TRACKER: A Flow History Based Load Balancing Adaptive Router for Mesh NoCs

John Jose, K.V. Mahathi and Madhu Mutyam
Computer Architecture and Systems Laboratory
Department of Computer Science and Engineering
Indian Institute of Technology Madras
Chennai, India 600036
johnjose@cse.iitm.ac.in, mahathi.karnam@gmail.com, madhu@cse.iitm.ac.in

***Abstract*—The effectivesness of an adaptive router in a Network on Chip (NoC) is evaluated by choice of the congestion metric it work with and end results it gives. In this paper, we propose a flit flow based adaptive routing strategy, where every node keeps track of flow of flits through all its ports and exchange this captured values to neighbors. Every router make use of these flit flow estimates in output port selection of incoming flits. Instead of relying solely on availability of free virtual channels across downstream nodes, the proposed models looks up past flit flow analysis and less utilized links are used. TRACKER outperforms the baseline architectures built on odd-even adaptive router model with conventional congestion metrics like free down stream virtual channels, stress values of neighbor switches and buffer availability on neighbors on path. Our experiments with synthetic traffic benchmarks on 16-core, 4x4 mesh NoC show that TRACKER is better than other conventional adaptive routing techniques. When compared with the best locally adaptive routing algorithm, TRACKER exhibits 19%average and 53% maximum latency reduction. The effectiveness of the proposed model is verified with SPEC-2006 multiprogrammed workloads which shows an average packet latency reduction of vv%. Results on larger meshes, longer packets and bursty traffic patterns show that this technique is scalable. The advantages obtained in latency and fairness of link utilization of the proposed model outperforms the negligible overhead incured in additional control logic, area, power and extra wiring.**

I. Introduction

NoC architectures have been proposed to replace design specific global on-chip wiring with general purpose on-chip interconnection network in multicore systems. NoCs are able to offer higher bandwidth compared to traditional bus based communication. NoCs scale better than traditional forms of on-chip interconnect, and enjoy superior performance and fault tolerance characteristics. In an NoC based system, each node is placed in a rectangular tile on the chip. Each tile interfaces with the network over an input port for inserting packets into the network and an output port over which the network delivers packets to the tile [1]. Routers are the basic building blocks of an NoC system. The router attached to a tile consists of five input ports and five output ports; one for each direction and one for local tile. Every input port is associated with a set of input buffers for each virtual channel(VC). Fig-1 helps in a better understanding of a VC-based router. The buffers in the VCs will provide interim storage space for flits once they pass through the router. Head flits upon reaching a router go through route computation and VC allocation [1]. The head flit carries the destination address, which is used by the interim routers to determine the proper output port and VC for that packet. At the arrival of a head flit on a router, the input controller uses the route field to select one of the five output ports. The flit then arbitrates with the other VCs on that input port, and if it wins the arbitration, the flit is forwarded to the output port. Other non-head flits are assigned the same output port and VC as their corresponding head flit [2]. Once the arbitration process is over, the flit is forwarded through the crossbar switch by switch allocation. Winning flits traverse the switching fabric, and thereby the flits are delivered to the proper output ports. Due to simple layout and short wire lengths, the 2D meshes and rings are most widely used in NoC designs [3]. The routing policy used in the system plays a vital role in determining processor performance. In this paper, our proposal is about a novel approach for adaptive routing on 2D mesh topology based on flit flow analysis. Here every node keeps track of flow of flits through all its ports and exchange this captured values to neighbors. Every router make use of these flit flow estimates in output port selection of incoming flits. Instead of using availability of free VCs across downstream nodes [4] as the congestion metric, we introduces a new congestion metric called rate of flit flow in the past across the possible future links. Based on this, routing decisions are taken in such a way that less frequently used links are preferred. The main contributions

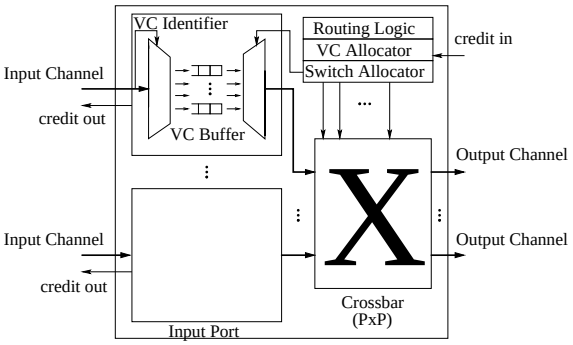

Fig. 1: Baseline router

of work are as follows: A flit flow based, adaptive router called TRACKER for on-chip mesh networks is proposed. The implementation of the router micro architecture details are discussed along with experimental results showing that the logic, storage and bandwidth overheads are within reasonable limits. The average packet latency and link utilization fairness of the TRACKER router is evaluated on a number of real and synthetic workloads. Results shows that TRACKER achieves considerable latency reductions and more fare in link utilization when compared with baseline architectures. During saturation loads, TRACKER achieves a latency reducation of 55% maximum, 23% average over free VC congestion metric and 53% maximum, 19% average over NoP on synthetic workloads in 4x4 mesh networks. TRACKER exhibits an improved fairness factor in link utilization by an average of 4% in normal load to 25% at saturation load across all workloads. The paper is organized as follows. Section 2 provides a review of the related work. Section 3 discuss the TRACKER router and its implementation in detail. Experimental results and comparison with baseline architectures are explored in Section 4, along with overheads incurred by the proposed design. Section 5 concludes with future enhancements planned.

II. Related Work

First generation NoC designs used simple routers employing deterministic routing algorithms [1]. In static routing models, the path between the source and the destination of a packet is predetermined and the current traffic status of the network is not considered. In adaptive algorithms the path between the source and the destination is determined node by node depending on the network status as the packet moves toward the destination[5]. The main advantage of an adaptive routing algorithm is the possibility of routing packets along alternative paths in order to avoid congested nodes. Adaptive routing techniques with reduced packet latency has been a focus of many researchers. Many routing schemes have been proposed in this domain. The odd-even routing models [6] and the turn models [7] are considered as first generation adaptive routing strategies. These methods provide the flexibility of having multiple possible paths between a pair of source and destination, which is not possible in traditional dimension order routing. Both odd-even and turn model are non-minimal routing models which suffered from livelocks under certain conditions [8]. Once there are multiple outgoing ports available for a flit in a router, choosing the best one that optimizes latency and throughput becomes an important design issue. This opened up discussions of criteria in port selections. If multiple ports are available for a flit at a router, traditional odd-even and turn models use the random selection of ports. The proximity congestion awareness [9] uses the stress values of the neighboring switches as the congestion metric. This stress value is sent from one switch to its neighbors in all directions. The stress value of a switch is defined as the number of packets it handles during that cycle. In this approach, flits are forwarded to neighbors having lower stress value. The count of free VCs in the downstream router is also considered as congestion metric [4]. The low latency adaptive router proposed by Kim et al. [10] utilizes a look-ahead congestion detection scheme. Each router keep track of the credits from each neighboring router on the candidate paths. Credits indicate the amount of free buffer space available for each VC. The ant colony based routing [11] make use of separate control packets to capture the traffic flow along various candidate paths. Based on the data collected by control packets the routing table is updated. This is not a promising approach because of the overhead of keeping a router routing table at every node. Moreover the routing table is not updated in real time. The Neighbors-on-Path(NOP) [12] strategy explores the free buffer status of the reachable neighbors of the adjacent routers of the current router. The count of unused buffers at nodes which are two hop away from the current node is obtained. This is used as the metric for port selection. This approach does not guarantee the availability of the input channel of the neighbor of adjacent router when the flit arrives at that node. NoP fails in certain cases, since it chooses port based on non real time, outdated buffer occupancy values. To the best of our knowledge, Regional Congestion Awareness [13] and Destination Based Adaptive routing [3] are the only known adaptive routing algorithms for on-chip mesh networks that uses non-local congestion estimates in route selection. Considering the complexity involved in capturing, computing and propagating non local congestion information, we are not comparing our technique with these two methods. Ours is an approach which make use of data captured by local and neighbor routers. Detailed discussion about the proposed approach is covered in next section.

III. The TRACKER Router architecture

The TRACKER is basically a VC-based router employing odd-even routing strategy. It consists of a VC-Alloc unit, a Routing unit and Switch-Alloc unit like the conventional baseline VC-router [14]. The proposed model incorporates few modifications on the traditional odd-even router model to make it non-minimal and livelock free. Deadlock is avoided in odd-even routing by restricting a packet not allowing a East-North(EN) or East-South (ES) turn in nodes located at even columns. Similarly North-West (NW) and South-West(SW) turns are not allowed on nodes at odd columns. We choose the minimal odd-even as the the basic routing method for our proposed model due to its uniform degree of adaptivity. The TRACKER works as follows. When a packet reaches a router, based on its destination quadrant, the odd-even router computes the outgoing ports related to possible minimal routes. If there are more than one possible outcomes the allocator uses a round robin approach to select one of these selected output ports. The router analyses the flit flow across the reachable output ports [12] of the neighboring routers and associates priority values to the selected output ports of the current router accordingly. The heart of the TRACKER router is a set of counters and associated control logic. Each output port except the ejection port of a router is associated with a pair of saturating counters. The two counters

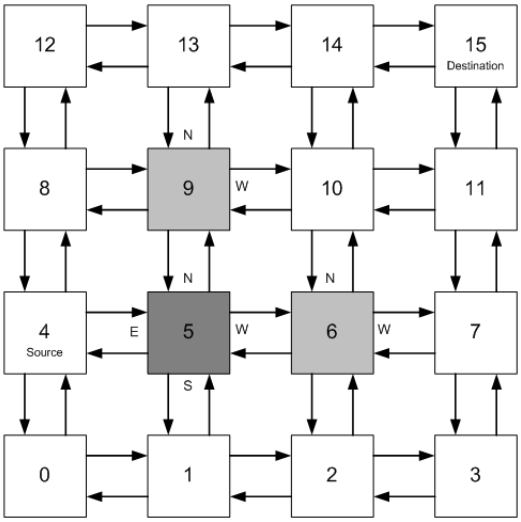

Fig. 2: 4x4 mesh

are of $9bit$ each and are named as *cumulative counter* and *present counter*. The control logic will keep track of flits moving out through the respective ports and updates these counters. The *present counter* value is incremented for every flit flowing through that port. For every T clock cycles, called the *refresh interval*, the *present counter* value is merged with the *cumulative counter* with an appropriate weight factor α typically between 0 and 1. This is done to avoid overflow of *present counter* at higher injection rates. Every router has a flit flow controller, which computes the *weighted-number-of-flits* (WNOF) from the *cumulative counter* and *present counter* as per the following aggregation function. WNOF=(1-α) PC+α CC, where PC and CC represents the *present counter* and *cumulative counter* values respectively. At the end of the *refresh interval* the CC is updated as $CC = CC$+α PC. The WNOF values are exchanged to the neighbors once in every two clock cycles. To reduce the range of value exchanged as WNOF, a range reduction rounding is used so that each WNOF value fits to a 5 bit value. This is done by performing right shift on WNOF 5 times. Every router forwards at-most three such WNOFs values to a router. But which all WNOFs are exchanged to a particular neighbor depends on the physical location of a neighbor with respect to a current neighbor. The WNOF value of the east, north and west ports of a router are forwarded to the neighbor located to its south. This is illustrated in Fig 2. For example, node 9 computes the WNOFs through all its outgoing ports linked to its neighbors, $8, 10, 5$ and 13 using the corresponding counters assigned to those ports. But only WNOFs to ports to $13, 8$ and 10 are forwarded to node-5. similarly WNOFs through ports leading to $8, 5$ and 13 are exchanged to 10. In this manner a node gets the WNOFs values of various ports of all its neighbors. This value is stored in the *status buffers* of a router. When a flit reaches a router through the flit channel, the WNOF values also reaches the router from its neighbors. It takes once cycle for routing and one cycle for VC and switch allocation for a flit in a router. In short a flit spends minimum of 2 cycles in a router before it is forwarded to next outgoing link. Along with route computation, the WNOF values are processed. Based on the candidate paths selected by the routing algorithm, and the possible number of reachable outgoing ports [12] of the first neighbor along candidate path the *mean*-WNOF is computed. The candidate path with lower *mean*-WNOF represents the link which is less frequently used in the past. This path is assigned the highest priority upon the output port selection. Activation signals for VC allocation and switch allocation are generated to facilitate the flit forwarding along this highest priority path.

Now let us examine how the flit forwarding works in a TRACKER router. As per Fig 2, assume a packet P, sourced at 4 and destined at 15 reaches node-5 at clock cycle. The minimal odd-even route function will choose west port(path to 6) and north port (path to 9) as possible routes. In the meantime WNOF values captured from node-9 and node-6 are processed. As per minimal odd-even routing, a flit from node-5 destined at node-15 upon reaching node-9 have two possible out links- north link of 9 to 13 and east link of 9 to 10. Overlooking this information from node-5, the *mean*-WNOF for P, along the north link of node-5 is the average of WNOF through east port and north port of node-9. Similarly the *mean*-WNOF value if the flit could have been routed through node-6 is computed. Note that in this case, the north port of node-6 is not a reachable port for a flit coming from node-5 due to odd-even turn restriction [6], [12]. Out of these possible two options from node-5, the flit is forwarded through the path with smaller *mean*-WNOF. This approach makes sure that flits will be forwarded through less used paths. Results across various synthetic and real traffic patterns shows that this approach not only reduces average packet latency when compared to traditional baseline architectures but also enhances the fairness in link utilization. Detailed results are discussed in next section.

IV. Evaluation

We analyze the performance of the proposed design against a conventional odd-even and other locally adaptive routing techniques in 64 core (8X8 mesh) and 16 core (4X4 mesh). We also examined the sensitivity and response of the system to various parameters proposed in the design like *refresh interval* and weight factor *alpha*.

A. Experimental Setup

We use a cycle accurate network simulator written in C++, that models the two cycle router micro architecture in sufficient detail [14]. We measure the performance of four baseline architectures: Minimal odd-even router [8], locally adaptive router that uses the free VC as congestion metric [4], and Neighbors-on-Path [12]. The router model in the simulator is redesigned as per technique proposed in previous section. In the beginning, the simulator is warmed up under load without taking any measurements until a steady state is reached. Then the next set of packets are labeled and injected. Measurements

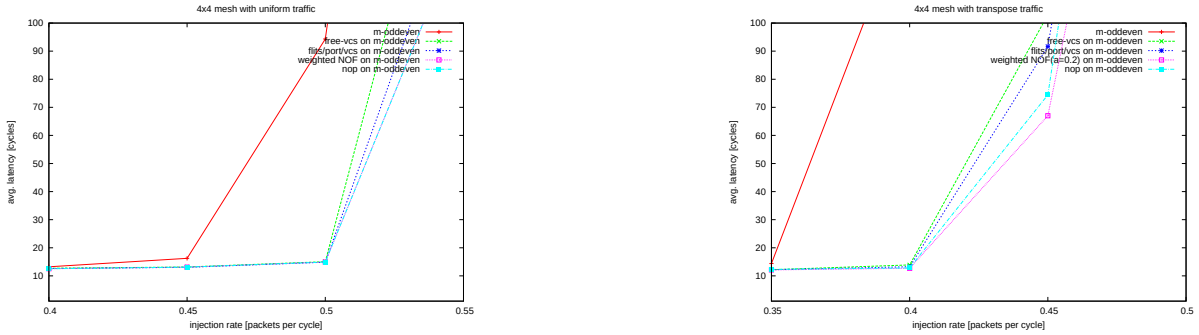

Fig. 3: Average Latency under (a) Uniform Traffic (b) Transpose Traffic 4 VCs per port, VC depth=1 flit, packet size=1 flit

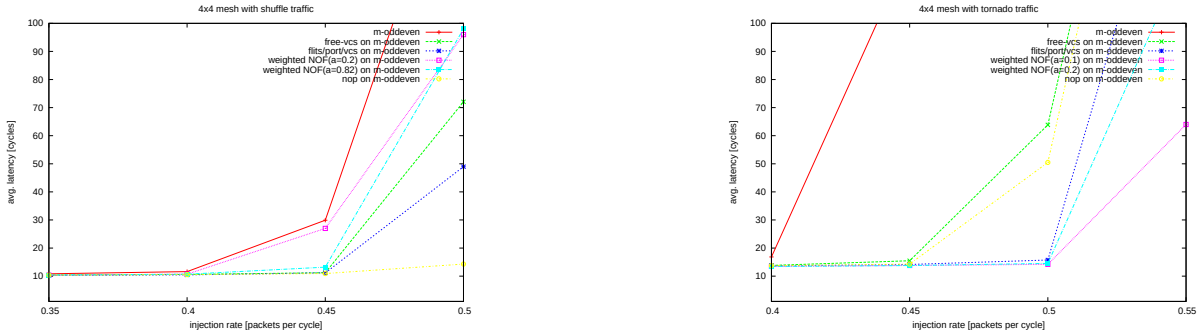

Fig. 4: Average Latency under (a) Shuffle Traffic (b) Tornado Traffic 4 VCs per port, VC depth=1 flit, packet size=1 flit

are taken during the simulation, which runs until these labeled packets get ejected from the system. Latency values and link usage rates are collected for different traffics under various injection rates. The saturation throughput and fairness in link utilization is also analyzed and compared with baseline models.

B. Traffic Pattern

Inorder to simulate an NoC system, we need to generate traffic, which essentially generates packets at regular or non-regular intervals from a source node. We evaluate our proposed technique using four standard synthetic traffic patterns: uniform, transpose, shuffle and tornado. These patterns are useful in providing an insight into the relative strengths and weaknesses of the different congestion metrics and route selection logic techniques under fixed injection rates. Traffic patterns, in which clusters of nodes send data for extended intervals, are very common in multiprocessor applications. Inorder to analyze the performance of the proposed router under dynamically varying injection rates, a bursty traffic model is developed with 20% of total nodes injecting one long packet of 20 flits in every R cycles. We tried with different values of R, to analyze the response of the system under different level of bursty modes. Finally, we evaluated the performance of the system on a trace driven traffic generated by 3 different combinations of SPEC-2006 multiprogrammed workloads which represents a typical CMP scientific workload. The traces were obtained from a 16-node, shared memory CMP system simulator [15]. Configurations in packet injection and generation of packet header are changed on our network simulator to match the environment in which the traces were captured.

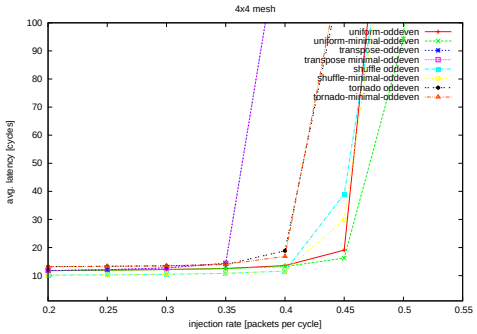

Fig. 5: oddeven and m-oddeven

C. Evaluation of average packet latency

Fig 3 and 4 contains injected load vs average packet latency plots for the TRACKER router compared to local adaptive routing with free VC and NOP-VC as congestion metric. A variant of TRACKER which adopts the ratio of number of flits per port to free VCs as congestion metric is also plotted. In all these four synthetic traffic patterns, TRACKER experiences the minimum latency at saturation throughput. Saturation bandwidth is measured as the point at which the average packet latency is crosses ten times the low load latency(injection

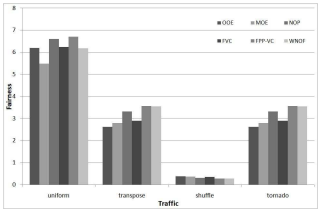

Fig. 6: Fairness in link utilization at IR=0.45

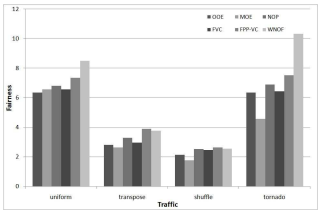

Fig. 7: Fairness in link utilization at IR=0.50

rate =0.2). At saturation load WNOF is having an average of 19% and maximum of 53% latency reduction with respect to the nearest baseline competitor NoP. In all except the shuffle workload the proposed technique considerably extends the saturation injection load to almost 5%.

D. Evaluation of fairness in link utilization

Yet another observation is the improved fairness in the link utilization in the proposed approach. None of the baseline architectures (ref) are considering the flow of flits. Our technique tracks the past flow of flits and choose the route through which are less used, thereby making sure the fairness of load distribution. We computed the total number of flits flowing through each network link under various loads across all traffic patterns. The ratio of average number of flits per link to the average standard deviation of flit count per link is taken as the fairness factor of link utilization. Higher the fairness factor, better is the load distribution across links. Experimental results shows that fairness factor of TRACKER is high compared to baseline architectures and is more predominant at saturation throughputs. Fig.6 and 7 shows the fairness factor of link usage of TRACKER under injection rates near saturation.

E. Sensitivity to various design parameters

One of the key design factor in computing WNOF is the the weight factor α. α is the parameter which determines how significant is the count of flits in the last *refresh interval*. Larger values of α (above 0.5) implies that contents of *cumulative counter* is given more credit. Lower values of α (less than 0.25) indicates that contents of *present counter* is having more credit. We have tried with different values of α from 0.05 to 0.9 across various synthetic traffic patterns. Experimental evaluation shows that for uniform and transpose traffic we got the best latency results for $\alpha = 0.2$. For tornado traffic, even though $\alpha = 0.1$ seems to be slightly better choice than $\alpha = 0.2$, for making alpha uniform across patterns the choice of $\alpha = 0.2$ is made. For shuffle traffic, none of the α values seems to be good enough to keep latency less than NoP model. So we have used $\alpha = 0.2$ as a standard parameter across all loads. All these experimental results showed that for lower values of α, the congestion is effectively tacked. This summarizes the fact that the rate of flow of flits within last 1000 clock cycles have significant impact on out port port selection of neighboring routers.

Another parameter, whose effect is explored is the timing *refresh interval*. Results show that the best latency values are for T=1000. *Present counter* values with low value for T, like 200, 500 and 750 found to be insufficient to distinguish between WNOF passed from adjacent routers. Moreover it will result in saturation of *present counter*. Higher values of T would require a larger *present counter* which will incur more power and area overhead. All these converged to a point that value of *refresh interval* should be large enough to capture the variations of flit flow and small enough so that counter design overhead is minimum.

F. Hardware Overhead

Few hardware enhancements are needed to realize the TRACKER router design. In the tracking and aggregation unit, inorder to track the count of flits moving through the router, two 9 bit saturating counters are used. The aggregation unit consists of a multiplier and adder, which calcualate the WNOF value once in two cycles. We need a shifter to truncate the 10 bit WNOF value to 5 bit. Every router use a status buffer of 15 bits to keep WNOF values passed from a neighboring router. The width of this status buffer may get reduced to 10 or 5 based on whether its neighbor is a corner or an edge router. The status buffer is updated at end of every even cycle. This status buffer is segmented into sections of 5 bit each. Each segment stores the truncated WNOF value of a port. To compute the mean WNOF we need an adder and a shifter. Special control channel is added between the routers to enable the exchange of WNOF values. The width of this channel is varying. It could be $15, 10$ or 5 depending on its physical location.

G. Power Dissipation

..... not yet added.. just sample data... We compare the area and power consumption of the baseline and TRACKER router Orion2.0[16]. We assume a 65nm technology at 2GHz and 128 bit flit channelk and 15 bit control channel. The Predictive Technology Model [17] show that a signal takes two cycles to traverse the channel at 2 GHz. The TRACKER router incurs an extra power ofxx% for the counters, adders, shifters, status buffers and additional control logic kept for tracking and monitoring the flit flow. The control channels incurs nn% of extra power.

V. Conclusion

An adaptive routing algorithm based on the flit flow details from downstream router is proposed. The output port selection takes the advantage of the minimal path that is less frequently

used as per history of flow of flits. Our model makes best use of the link bandwidth and spread traffic across less frequently used links to balance the load. TRACKER, without much additional overhead, gathers non-local flit flow history from neighbors and route packets accordingly. Since the traditional odd-even routing act as back-born of the routing, the deadlock issue is taken care off. The light weight monitoring logic and minimal extra control network ensures that the the small power and area overhead in the proposed design is negligible compared to the latency reduction achieved. Almost across all synthetic workloads and the bursty traffic our proposed design is better in latency and fairness in link utilization than the other baseline architectures. Our experimental results prove that past flit flow could be used as a good congestion metric in future NoC router designs. One of the assumptions we took is that packets are short. Effect of larger packets and effectiveness of this routing method need to be explored in future work. Another area we could explore in future designs is the effectives of the proposed router in other topologies like torus and tree networks.

REFERENCES

[1] W. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *DAC-'01: Proceedings of the Design Automation Conference*, 2001, pp. 684–689.

[2] D. Sanchez, G. Michelogiannakis, and C. Kozyrakis, "An analysis of on-chip interconnection networks for large scale chip multiprocessors," *ACM Transactions on Architecture and Code Optimization*, vol. 7, no. 1, pp. 4:1–4:28, April 2010.

[3] R. S. Ramanujam and B. Lin, "Destination-based adaptive routing on 2d mesh networks," in *ANCS-'10: Proceedings of Symposium on Architecture for Networking and Communications*, June 2010, pp. 194–205.

[4] W. Dally, "Virtual-channel flow control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194–205, March 1992.

[5] A. Agarwal, B. Raton, C. Iskander, and R. Shankar, "Survey of network-on-chip architectures and contributions," *Journal of Engineering, Computing, and Architecture*, vol. 3, no. 1, pp. 1–15, 2009.

[6] G. M. Chiu, "The odd-even turn model for adaptive routing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 7, pp. 729–738, July 2000.

[7] 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*, May 1992, pp. 278–287.

[8] J. Hu and R. Marculescu, "DyAD: Smart routing for networks-on-chip," in *DAC-'04: Proceedings of the Design Automation Conference*, 2004, pp. 260–264.

[9] E. Nilsson, M. Millberg, J. Oberg, and R. Robin, "Load distribution with the proximity congestion awareness in a network-on-chip," in *DATE-'03: Proceedings of the Design, Automation and Test in Europe Conference*, 2003.

[10] J. Kim, D. Park, T. Theocharides, and N. Vijaykrishnan, "A low latency router supporting adaptivity for on-chip interconnects," in *DAC-'05: Proceedings of the Design Automation Conference*, 2005, pp. 559–564.

[11] M. Daneshtalab, A. A. Kusha, and O. Fatemi, "Ant colony based routing architecture for minimizing hotspots in NoCs," in *SBCCI-'06: Proceedings of the 19th Annual Symposium on Integrated Circuits and Systems*, 2006, pp. 56–61.

[12] G. Ascia, V. Catania, M. Palesi, and D. Patti, "Neighbors-on-Path: A new selection strategy for on-chip networks," in *Proceedings of the IEEE-ACM-Workshop on Embedded Systems for Real Time Multimedia*, 2006, pp. 79–84.

[13] P. Gratz, B. Grot, and S. W. Keckler, "Regional congestion awareness for load balance in networks-on-chip," in *HPCA-'08: Proceedings of the International Symposium on High Performance Computer Architecture*, February 2008, pp. 203–214.

[14] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[15] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt, "The M5 simulator : Modeling networked systems," *IEEE Micro Journal*, vol. 26, pp. 52–60, August 2006.

[16] A. B. Kahng, L. Bin, L.-S. Peh, and K. Samadi, "Orion 2.0: A fast and accurate NoC power and area model for early stage design space exploration." in *DATE-'09: Proceedings of the Design, Automation and Test in Europe Conference*, 2009, pp. 423–429.

[17] W. Zhao and Y. Cao, "Predictive technology model for nano-cmos design exploration," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 3, pp. 1–17, April 2007.