptive routing [6] routes only through the shortest path from source to sink. The best example of this routing is XY-routing [7]. This algorithm routes first via the x-axis followed by y-axis. But in the case of partially-adaptive routing [8] and fully-adaptive routing [9], the virtual channels are used to make it adaptive. But in this non-adaptive, partially-adaptive and fully-adaptive routing there is no assurance that the packets are forwarded during faulty links and nodes.

Glas and Ni proposed the negative-first routing algorithm [10] which can tolerate up to (n-1) faults in the n-dimensional mesh. But the drawback of negative-first is that it uses nil virtual channels. Ye, Benini and Micheli introduced a contention-look-ahead routing algorithm and Li, Zeng and Jone proposed another routing algorithm. In both the case, they route by setting the status of the neighboring switches. But both failed to consider if the neighboring switch is busy or not.

Nicola, Salvatore and Bononi [1] proposed a novel routing algorithm called aEqualised algorithm. This algorithm basically consist of two parts namely aFirst and aLast algorithm. Basically in Spidergon architecture, there exist two paths for all source to sink that could be reached in more than two hops. Taking this as the main fact, the aFirst and aLast algorithm came into existence. But the main drawback in this algorithm is that fault tolerance and load distribution part is not dealt with.

Considering the fully adaptive Fault-tolerant Routing algorithm [2] by Timo, Jochen and Oliver, they have made a better routing algorithm using wormhole switching. But the main drawback lies in the construction of the routing table. For each entry in the table, lots of flit messages are sent and so traffic level within network increases. We replace this algorithm by an alternative DSWR which decreases the flit traffic and makes mathematical computations which is quite faster.

III. OUR CONTRIBUTION

The selection of Spidergon Architecture and making fine tuning over virtual channel allocation, buffer size reduction, switching, forwarding and flow control has given an increased performance even at architectural level. Also the existing Mesh architecture can be easily changed into Spidergon architecture.

Finally the new proposal of DSWR aids in dynamic shortest path routing and unifrom load distribution. We have made a performance oriented comparison of our routing algorithm for different architecture level of mesh, ring and Spidergon NoC.

IV. SPIDERGON ARCHITECTURE

The Spidergon architecture in figure1 engrosses the following characteristics. 1) a regular topology with even number of nodes 2) vertex symmetry 3) each node has a constant number of links (only three) 4) edge transitivity 5) enriched across channels with additional and optimal number of virtual channel [1]. Still increasing the connectivity of nodes with more links may reduce the number of hops from source to sink considerably but power consumption increases (e.g,. Torus).

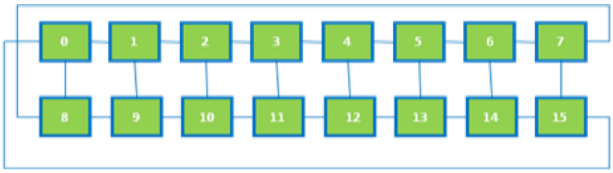

Figure 1: The modified (2 X 8) Mesh Architecture expressing the same Spidergon Architecture for 16 nodes

Figure 1 depicts the conversion of the existing mesh to Spidergon architecture. From the time immemorial, the Spidergon is a ring topology enriched with cross channel. But the same topology can be expressed as (2 X n) mesh with extra two links connecting the top left corner node with bottom right and top right corner with that of bottom left corner.

At each node, we have increased the number of virtual channels [2] is set as three along the across channel. Since wormhole switching is used, increasing the number of virtual channels will be effectively used for the parallel flow control of flits. Also by taking advantage of parallel processing and the smaller size of flits when compared to entire packet, we have reduced the buffer size considerably.

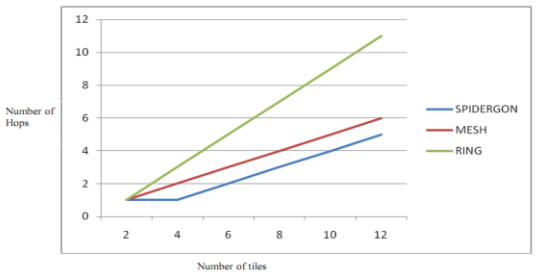

Figure 2: Comparison of Spidergon Noc with Mesh and Ring topologies

In the figure 2 increasing the number of tiles and considering the longest path from some source to destination, the Spidergon NoC clearly confirms the architectural advantage.

V. DYNAMIC STRESS WORMHOLE ROUTING

The very name of this algorithm depicts its various functionalities. The term Dynamic states its flexibility to route even in case of faulty links and nodes and to re-compute new alternate shortest paths. We roll out the next novel term, Stress to address the traffic stress of each node. Using this concept, we maintain an uniform load distribution. Our routing algorithm routes via the shortest path. If one or more nodes or links suddenly go down, an updation is made to make it adaptive fault tolerant routing with uniform load distribution. DSWR uses Wormhole Routing, the main advantage is the small buffer space and data transfer in parallel manner.

Whenever the header flit forwards itself by one hop and reaches the next adjacent node from source, it increments the stress value of that node to the remaining number of flits of that packet. It repeats this process until it reaches the destination switch. In the mean while, when other packet's header flit contacts the nodes with incremented stress value, it omits these node and takes the next adjacent node to uniformly distribute the traffic along the entire network

Building Routing Table

Each node sends the discovery message to all its neighbors and the neighbors will reply with answer message. This information is used for constructing the TAM (Topology Adjacency Matrix) for the Spidergon architecture.

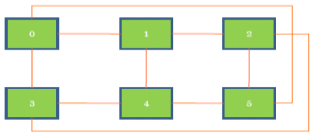

Figure 3: A Spidergon Architecture for (2 X 3) nodes

The adjacency matrix is a 2-dimensional matrix representing each of the adjacent node's link. Its size varies based on the total number of tiles in the Spidergon architecture. The simulation of the Topology Adjacency Matrix (TAM) of the Spidergon architecture in figure 4 is as follows:

Nodes	*0*	*1*	*2*	*3*	*4*	*5*
0	0	1	0	1	0	1
1	1	0	1	0	1	0
2	0	1	0	1	0	1
3	1	0	1	0	1	0
4	0	1	0	1	0	1
5	1	0	1	0	1	0

Figure 4: Topology Adjacency Matrix (TAM)

Each Node has a bit value associated with it as below and is shown in figure 5. This table value is maintained for preventing the usage of node again in the binary tree. Whenever the binary tree is constructed again for finding the next routing table values, the table is refreshed and initialized back to zero for each node.

Node	*1*	*2*	*3*	*4*	*5*
0	0	0	0	0	0

Figure 5: Node Usage Matrix

Initially all the bit values are initialized to '0'.A binary tree is constructed to find the number of hops from each node to every

other node with each level of the tree indicating the number of hops. The bit vector is used so that the node is used only once in the tree construction. Consider the node 0. The routing table for its south dimension is built using the binary tree as shown in figure 5.

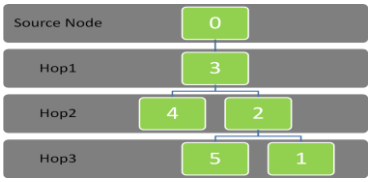

Figure 6: The tree construction hierarchy for finding routing table value for source 0 to all other destination via South channel.

From the figure 6, we can have a model construction of a binary tree for source node '0' via its South channel. This tree is constructed based on the Topology Adjacency Matrix (TAM). If node 0 is the source and node 4 is the sink, then the number of hops required to traverse from node 0 to node 4 is 2. This is because of the fact that node 4 is found at the level 3 and hop 2.

Destination Port	*North*	*East*	*South*	*West*
No. of hops to node 0	2	4	0	2
No. of hops to node 1	1	3	0	1
No. of hops to node 2	1	1	0	3
No. of hops to node 3	0	0	0	0
No. of hops to node 4	2	2	0	4

Figure 7: A partial Routing Table for (2 x 3) nodes

By the construction of single binary tree, we fill all the appropriate values in the routing table starting from level 1 to level n of that binary tree. This increases the efficiency of the mathematical calculation. Figure 7 shows the partially constructed routing table by DSWR algorithm. Since the value corresponding to number of hops of destination 3 is zero, it confirms node 3 is the source and the above table is maintained by node 3. The value of fourth column, corresponding to south is zero which shows that node 3 does not have any South channel.

ROUTING PACKETS

The scalability of the global routing table declines when the number of components increases. Our DSWR algorithm eliminates the concept of global routing table and maintains node level local routing table. The elimination of the global routing algorithm overcomes the single point failure. The first flit is the header flit which contains source and destination address. For the construction of Topology Adjacency Matrix (TAM), an ADDRESS flit is used as DISCOVERY flit. The DISCOVERY flit also checks if the node is busy or not.

The number of flits are drastically reduced such that flits are used only for the construction Topology Adjacency Matrix (TAM). After that once the routing table is constructed by DSWR algorithm, only DATA flits are used within the network. The CONFIRM flits are used for the ordering of flits.

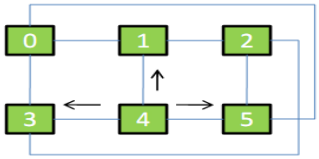

Figure 8: Node 4 sending Address flit to all its neighbors in the 2 X 3 Spidergon NoC

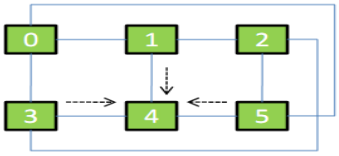

Figure 9: Node 4 receiving Answer flits from all its neighbors (1,3,5) in the 2 X 3 Spidergon NoC

The main advantage of this algorithm is that, the flits transformation is used only up to the construction of Topology Adjacency Matrix (TAM). In Figure 8, the node 4 is sending the ADDRESS flits to all its neighbours. It acts as DISCOVERY flit. Thus after sending the ADDRESS flit, node 4 gets the information of all its neighbours from the received ANSWER flit (figure 9). This same process is being carried on throughout the topology and finally all the nodes share this information which aids in the construction of the Topology Adjacency Matrix (TAM). After the construction of TAM is over, DSWR algorithm mathematically calculates and forms the routing table. After this stage, the routing is done by the Local Routing Table. As far as the routing works in a smooth way, there is no need for the flits to be sent within the network. If any link or node suddenly goes down, the information is updated throughout the topology with the help of flits.

VI. LOAD BALANCING

DSWR algorithm uses a novel concept called Stress value to dynamically distribute the traffic along the entire network. Each link in the topology is associated with a unique Stress value. Initially the Stress value of the link is zero. But as the header flit travels via a particular link, it just increments its Stress value by its Payload length (length of the packet in terms of flits).

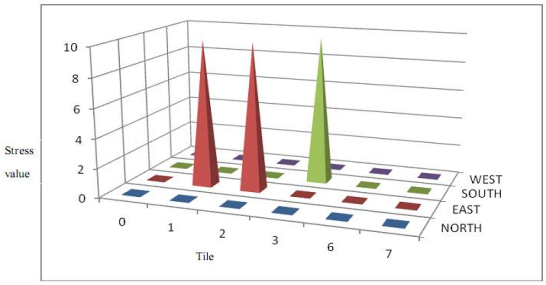

Figure 10: The undistributed traffic channel-wise under random traffic.

Figure 10 shows the undistributed traffic channel-wise under random traffic. But after applying DSWR algorithm, there is a uniform distribution of the traffic along the entire network. Thus the Stress value associated with each link plays a vital role in uniform distribution of traffic.

STRESS = PAYLOADLENGTH - # Sent flits (1)

The formula [4] initially assigns a stress value to each of the link i.e zero. When the header flit travels through link, it increases the stress value to its payload length. Once the path is established and the stress value is constantly decremented whenever a flit of the prior header is sent. The distance from any source to destination depends on the following factors.

Distance = α * Routing Table Value + β * Stress value (2)

where $0 \leq \alpha \leq 1$ and $0 \leq \beta \leq 1$

Depending on the proportion of value to be assigned for the Routing table and Stress value α and β values varies from 0 to 1.

Node	*North*	*East*	*South*	*West*
0	0	5	6	3
1	0	9	5	2
2	0	7	7	9
3	5	9	0	1
4	2	2	0	9
5	5	7	0	4

Figure 11: Stress table for nodes in 2x3 Spidergon NoC

Figure 11 shows the simulation value at some instance under constant bit rate random traffic.

VII. FAULT TOLERANCE

The robustness of the algorithm is established only when the DSWR withstands any kind of wear and tear. For this, we have extended our routing algorithm to efficiently handle to fault tolerance part with minimum number of usage of flits. The flits are needed only to make the minimum change in the TAM. After which the routing table is updated through mathematical calculations. This fault tolerance part handles both the faulty links and switches simultaneously.

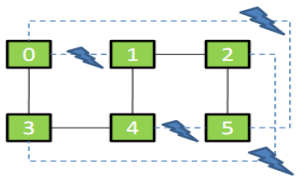

Figure 12: Working of DSWR even in the case of numerous link failure of 2x3 Spidergon NoC.

Figure 12 shows a series of link failures. The faulty links are (0-1), (4-5), (0-5) and (2-3). In spite of three link failures, the algorithm will try to route with the currently available links.

For node 0 as the source and node 5 as the destination, there exists only one path with the links (0-3), (3-4), (4-1), (1-2) and (2-5). Thus DSWR handles faulty links and nodes even under extreme cases.

VIII. PERFORMANCE ANALYSIS

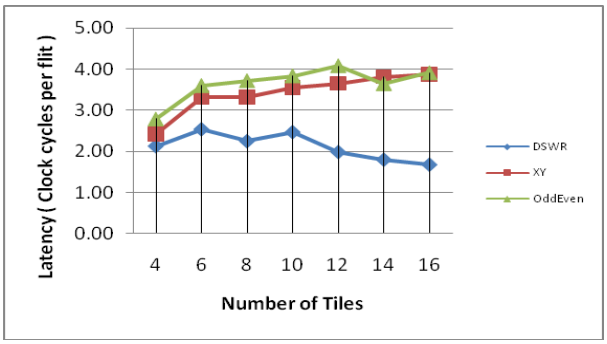

Figure 13 : Comparison of DSWR algorithm with XY and OddEven routing algorithm for Spidergon Architecture.

Figure 13 compares the performance of DSWR, XY and OddEven routing with Spidergon Architecture by varying number of nodes. This clearly shows that when number of nodes in the Network is small, they almost perform equally better but when number of nodes increases DSWR performs better than other two routing algorithms.

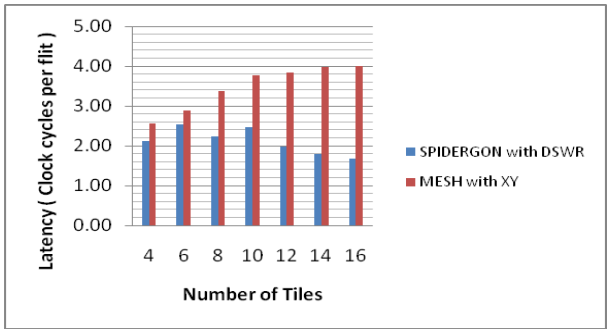

Figure 14: Comparison of Spidergon and Mesh Architecture with their best Routing algorithm.

Figure 14 shows that Spidergon Architecture with DSWR algorithm is best suited if there are more number of nodes in the network.

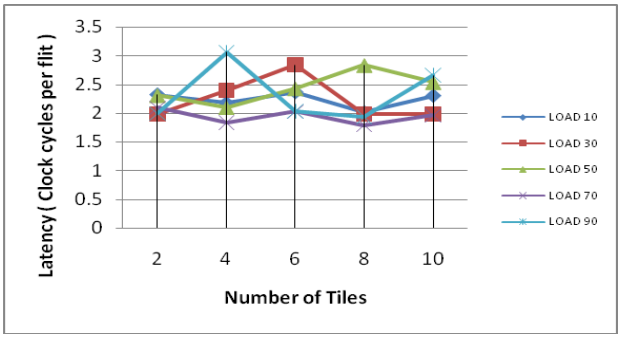

Figure 15 : Behavior of DSWR algorithm by varying Traffic Load (Random Traffic)

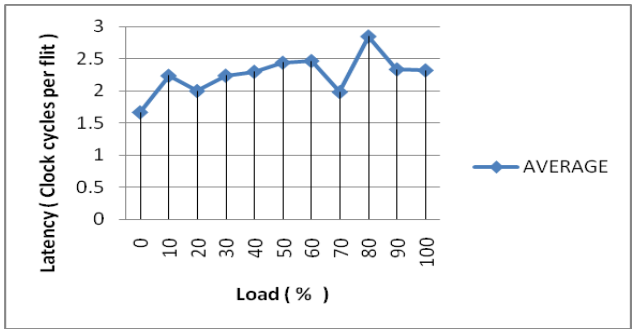

Figure 16 : Average Latency by Varying Traffic Load

Figure 15 shows the behavior of DSWR algorithm by varying Traffic load and number of nodes. Figure 16 shows the average Latency value at each Traffic load. Both these figures clearly depicts that there is no abnormal fluctuation in average Latency value when Traffic load is varied.

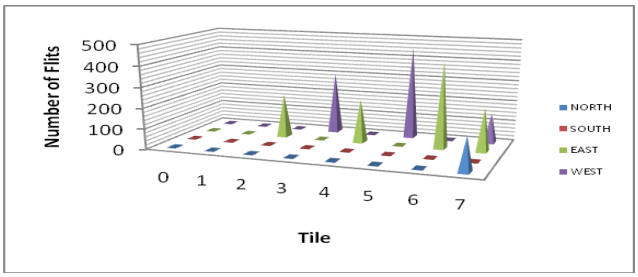

Figure 17: Penalty Table without Traffic Distribution

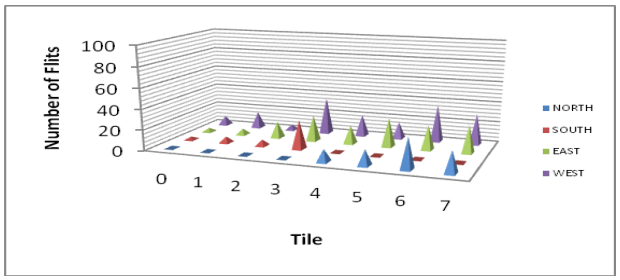

Figure 18: Penalty Table with Traffic Distribution

Figure 17 shows the Penalty table values while using DSWR algorithm without Traffic Balancing. It shows that flits are taking the shortest path always and hence those paths are heavily congested. Figure 18 shows the Penalty table while using DSWR algorithm with Traffic Balancing. It is clear from the figure that flits are distributed so that no particular link is heavily congested.

IX. CONCLUSION

At architectural level Spidergon NoC is chosen which comparatively reduces the number of hops than the others. In the case of routing, DSWR uses minimum number of flits to compute routing information. Since the algorithm mainly relies on mathematical computations, flit traffic is reduced and faster computations are made. Flits are used only for mandatory updations like link or node failure. The load balancing is computed based on stress value of each link. Fault Tolerance is easily guided through minimal change in the TAM followed by routing updation using flit communication. We have compared our results with the other routing techniques and found that DSWR shows improved performance in terms of latency over all the previous existing algorithms.

X. REFERENCES

[1] Jose Duao, Sudhakar Yalamanchili, Lionel M. Li, “Interconnection Networks An Engineering Approach”, Morgan Kaufmann Publishers, 2003

[2] Lih-Hsing Hsu and Cheng-Kuan Lin, “Graph Theory and Interconnection Networks”, CRC Press, 2008.

[3] Nicola Concer, Salvatore lamundo, Luciana Bononi, "aEqualized: a Novel Routing Algorithm For The Spidergon Network On Chip", DATE, 2009.

[4] Timo Schonwald, Jochen Zimmermann, Oliver Bringmann, "Fully Adaptive Fault-Tolerant Routing Algorithm for NoC Architectures",Proceedings of the 10th Euromicro conference on Digital System Design Archictures, Methods and Tools, 2007.

[5] C. J. Glass and L. M. Ni, “The Turn Model for Adaptive Routing,” in ISCA ’92: Proceedings of the 19th annual international symposium on Computer architecture, 1992.

[6] G.-M. Chiu, “The Odd-Even Turn Model for Adaptive Routing,” IEEE Transactions on Parallel and Distributed Systems, vol. 1, no. 7, 2000.

[7] Y. M. Boura and C. R. Das, “Effcient Fully Adaptive Wormhole Routing in n-Dimensional Meshes,” in Proceedings of the 14th International Conference on Distributed Computing Systems (ICDCS), 1994.

[8] A. A. Chien and J. H. Kim, “Planar-Adaptive Routing: Low-Cost Adaptive Networks for Multiprocessors,” Journal of the ACM, vol. 42, no. 1, 1995.

[9] NIRGAM, http://www.nirgam.ecs.soton.ac.uk/.

[10] C. J. Glass and L. M. Ni, “Fault-Tolerant Wormhole Routing in Meshes,” in Proceedings of the 23rd annual International Symposium on Fault-Tolerant Computing (FTCS), 1993.