Application of Advanced Routing Schemes for Energy Consumption Optimization in NoC-Based Ethernet Switches

Ile Dimitrievski, Valentin S. Mollov

Department of Computer Systems, Technical University of Sofia, 8 Kl.Ochridsky Blvd, 1000 Sofia, Bulgaria, ph:+359 2 965 3523, e-mail: ile.dimitrievski@gmail.com; mollov@tu-sofia.bg

Abstract: *This paper presents an evaluation of the energy consumption obtained by the application of the smart Dynamic Adaptive Deterministic (DyAD) routing algorithm with respect to the most popular XY algorithm which is not an energy-saving one. DyAD routing algorithm for Network on Chip (NoC) combines the advantages of both deterministic and adaptive routing algorithms. The comparison between the above algorithms is performed by means of the NOXIM simulator with examples on Torus topology.*

Keywords: *smart routing algorithm, XY, DyAD, energy consumption, NOXIM*

1. Introduction

The state-of-the art Network on Chip (NoC) can integrate numerous IP cores: DSP unit, RAM, soft-cores processors, etc. These blocks communicate each other by applying different routing algorithms (schemes). The data exchange between the network nodes is performed through dedicated routers labelled by Ri.

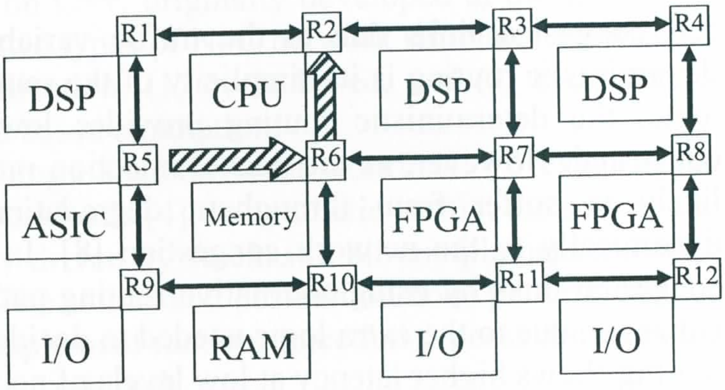


Fig. 1. XY routing scheme on Torus NoC topology

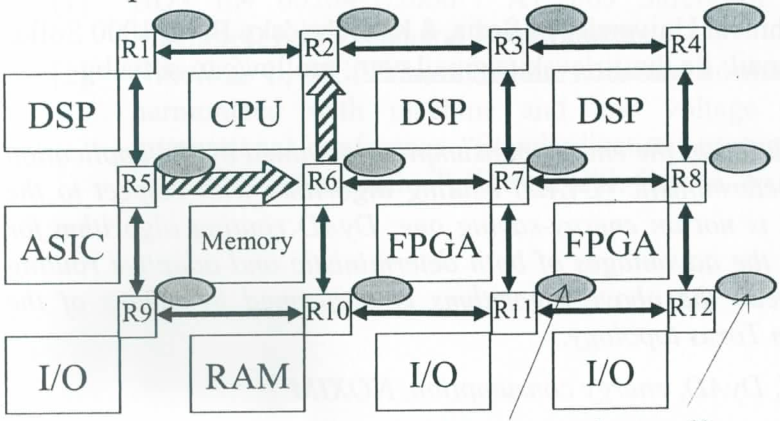
In this study, we present two different routing algorithms used in routers integrated on NoCs. The first algorithm that will be evaluated is the simplest and most frequently used routing algorithm, named XY, while the second one is the most advanced routing scheme called Dynamic Adaptive Deterministic (DyAD).

2. Routing Schemes. Problem Formulation

Basically, the most popular XY routing algorithm applies a simple principle, as shown in the example on Fig.1 implemented to the Torus NoC topology [7]. Packet from the *source router R5* must be received on the *destination R2 router*, so the packet should travel firstly by X axis and will be received on R6 router. Data flow paths are shown with dash-line wide arrows on both Fig. 1 and Fig. 2. The router R6 will receive packet from router R5 and then the received packet will be redirected on

Y axis pointing to the R2 router. Finally, the packet sent from the source router R5 will be received on the destination router R2.

The second algorithm that we evaluate here is the Dynamical Adaptive Deterministic switching [2], shown in Fig. 2. Actually, this algorithm is an extension of the Adaptive routing scheme. The adaptive algorithms are working on finding the shortest path.



Monitors the traffic

Fig. 2. DyAD routing on Torus NoC topology

Routers Ri can be generally classified as *deterministic* and *adaptive*. In deterministic routing (also called oblivious routing), the path is completely determined by the source and the destination addresses.

On the other hand, a routing technique is called adaptive if, given a source and a destination addresses, the path taken by a particular packet depends on dynamic network conditions, e.g. congested links due to the traffic variability. The main advantage of using deterministic routing is its simplicity of the routers design. Because of the simplified logic, the deterministic routing provides low latency when the network is not congested. However, as the packet injection rate increases, deterministic routers are likely to suffer from throughput degradation, as they are unable to respond dynamically to the network congestion [8]. In contrast, the adaptive routers avoid congested links by using alternative routing paths which leads to higher throughput. However, due to the extra logic needed to decide on a good routing path, the adaptive routing shows higher latency at low levels of network congestion.

Currently, the most frequently used scheme integrated in the routers Ri is the XY algorithm. XY algorithm was suggested for basic communication in the computer networks, so it has been optimised to be used in NoCs. Some of the advantages of the XY routing scheme is that it is deterministic and because of its relative simplicity.

Obviously, XY routing scheme is a minimal path routing algorithm and is free of deadlock and livelock [1]. Unlike deterministic routing, where the routing path is fixed once the source and the destination addresses are given, the adaptive routing offers packets more flexible in choosing their routing paths, if multiple routing paths exist. However, when using adaptive routing, caution must be taken in order to solve the deadlock problem, which may be caused by packet waiting for each other in a cycle.

Starting from these observations, we apply routing algorithms that require no virtual channels for NoC. To be deadlock free, the routing algorithm needs to prohibit at least one turn in each of the possible routing cycles. In addition, in order to preserve its adaptivity, it should not prohibit more turns than necessary [6].

In this paper we propose an application of the DyAD algorithm for NoC. The advantage of this algorithm is that it combines both deterministic and adaptive routing schemes. While using DyAD routing algorithm, each router Ri in the NoC monitors its local network load and make decisions based on this information. When the NoC network is not congested, a DyAD router works in deterministic mode, thus enjoying the low routing latency enabled by deterministic routing. On the contrary, when the network becomes congested, the DyAD router switches back to the adaptive routing mode and thus avoids the congested links by exploration other routing paths. It leads to higher network throughput which is highly desirable for NoC-based applications. To propose a valid approach, we also show how to obtain freedom from deadlock and livelock [1] that can be guaranteed when mixing deterministic and adaptive routing modes into the same NoC.

3. RESULTS

3.1. Simulator overview

We used the *Noxim* simulator [3], [5] to calculate the average latency, average throughput, and dynamically consumed energy. The simulator is written in SystemC, a system description language based on C++, originally developed at the University of Catania, Italy. Basically, there are two types of NoC network simulator approaches to design the simulator software application: (1) cycle based, and (2) event-driven simulators. In our case, we use a cycle based simulation.

We will evaluate Torus NoC topology after different workload. To evaluate the NoC performance we use synthetic workloads [4]. In synthetic traffic, the source and destination node patterns are typically driven by stochastic bit-complement injection process. The spatial characteristics of the source and destination node patterns and the temporal characteristics of the bit-complement random injection process are intended to model the characteristics of realistic workloads.

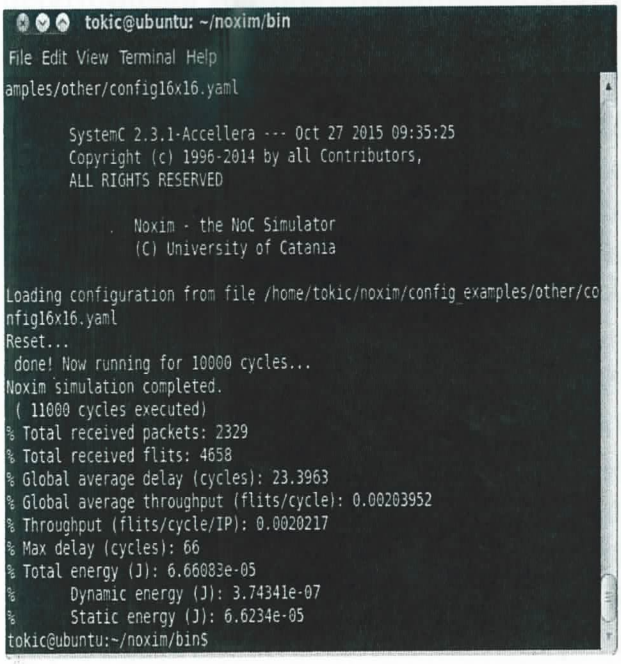
So, there are different types of synthetic patterns used in NoC design includes the following cases about the traffic loads: (1) *Uniform Random*: Here, the source and destination nodes are chosen via a uniform random process. The load is balanced, because the source and destination coordinates are uniformly distributed, although the random process that generates the coordinates may cause transient hotspots; (2) *Transpose*: In transpose traffic, the destination coordinates are the transpose of the source coordinates. Under this load the network's diagonal bisection is a bottleneck as all packets must cross it. The transpose traffic, in combination with the Dimension Router Ordering (DOR) algorithm commonly found in NoC's, produces a highly imbalanced network load. In transpose traffic, the links located counter-clockwise from the centre of the mesh are utilized while he clockwise links remain unused; (3)*Bit-complement*: In this traffic load, the destination coordinates are the bit-wise inversion of the source coordinates. This load stresses the horizontal and vertical network bisection. Under this load DOR statically spreads traffic across all of the bisectional links, providing a perfectly balanced network load.

3.2. Simulator scenarios

Here, the simulations are performed using the *Noxim* simulator. As it has been mentioned above, we examine just Torus topology as it is among the most known and frequently used for NoC. Different scenarios are applied for simulations, which are given as follows:

1. Torus k=16, n=2 topology, e.g. 16x16 was evaluated for average throughput and average latency with *bit-complement traffic* under *fixed packet size* and *variable injection rate.* XY routing algorithm and DyAD routing algorithm are used in this case.
2. Torus with k=16, n=2 topology e.g. 16x16 was evaluated for global average throughput and global average latency with *bit-complement traffic patterns* under *fixed packet size* under *different injection rate.* XY routing algorithm and DyAD routing algorithm are used here.
3. Torus with k=16, n=2 topology e.g. 16x16 topology was evaluated for energy consumption and average throughput with *bit-complement traffic patterns, fixed packet size*, and *variable injection rate.* XY routing algorithm and DyAD routing algorithm are used in this case.

Sample Noxim simulation outputs for Torus 16x16 XY and DyAD routings are shown in Fig. 3.

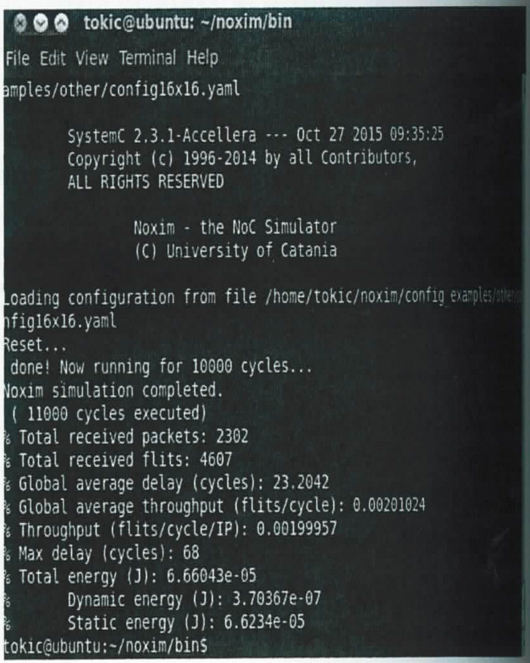


```
amples/other/config16x16.yaml

        SystemC 2.3.1-Accellera --- Oct 27 2015 09:35:25
        Copyright (c) 1996-2014 by all Contributors,
        ALL RIGHTS RESERVED

                 Noxim - the NoC Simulator
                 (C) University of Catania

Loading configuration from file /home/tokic/noxim/config_examples/other/co
nfig16x16.yaml
Reset...
 done! Now running for 10000 cycles...
Noxim simulation completed.
 ( 11000 cycles executed)
% Total received packets: 2329
% Total received flits: 4658
% Global average delay (cycles): 23.3963
% Global average throughput (flits/cycle): 0.00203952
% Throughput (flits/cycle/IP): 0.0020217
% Max delay (cycles): 66
% Total energy (J): 6.66083e-05
%       Dynamic energy (J): 3.74341e-07
%       Static energy (J): 6.6234e-05
tokic@ubuntu:~/noxim/bin$
```



```
amples/other/config16x16.yaml

         SystemC 2.3.1-Accellera --- Oct 27 2015 09:35:25
         Copyright (c) 1996-2014 by all Contributors,
         ALL RIGHTS RESERVED

                  Noxim - the NoC Simulator
                  (C) University of Catania

Loading configuration from file /home/tokic/noxim/config_examples/other/co
nfig16x16.yaml
Reset...
 done! Now running for 10000 cycles...
Noxim simulation completed.
 ( 11000 cycles executed)
% Total received packets: 2302
% Total received flits: 4607
% Global average delay (cycles): 23.2042
% Global average throughput (flits/cycle): 0.00201024
% Throughput (flits/cycle/IP): 0.00199957
% Max delay (cycles): 68
% Total energy (J): 6.66043e-05
%        Dynamic energy (J): 3.70367e-07
%        Static energy (J): 6.6234e-05
tokic@ubuntu:~/noxim/bin$
```

Fig. 3. Sample sim outputs Torus 16x16: XY routing (left), DyAD routing (right)

3.3. Numerical results. Major parameters calculation

Below, the most important NoC parameters are calculated and shown as plots to make a comparison between the XY and DyAD routing algorithms.

Fig. 4 gives the plots of the offerted traffic vs. the average latency of the netwotk, while Fig. 5 presents the global throughput vs. global average latency for Torus 16x16 topology. Both the plots on Fig. 4 and Fig. 5 are identical due to the physical proximity of the measured parameters.

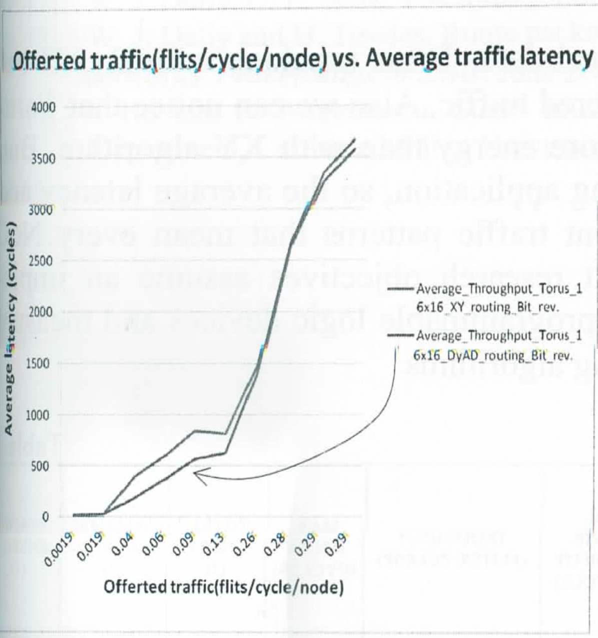


Fig. 4. Offerted traffic vs. average latency vs. Torus 16x16

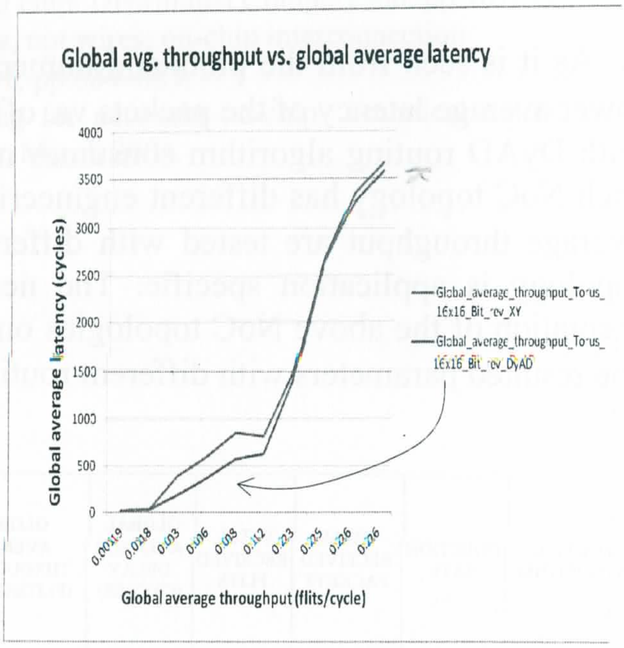


Fig. 5. Global average throughput global average latency Torus 16x16

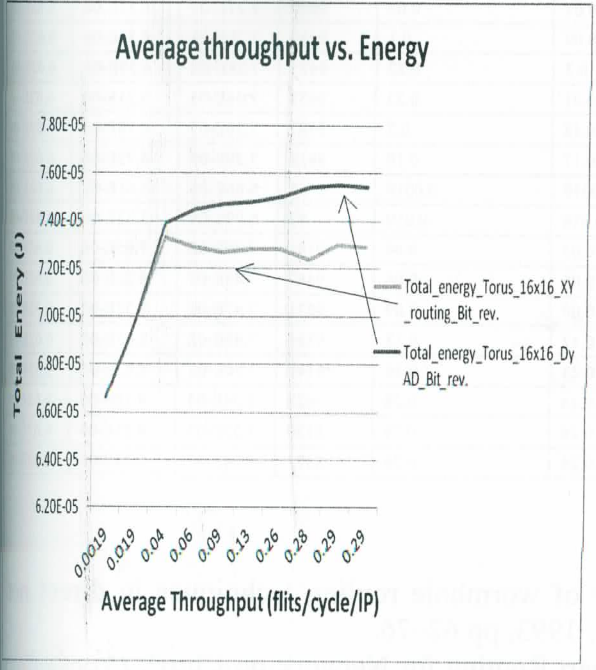


Fig. 6. Total energy vs average througput for XY and DyAD routing schemes (Torus 16x16)

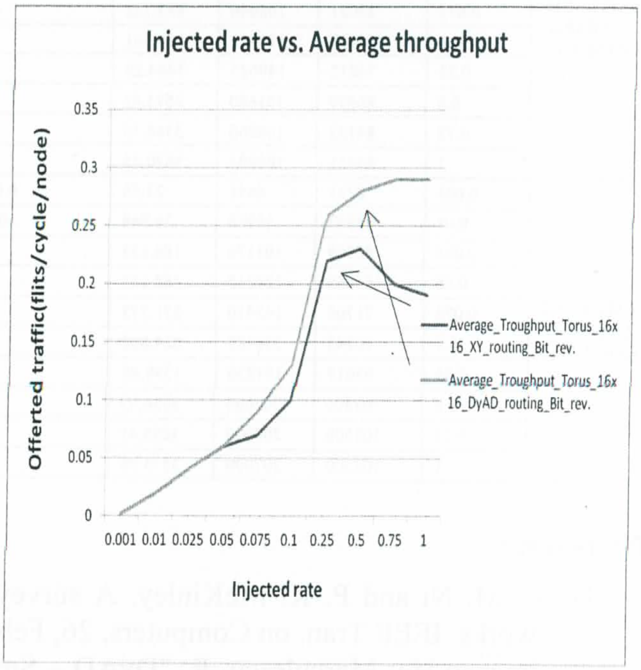


Fig. 7. Offerted traffic vs. injected rate for XY and DyAD routing schemes (Torus 16x16)

The Fig. 6 plot gives the total energy vs average througput and Fig. 7 – the offerted traffic vs. the injected rate of the network for the Torus 16x16 case.

Table 1 represents numerical data for all major network parameters, obtained as a result from the simulation with Noxim. As for the above plots, the simulations have been run for Torus 16x16 topology and both routing algorithms: XY and DyAD.

4. Conclusion

As it is seen from the plots and numerical data, Torus topology with DyAD has lower average latency of the packets vs. offered traffic. Also we can notice that Torus with DyAD routing algorithm consumes more energy than with XY algorithm. But, each NoC topology has different engineering application, so the average latency and average throughput are tested with different traffic patterns that mean every NoC topology is application specific. The next research objectives assume an implementation of the above NoC topologies on programmable logic devices and measure the resulted parameters with different routing algorithms.

Table 1

ROUTING ALGORITHM	INJECTION RATE	TOTAL RECEIVED PACKETS	TOTAL RECEIVED FLITS	GLOBAL AVERAGE DELAY (CYCLES)	GLOBAL AVERAGE THROUGHPUT (FLITS/CYCLE)	TROUGHPUT (FLITS/CYCLE/IP)	MAX DELAY (CYCLES)	TOTAL ENERY (J)	DYNAMIC ENERGY (J)	STATIC ENERGY (J)
TRAFFIC BIT REVERSAL Routing XY	0.001	2302	4607	23.2	0.002	0.0019	68	6.66E-05	3.70E-07	6.62E-05
	0.01	22711	45420	25.332	0.017	0.019	147	6.99E-05	3.67E-06	6.62E-05
	0.025	45348	90699	393.821	0.03	0.04	8180	7.33E-05	7.09E-06	6.62E-05
	0.05	50246	100493	595.666	0.05	0.06	9869	7.29E-05	6.75E-05	6.62E-05
	0.075	54021	108046	845.722	0.07	0.07	9630	7.27E-05	6.53E-06	6.62E-05
	0.1	56477	112953	814.561	0.09	0.1	9439	7.28E-05	6.56E-06	6.62E-05
	0.25	74812	149625	1484.22	0.2	0.22	9475	7.28E-05	6.59E-06	6.62E-05
	0.5	80839	161680	2673.62	0.21	0.23	9653	7.24E-05	6.21E-06	6.62E-05
	0.75	84133	168266	3344.15	0.18	0.2	9787	7.30E-05	6.78E-06	6.62E-05
	1	84451	168903	3670.25	0.17	0.19	9812	7.29E-05	6.72E-06	6.62E-05
TRAFFIC BIT REVERSAL Routing DYAD	0.001	2221	4441	23.56	0.0019	0.0019	65	6.66E-05	3.61E-07	6.62E-05
	0.01	22952	45908	24.948	0.018	0.019	82	6.99E-05	3.73E-06	6.62E-05
	0.025	50589	101176	186.133	0.03	0.04	9181	7.39E-05	7.65E-06	6.62E-05
	0.05	64858	129715	368.165	0.06	0.06	9487	7.45E-05	8.25E-05	6.62E-05
	0.075	71706	143410	571.775	0.08	0.09	9631	7.47E-05	8.52E-06	6.62E-05
	0.1	76743	153483	624.802	0.12	0.13	9354	7.48E-05	8.65E-06	6.62E-05
	0.25	95619	191236	1398.88	0.23	0.26	6147	7.50E-05	8.82E-06	6.62E-05
	0.5	103500	207000	2636.93	0.25	0.28	7623	7.54E-05	9.18E-06	6.62E-05
	0.75	103500	207000	3255.41	0.26	0.29	8120	7.55E-05	9.25E-06	6.62E-05
	1	103500	207000	3573.28	0.26	0.29	8371	7.54E-05	9.24E-06	6.62E-05

References:

[1] L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in direct networks. IEEE Tran. on Computers, 26, Feb. 1993, pp.62–76.

[2] Jingcao Hu; Marculescu, R. "DyAD – Smart Routing for Network-on-Chip", *Proceedings of the 41st annual Design Automation Conference*, June 07-11, 2004, pp.260 -263.

[3] *Noxim* simulator: https://github.com/davidepatti/noxim

[4] P. V. Gratz., S. W. Keckler. Realistic Workload Characterization and Analysis for Network-on-Chip Design, *4th Workshop on Chip Multiprocessor Memory Systems and Interconnects (CMP-MSI)*, January 2010.

[5] V. Catania, A. Mineo, S. Monteleone, M. Palesi, D. Patti. Noxim: An Open, Extensible and Cycle-accurate Network on Chip Simulator. *IEEE Int.Conference on Application-specific Systems, Architectures and Processors* 2015. July 27-29, 2015, Toronto, Canada.

[6] G.-M. Chiu. The odd-even turn model for adaptive routing. *IEEE Trans. on Parallel and Distributed Systems*, July 2000. pp.729–738.

[7] W. J. Dally, C. L. Seitz. The torus routing chip. Distributed Comp., 1986. pp.187–196

[8] W. J. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. *Proceedings of DAC*. June 2001, pp.684–689.

[9] J. Hu and R. Marculescu. Smart routing for networks-on-chip. Technical report, ECE Department, Carnegie Mellon University, March 2004.