Ant Colony Scheduling for Network On Chip

Neda Dousttalab*
Department of Engineering, Tabriz Branch, Islamic Azad University, Tabriz ,Iran
n.dousttalab@gmail.com
Muhammad Ali Jabraeil Jamali
Department of Engineering, Shabestar Branch, Islamic Azad University, Shabestar, Iran
m_jamali@itrc.ac.ir
Ali Ghaffari
Department of Engineering, Tabriz Branch, Islamic Azad University, Tabriz ,Iran
a.ghaffari@iaut.ac.ir

Received: 08/Sep/2013 Revised: 03/Jan/2015 Accepted: 04/Jan/2015

Abstract

It is undeniable that scheduling plays an important role in increasing the network quality on chip.If experts realize the significant of mapping and scheduling in getting rid of delays and increasing performance of these systems, they will ponder over these activities much more scrupulously. The operation scheduling problem in network on chip (NoC) is NP-hard; therefore, effective heuristic methods are needed to provide modal solutions. In this paper, ant colony scheduling was introduced as a simple and effective method to increase allocator matching efficiency and hence network performance, particularly suited to networks with complex topology and asymmetric traffic patterns. The proposed algorithm was studied in torus and flattened-butterfly topologies with multiple types of traffic pattern. For evaluating the performance of the proposed algorithm, specialized simulator network on chip entitled by BookSim working under Linux operation system was used. Evaluation results showed that this algorithm, in many causes, had positive effects on reducing network delays and increasing chip performance compared with other algorithms. For instance, for a complex topologies, this algorithm under maximum injection_rate of up to (10%) increasing throughput have been observed, injection rate, on average, compared to other existing algorithms.

Keywords: On-Chip Interconnection Networks; Switch Allocator; Ant Colony; Scheduling.

1. Introduction

Network-on-chip is an approach for design communication subsystems between IP cores in a system-on-achip (SoC). NoCs can measure synchronous and asynchronous clock domains or use uncloaked a synchronous logic. Networking theory and methods are applied by NoC for on-chip communication and thus notable improvement is made in conventional bus and crossbar interconnections. NoC promotes the scalability of SoCs and power efficiency of complex SoCs compared with other designs [1].

A main agent in specifying network efficiency to network-level design includes topology, routing function, flow control, and router microarchitecture. One parameter which affects the performance of a NoC router is selection of switch allocator and VC allocator.

In general, role of switch allocators is scheduling flits that are buffered at the router's input to pass through crossbar thus; in order to achieve maximum router performance, output quality of matching between resources and requests should be high [2].

This paper is structured as follows: in Section 2, the related works are expressed. Section 3 covers the background including switch allocator, VC allocator, flatted butterfly topology, and ant colony optimization. The proposed scheduler is presented in Section 4. In Section 5, the methodology is explained. In Section 6, simulation of the proposed algorithm is discussed. Conclusion is given in Section 7.

2. Related Works

Max matching method proposed by Hopcroft and Karp [3], due to maximum size matching, does not assure 100% throughput on non-uniform traffic. McKeown et al. performed a more complex maximum weight matching [4]. Various techniques have since been presented to approximate maximum weight matching and reach 100% throughput in [5, 6]. In this approach, although this algorithm guarantees finding a maximum match, it is difficult to parallelize or pipeline and too slow for latency-sensitive application. Nick McKeown et al. [7] presented a scheduling algorithm for input-queued switches which improved RRM by reducing the synchronization of the output arbiters. Anderson et al. [8] proposed a technique called parallel iterative matching (pim), which rapidly detected a conflict-free pairing of inputs to outputs. Wavefront allocator was also developed by Tamir [9]. Owing to the advantage of the router of the Alpha 21364, incremental allocation was used [10,11].

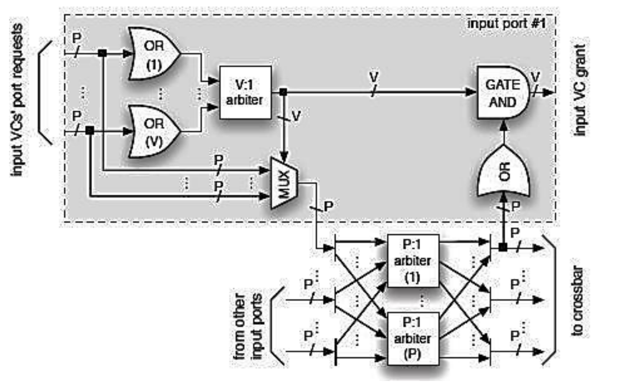
* Corresponding Author

3. Background

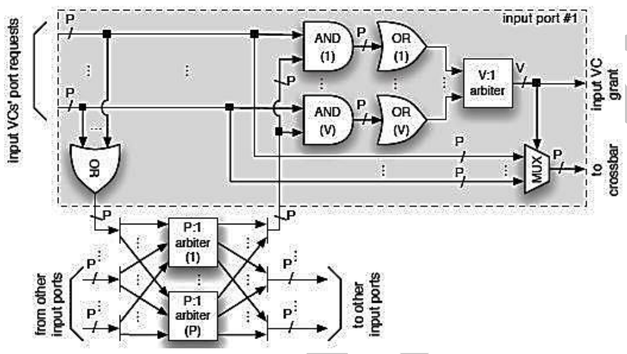
In this section, a brief review of the basic concepts required for this paper is presented.

3.1 Switch Allocator

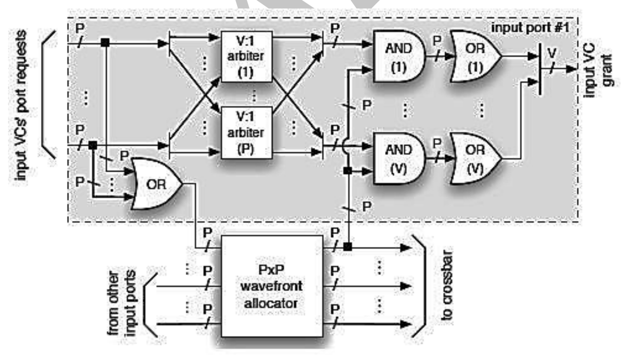
Switch allocation attempts to match the set of requests from the input to the set of available crossbar slot. The aim is that at most one VC at each input port could receive a grant. The block diagram for switch allocator implementation is demonstrated in Fig. 1.



a. Separable input-first allocator (sep if)



b. Separable output-first allocator (sep of)



c. Wavefront allocator (wf)

Fig.1. Switch allocator block diagram [2]

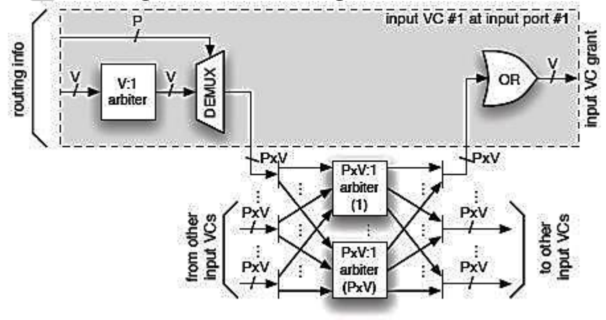
Separable input-first architecture has three stages: pending the first stage one of these active VCs selected as a winner at each input port by arbiter. In the two stages, the winning virtual channel requests are sent to the assigned output ports. Finally, output arbiters generate a grant signal.

For separable output-first allocation, as illustrated in Fig. 1(b), rearbitration is performed among all the requesting input ports that are forwarded to the associated output ports. The probability of selecting a given input by multiple output arbitrations is performed twofold. Will be conducted by one or more ports in the first referee is awarded to an input port and then arbitration performed among all wining virtual channels.

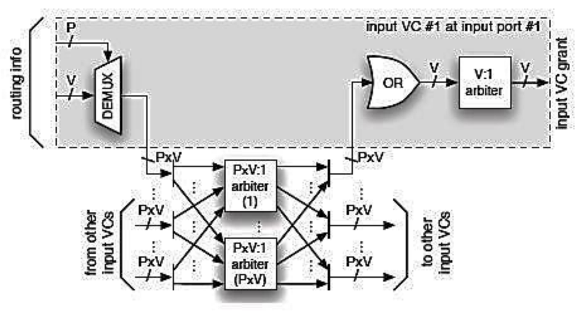
Finally, the wavefront-based implementation, shown in Fig. 1(c), Wavefront allocation, unlike the separable simultaneously does the arbitration between the requests for inputs and outputs. No justice assurances are provided outside, which ensure that all requests are eventually answered [2].

3.2 VC Allocator

Head flits buffered at any of the router's input VCs must be allocated an appropriate output VC at the output port which is selected by the routing function before proceeding through the router pipeline. Hence, the VC allocator implements a matching between P×V requesters and P×V resources under the constraint that any output VCs requested by a given input VC at any given time shares the same output port. Block diagrams for the VC allocator designs is shown in Fig. 2.



a. Separable input-first allocator (sep if)



b. Separable output-first allocator (sep of)

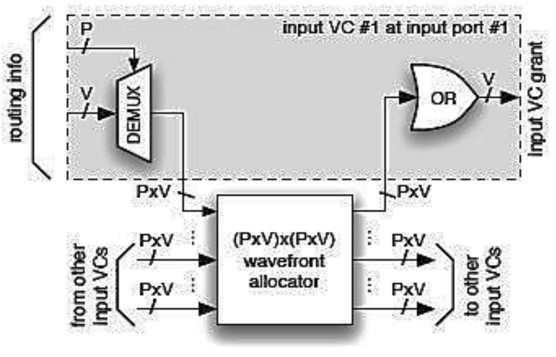


Fig. 2. VC allocator block diagrams [2]

In the separable input-first implementation (which is given in Fig. 2(a)), each input VC first determines the output VC at the destination output port for suggestion. Requests are sent to the stage of P×V-input arbiters at the output VCs as in the canonical implementation; these large P×V–input arbiters can be implemented as three arbiters: A stage of PV-input arbiters in parallel to a single Pin put arbiter selected among them in order to reduce delay. Finally, the grants of each input VC are categorized and reduced to a V–wide vector, which indicates the granted output VC.

The separable output-first implementation is demonstrated in Fig. 3(b). In this figure, each input VC sends requests to all the candidate output VCs at the destination port, in which arbitration is again done between all of the incoming requests. As a given input, VC requests might win the arbitration at multiple output VCs and an additional stage of arbitration is required after categorizing and reducing the grants for selecting a single winning VC.

Finally, the wavefront-based implementation (as depicted in Fig. 3(c)) includes a canonical P×V–input wavefront allocator with an additional logic for generating the P×V–wide request vector for each input VC, like the separable output-first case, and reduction of P×V-wide grant vectors to a V–wide vector like the input-first case [2].

3.3 Flattened Butterfly Topology

Fig. 3 shows flattened butterfly topology that can be derived by combining (or flattening) the routers in each row of a contractual butterfly topology, while securing the inter-router connections. It is structurally like the generalized hypercube; the flattened butterfly significantly reduces the wiring complexity of the topology and more scalable because it condensate in the routers [12].

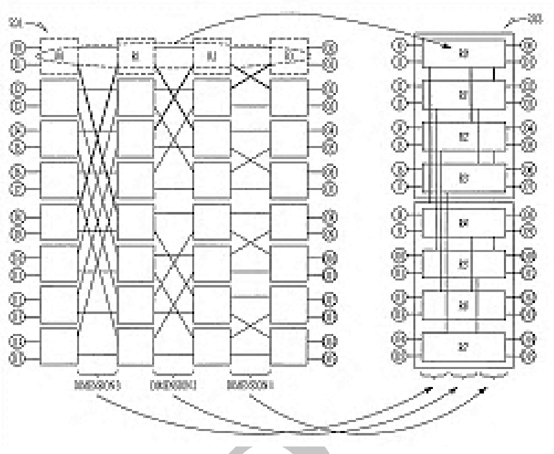


Fig. 3. Block diagram of the flattened butterfly

3.4 Ant Colony Optimization Algorithm

Ant colony optimization algorithm is a probabilistic technique which aimed to solve computational problems; it can be also reduced to finding good paths through graphs. This algorithm is a member of the family of ant colony algorithms in swarm intelligence methods, which constitutes some meta-heuristic optimizations. It was first proposed by Marco Dorigoin 1992 [13, 14].

In the natural world, ants (initially) go randomly around and, after finding the food, they return to their colonies while leaving some pheromone trails. If such a path is found by other ants, they probably do not keep on random travelling and instead follow the same trail, return, and reinforce it in case of eventually finding the food. However, the pheromone trail starts to evaporate over time and, thus, its attractive strength is reduced. The more the time for an ant to travel down the path and come back, the more the time for the evaporation of pheromones would be. By comparison, short paths got marched over more frequently; therefore, the pheromone density becomes higher on shorter paths than longer ones. Pheromone evaporation is advantageous due to avoiding convergence to a locally optimal solution. In the case of no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive for the following ones. In such a case, exploration of the solution space would be limited. Therefore, when one ant finds a good (and short) path from the colony to a food source, other ants probably follow that path and the positive feedback causes all the ants to eventually follow a single path. The idea of ant colony algorithm is to imitate this behavior by "simulated ants" which walk around the graph while showing the problem to be solved [15].

4. Proposed Ant Colony Scheduling Algorithm

In this section, the proposed ant colony algorithm is introduced for solving the scheduling problem. The purpose of this scheduling algorithm is to rapidly find a conflict-free pairing of inputs to outputs, which use a queued cell to transmit between them. This pairing specifies which inputs transmit cells over the crossbar to which outputs in a determined time slot. Ant colony scheduling is a commonly used heuristic, since it generates "good" quality results for large complex problems. The proposed algorithm used separable allocation as a switch allocation. Additionally a dense allocator was applied as a VC allocator. The proposed allocator was based on a basic separable allocator and acted iterant. At each iterance, three steps are performed:

1. Each input port sends a request for using the queue cell.
2. Each output port selects one among all the received requests according to ACO that uses ant colony algorithm for computing priorities.
3. Each input receives multiple grants and accept desirable grant according to ACO and inform to output port after grant by output ports.

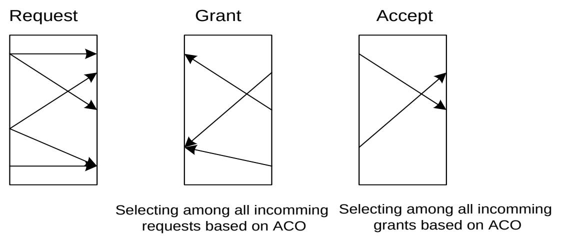

Request
Grant
Accept
Selecting among all incomming requests based on ACO
Selecting among all incomming grants based on ACO

Fig. 4. ACO scheduling: One iteration

To prevent the same selection during each arbitration and eventually eliminate any possible starvation, we perform referee randomly throughout arbitrations.

5. Methodology

Evaluation is performed using BookSim simulator. A 2D torus and flat-fly was used. Each router was connected to one network terminal and all the channels had a single cycle of latency. Deterministic dimension-order routing (DOR) was used for torus and ran-min routing for flat-fly topology. Also, 4 VCs with 8 buffer slots statically assigned to each were applied. Uniform random, random permutation, bit complement, and asymmetric traffic patterns were also employed. Injection rate was in flits.

6. Simulation

As illustrated in Fig. 5, in flat-fly topology with uniform and asymmetric traffic patterns, compared to iSLIP, ant algorithm increased throughput under maximum injection rate up to (12% on average). Ant algorithm offered (12.77% on average) more throughput at maximum injection rate than wavefront and comparable throughput (7.35% on average) more pim and (10.94% on average) more loa.

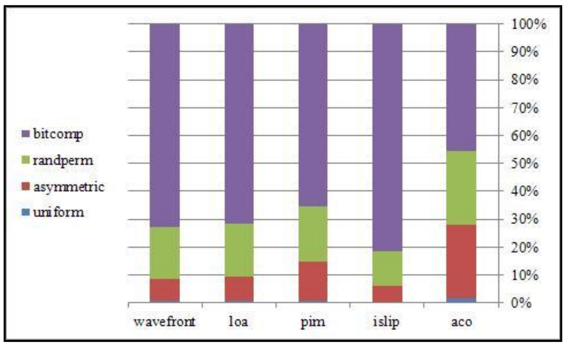


Fig. 5. Throughput comparison of flat-fly for uniform, asymmetric, randperm, and bit complement traffic patterns

Fig. 6 compares throughput by traffic pattern in torus topology. Compared to iSLIP, ant algorithm offered (0.1% on average) higher throughput. Compared to wavefront, throughput increased up to (0.17% on average). Ant algorithm can provide up to 0.14% increase in throughput compared to pim and (0.5% on average) more loa. Although the ant algorithm can provide higher throughput for some traffic patterns, it provides less throughput for bitcomp (bit complement) and randperm, because bitcomp creates some continuous flows of traffic which starve other flows.

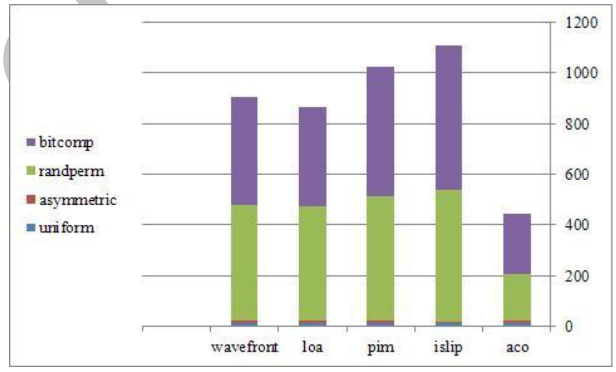


Fig. 6.Throughput comparison of torus for uniform, asymmetric, randperm, and bit complement traffic patterns

In Figs. 7 and 8, throughput of the proposed algorithm is investigated in flat-fly and torus topology with asymmetric and uniform patterns, respectively.

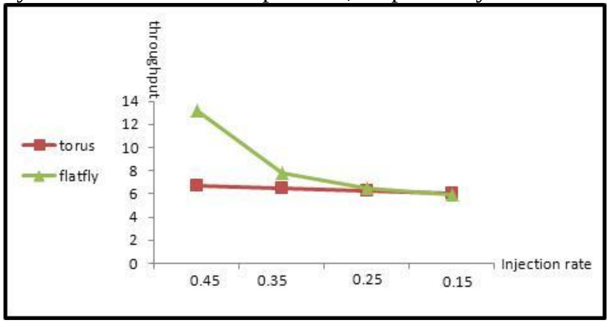


Fig. 7. Throughput comparison of torus and flat-fly across injection in asymmetric traffic

Considering this figure, it can be observed that, in the uniform pattern, the circular mesh had better performance for low and medium traffic; however, flat-fly had higher throughput for high traffics.

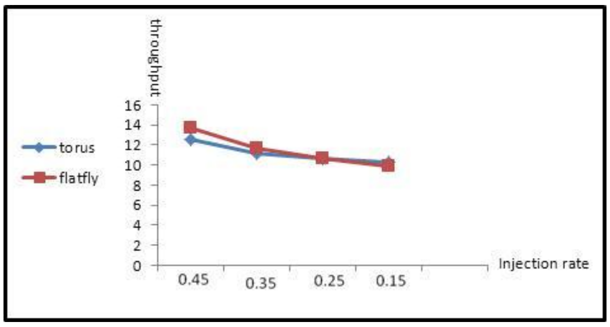


Fig. 8.Throughput comparison of torus and flat-fly across injection rate in uniform traffic pattern

7. Conclusions

Inspired by the nature and collective intelligence of ants, ant colony-based scheduling algorithm in the network was investigated and applied to the chip. We used ant colony approach in our scheduling because of two reasons: first of all the, possibility of employing initial information of the problem is unique in ant colony algorithm the second reason is that fat convergence in obtaining an optimal response and its resistant parameters in initial valuing. The results of this implementation showed that ant colony algorithm is destroyed due to the evaporation of extra pheromone pathways, which prevents ant trapping in local minimums and leads to better algorithm efficiency than the existing scheduling algorithms. Moreover, this algorithm performed at a very good level in most cases due to being smart in the traffic patterns close to real traffic-like asymmetric pattern.

Acknowledgment

This article was extracted from the thesis prepared by Neda Dousttalab to fulfill the requirements for earning the Master's degree. We thank Dr. Bahman Arasteh (Department of Computer Engineering, Islamic Azad University, Tabriz Branch) and Nan Jiang (Electrical Engineering, Stanford University, Stanford) for providing technical support.

References

[1] B. T. W.J. Dally, “Principles and Practices of Interconnection Networks,” Morgan Kaufmann Publishers, San Francisco, CA, 2004.
[2] D. U. B. W.J.Dally, “Allocator Implementations for Network-on-Chip Routers,” In Proceedings of the 2009 ACM/IEEE Conference on High Performance Computing, Networking, Storage and Analysis, 2009.
[3] J. E. H. a. R. M. Karp, “An n5/2 algorithm for maximum matching in bipartite graphs,” SIAM Journal on Computing, vol. 2, pp. 225-231, 1973.
[4] A. M. N.McKeown, V.Anantharam ,and J.Walrand, “Achieving 100% throughput in an input-queued switch,” In Proc. of IEEE INFOCOM, pp. 296–302, March, 1996.
[5] L.Tassiulas, “Linear complexity algorithms for maximum throughput in radio networks and input queued switches,” In Proc. of IEEE INFOCOM, pp. 533–539, 29 Mar-2 Apr 1998, 1998.
[6] B. P. P. Giaccone, and D.Shah, “Towards simple high performance schedulers for high-aggregate bandwidth switches,” In Proc. Of IEEE INFOCOM, pp. 1160–1169, June, 2002.
[7] N. McKeown, “The iSLIP Scheduling Algorithm for Input-Queued Switches,” IEEE/ACM TRANSACTIONS ON NETWORKING, vol. 7, April, 1999.
[8] S. S. O. T. E. Anderson, J.B.Saxe, and Ch.P. Thacker, “High speed switch scheduling for local area networks,” ACMTransactions on Computer Systems, pp. 3, 352-19, Novamber, 1993.
[9] Y. T. a. H.C.Chi, “Symmetric crossbar arbiters for VLSI communication switches,” IEEE Transactions on Parallel and Distributed Systems, pp. 13–27, January, 1993.
[10] F. S. S. S. Mukherjee, P.Bannon, J.Emer, S.Lang, and D. Webb“ ,A comparative study of arbitration algorithms for the Alpha 21364 pipelined router,” In Proc of. Architectural Support or Programming Languages and Operating Systems (ASPLOS), pp. pages 223–234, octobr, 2002.
[11] P. B. S.S. Mukherjee, S.Lang, A.Spink ,and D.Webb, “The Alpha 21364 network architecture,” In Proc. of the Symposium on Hot Interconnects, pp. 113–117, august, 2001.
[12] J. B. J. Kim, and W. Dally, “Flattened Butterfly Topology for On-chip Networks,” In International Symposium on Microarchitecture, pp. 172–182, December, 2007.
[13] M.Dorigo, “ant colony optimization”.
[14] G. D. C. a. L. G. M.Dorigo, “Ant Algorithm for Discrete Optimization,” Artificial Life, vol. 5, pp. 137-172, 1999.
[15] S. S. a. S. N. S. P.Mathiyalagan “Modified Ant Colony Algorithm for Grid Scheduling,” IJCSE) International Journal on Computer Science and Engineering, vol. 2, 2010.

Neda Dousttalab received the B.Sc degree in Computer Engineering from the Islamic Azad University, Shabestar Branch, Shabestar, Iran, in 2008 and the M.Sc degree in Computer Engineering from Islamic Azad University, Tabriz Branch, Tabriz, Iran, in 2013. Her research interests include multiprocessor systems-on-chip, networks on chip and Fault tolerance

Mohammad Ali Jabraeil Jamali received B.Sc degree in Electrical Engineering from Urmia University, Urmia, Iran, the M.Sc degree in Electrical Engineering from Tabriz University, Tabriz, Iran, the M.Sc degree in Computer Engineering from Islamic Azad University, Science and Research Branch, Tehran, Iran and the Ph.D degree in Computer Engineering from Islamic Azad University, Science and Research Branch, Tehran, Iran, in 1994, 1997, 2003 and 2009, respectively.
He is an assistant professor of Computer Engineering at Islamic Azad University, Shabestar Branch. He is the author/co-author of more than 50 publications in technical journals and conferences. His current research interests include processor and computer architectures, chip multiprocessors, multiprocessor systems-on-chip, networks on chip, ad hoc and sensor networks.

Ali Ghaffari received the B.Sc degree in Computer Engineering from the University of Tehran, Tehran, Iran, in 1994 and the M.Sc degree in Computer Engineering from the University of Tehran, Tehran, Iran, in 2002 and the Ph.D degree in Computer Engineering from the Islamic Azad University, Science and Research Branch, Tehran. His research interests include information security, wireless networks, and ad hoc networks.