

DyAD – Smart Routing for Networks-on-Chip*

Jingcao Hu
Carnegie Mellon University
Pittsburgh, PA 15213-3890, USA
jingcao@ece.cmu.edu

Radu Marculescu
Carnegie Mellon University
Pittsburgh, PA 15213-3890, USA
radum@ece.cmu.edu

ABSTRACT

In this paper, we present and evaluate a novel routing scheme called *DyAD* which combines the advantages of both deterministic and adaptive routing schemes. More precisely, we envision a new routing technique which judiciously switches between deterministic and adaptive routing based on the network's congestion conditions. The simulation results show the effectiveness of *DyAD* by comparing it with purely deterministic and adaptive routing schemes under different traffic patterns. Moreover, a prototype router based on the *DyAD* idea has been designed and evaluated. Compared to purely adaptive routers, the overhead of implementing *DyAD* is negligible (less than 7%), while the performance is consistently better.

Categories and Subject Descriptors

B.4 [Hardware]: Input/output and data communications

General Terms

Algorithms, Performance, Design

Keywords

Networks-on-Chip, Systems-on-Chip, router design

1. INTRODUCTION

The regular tile-based NoC architecture was recently proposed as a solution to the complex on-chip communication problems [3]. Such a chip consists of a grid of regular tiles where each tile can be a general-purpose processor, a DSP, a memory subsystem, *etc.* A router is embedded within each tile with the objective of connecting it to its neighboring tiles. Thus, instead of routing design-specific global on-chip wires, the inter-tile communication can be achieved by routing packets. The performance and the efficiency of the NoC depends on the underlying communication infrastructure; this, in turn, depends on the performance (latency and throughput) of the on-chip routers. Thus, the design of efficient, high performance routers represents a critical issue for the success of the NoC approach.

Routers can be generally classified into *deterministic* and *adaptive* [8]. In deterministic routing (also called oblivious routing), the path is completely determined by the source and the destination address. On the other hand, a routing technique is called adaptive if, given a source and a destination address, the path taken by

*Research supported by NSF CCR-00-93104 and Marco Gigascale Research Center (GSRC)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DAC 2004, June 7–11, 2004, San Diego, California, USA.
Copyright 2004 ACM 1-58113-828-8/04/0006 ...\$5.00.

a particular packet depends on dynamic network conditions (e.g. congested links due to 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 can not dynamically respond to network congestion. In contrast, adaptive routers avoid congested links by using alternative routing paths; this leads to higher throughput. However, due to the extra logic needed to decide on a good routing path, adaptive routing has a higher latency at low levels of network congestion.

In this paper, we present a novel routing scheme which combines the advantages of both deterministic and adaptive routing schemes. The proposed routing scheme, dubbed *DyAD* from **D**ynamic **A**daptive **D**eterministic switching, is based on the current network congestion. More precisely, with *DyAD* routing each router in the network continuously monitors its *local* network load and makes decisions based on this information. When the network is not congested, a *DyAD* router works in a 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 exploiting other routing paths; this leads to higher network throughput which is highly desirable for applications implemented using the NoC approach. To propose a valid approach, we also show how the freedom from deadlock and livelock [8] can be guaranteed when mixing deterministic and adaptive routing modes into the same NoC.

Experimental results show that, compared to both deterministic and adaptive routing, significant performance improvements can be achieved by using the *DyAD* approach. At the same time, by prototyping a *DyAD* router based on a popular adaptive routing strategy, we show that the chip area overhead is marginal (typically less than 7%), while its performance consistently outperforms that of a purely adaptive router. We believe that the proposed scheme based on combining the deterministic and adaptive routing modes has great potential for future NoC implementations.

The paper is organized as follows. In Section. 2 we review the related work. In Section 3, we present the *DyAD* router architecture and a practical implementation. Experimental results in Section 4 validate the performance improvements and the prototype design (Section 5) shows that the implementation overhead compared to a traditional adaptive router is indeed negligible.

2. RELATED WORK

There has been significant work on efficient routing schemes in parallel and distributed computing areas [4][1]. Because of limited space, the reader is referred to [8] for a survey on routing techniques developed for direct networks.

In [11], Shin *et al.* propose a hybrid switching scheme that dynamically combines both virtual cut-through [6] and wormhole

switching [2] to provide higher achievable throughput values compared to wormhole switching alone, while reducing the buffer space required at the intermediate nodes when compared to virtual cut-through. In this paper, we are looking at the router design from another perspective; that is, we try to combine the advantages provided by deterministic and adaptive routing instead of relying on different switching schemes. Thus, the work presented here is orthogonal to that in [11]. Interestingly enough, they can be eventually combined, if needed.

From another perspective, several on-chip routers have been proposed for NoC (e.g. [10][7][9]). However, none of this previous work has addressed the issue of combining deterministic and adaptive routing into a new routing scheme. As we will see later in this paper, by doing so one can achieve significant better performance compared to purely adaptive routers only with negligible implementation overhead.

3. THE DyAD ROUTER ARCHITECTURE

Rather than a detailed implementation, *DyAD* is about a new *paradigm* for NoC router design which exploits the advantages of deterministic and adaptive routing. Indeed, based on this idea, *any* suitable deterministic and adaptive routing scheme can be combined to form a *DyAD* router (although care must be taken to issues such as deadlock freedom, as it will be discussed in Section 3.3). Similar ideas can be extended to other routers (e.g. routers for multi-computer networks) as well.

3.1 Platform description

Due to its popularity, the platform under consideration is composed of a $n \times n$ array of tiles which are inter-connected by a 2D mesh network (see [7][9]). In such a regular architecture, each tile is composed of a *processing element* (PE) and a *router*. The router embedded onto each tile is connected to the four neighboring tiles and its local PE via channels. Each channel consists of two *directional point-to-point* links between two routers or a router and a local PE [5].

Because of the limited silicon resources and the low-latency requirements for typical NoC applications, wormhole switching is used as the switching scheme for the on-chip routers. Under this scheme, a packet is split into so-called *flits* (flow control digits), which are then routed in a pipelined fashion. To minimize the implementation costs, the on-chip network should be implemented with very little silicon area overhead. Thus, instead of having huge memories (e.g. SRAM or DRAM) as buffering space, it's more reasonable to use registers. A 5×5 crossbar switch is used as the switching fabric because of its nice cost/performance trade-offs for switches with small number of ports.

In our experiments, the *XY* routing is used as a representative deterministic routing scheme because of its simplicity and wide popularity. Obviously, *XY* routing is a *minimal* path routing algorithm and is *free* of deadlock and livelock [8]. Unlike the deterministic routing where the routing path is fixed once the source and the destination addresses are given, the adaptive routing offers packets more flexibility 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 packets waiting for each other in a cycle.

Starting from these observations, we use 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 the adaptiveness, it should *not* prohibit more turns than necessary. Chiu in [1] proposed the *odd-even* turn model which restricts the locations

where some types of turns can take place such that the algorithm remains deadlock-free. More precisely, the *odd-even* routing prohibits the *east*→*north* and *east*→*south* turns at any tiles located in an even column. It also prohibits the *north*→*west* and *south*→*west* turns at any tiles located in an odd column. Compared to other adaptive routing algorithms without virtual channel support (e.g. [4]), the degree of the adaptiveness provided by the *odd-even* routing is distributed more evenly across the network. Thus, in this paper, we choose the *minimal*¹ *odd-even* routing as the adaptive routing scheme for on-chip routers. The use of *minimal* routing helps not only in reducing the energy consumption of communication, but also in keeping the network free from the livelock.

3.2 Motivation for the DyAD approach

As a motivational example, Fig. 1 shows how the performance of the deterministic and adaptive routing changes with respect to the network load for a 6×6 mesh under *transpose1* traffic pattern (please refer to Section 4 for detailed description of the simulation setup and the traffic pattern's characteristics).

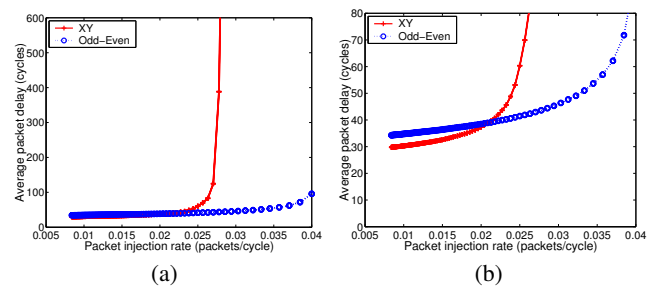


Figure 1: Performance comparison between XY and odd-even

In Fig. 1(a), we report the measured average communication latency of the packets and the average sustainable network throughput, in terms of packet injection rates at each node. Fig. 1(b) shows the magnified view of the average packet latency at low injection rates. These figures clearly show the trade off between deterministic (*XY*) and adaptive (*odd-even*) routing. More precisely, the *odd-even* is able to achieve much higher saturation throughput compared to *XY* routing (more than 60% in this experiment). However, at low network workloads (below 0.023 packets/cycle in this case), *XY* beats the *odd-even* routing in terms of average packet latency². This is exactly the trade off that motivated us to develop the *DyAD* routing, which tries to combine the advantages of both deterministic and adaptive routing by judiciously choosing the appropriate routing mode under different traffic conditions.

3.3 DyAD-OE: A DyAD implementation of adaptive odd-even routing

In this subsection, we present the actual router design, *DyAD-OE*, which implements the concept of *DyAD* for *odd-even* routing. Combining *odd-even* and *XY* to form a *DyAD* router may lead to deadlock. Thus, we develop a new routing scheme, called *oe-fixed*, as the deterministic routing mode in *DyAD-OE*. *Oe-fixed* is indeed a deterministic version of *odd-even* based on removing the *odd-even*'s adaptiveness. For instance, in *odd-even* mode, if a packet with a given source and destination can be routed to both output p_1 and p_2 , it will always be routed to p_1 in *oe-fixed*. Fig. 2 illustrates the architecture of the *DyAD-OE* implementation.

¹A *minimal* adaptive routing algorithm routes *all* packets through the *shortest* paths to the destination.

²To fairly compare different routing schemes, more traffic load/patterns and network configurations need to be tested, as shown in Section 4.

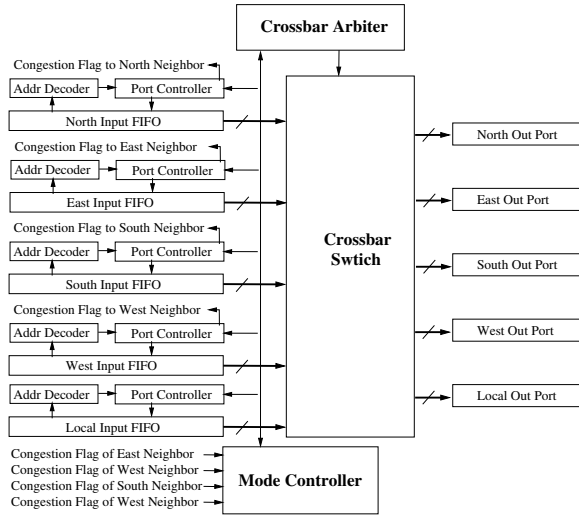


Figure 2: The *DyAD-OE* router architecture

Each input controller in Fig. 2 has a separate FIFO (typically several flits implemented by registers for performance and power efficiency) which buffers the input packets before delivering them to the output ports. When a new header flit is received, the address decoder processes that flit and sends the destination address to the port controller; this determines which output port the packet should be delivered to. In the *odd-even* mode, there can be more than one output direction to route packets. In this case, the port controller will choose the direction in which the corresponding downstream router has more empty slots in its input FIFO. Once the router has made its decision on which direction to route, the port controller sends the connection request to the crossbar arbiter in order to set up a path to the corresponding output port.

Except for the local input controller, each input port controller also monitors its FIFO occupation ratio. If the ratio reaches the preset *congestion threshold*, a value 1 will be asserted (indicating to the upstream router that the downstream router is congested) on the corresponding congestion flag wire. Otherwise, a value of 0 will be asserted, indicating to the upstream router that congestion is not an issue.

The *Crossbar Arbiter* maintains the status of the current crossbar connection and determines whether to grant connection permission to the port controller. When there are multiple input port controllers requests for the same available output port, the *Crossbar Arbiter* uses the first-come-first-served policy to decide which input port to grant the access, such that the starvation at a particular port can be avoided.

The *Mode Controller* continuously monitors its neighboring congestion to determine if the deterministic or the adaptive routing mode need to be used. Although more advanced techniques can be used to determine the optimal routing mode, we use the following simple policy: if any congestion flag from its neighboring routers are asserted, then the *Mode Controller* commands all the input port controllers to work in the adaptive (*odd-even*) mode; otherwise, it switches the port controllers to the deterministic (*oe-fixed*) mode.

4. EXPERIMENTAL RESULTS

To evaluate the performance gains that can be achieved with *DyAD*, we simulate four types of mesh networks, which use *XY*, *odd-even*, *oe-fixed* and *DyAD-OE*, respectively. The efficiency of each type of routing is evaluated through latency-throughput curves. Similar to previous work, we assume that the packet latency spans

the instant from when the first flit of the packet is created, to the time when the last flit is ejected to the destination node, including the queuing time at the source. We assume that the packets are consumed immediately once they reach their destination nodes. Each simulation is run for a warm-up period of 2000 cycles. Thereafter, performance data is collected after 20,000 packets are sent. A cycle-accurate interconnection network simulator (*worm_sim*) was implemented in C++. *Worm_sim* supports 2D mesh networks with wormhole switching; it has been designed to be easy customized to simulate different designs under different traffic patterns. Since many factors (e.g. routing path selection delay, crossbar arbitration delay, etc.) have a significant impact on the NoC performance, *worm_sim* models them accurately with their actual values taken from the prototype router designs.

4.1 Evaluation under random traffic

In this set of experiments, we use random traffic to simulate the performance of the network under different routing strategies. The processing elements (PEs) generate 5-flit packets at time intervals chosen with exponential distributions. The network size during simulation is fixed to be 6×6 tiles. The input ports have a FIFO size of 5 flits, with the congestion threshold set at 60% of the total FIFO capacity.

Fig. 3(a-b) show the latency/throughput figures under *uniform* and *transpose1* traffic patterns, respectively. Under the *uniform* traffic pattern, a PE sends a message to any other node with equal probability. Let E to be the edge size of the square mesh under simulation. Under the pattern *transpose1*, a PE at (i, j) ($i, j \in [0, E)$) only sends messages to node $(E - i, E - j)$.

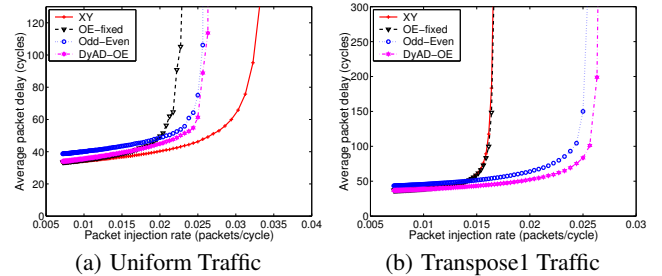


Figure 3: Performance evaluation under random traffic

As shown in Fig. 3(a), *XY* routing performs better than both *odd-even* and *DyAD-OE* routing under *uniform* traffic load. This result is consistent with other results reported in the literature (e.g. [1][4]). The reason why *XY* performs best under *uniform* traffic is because it embodies global, long-term information about this traffic pattern. On the other hand, the adaptive algorithms select the routing paths based on *local*, short-term information. This type of decision benefits only the packets in the immediate future, which tend to interfere with other packets. Thus, the evenness of uniform traffic is not necessarily maintained in the long run [4].

However, for most of the applications in real world, each node will communicate with some nodes more frequently compared to others. *XY* routing has serious problems in dealing with such non-uniform traffic patterns because of its determinism. More precisely, *XY* routing blindly maintains the unevenness of the nonuniform traffic, just as it maintains the evenness for the uniform traffic. In this spirit, Fig. 3(b) shows that *XY* routing is clearly outperformed by *odd-even* and *DyAD-OE* under *transpose1* traffic. In this case, the network using *XY* saturates at an injection rate of 0.0167 packets/cycle, while *odd-even* and *DyAD-OE* are able to achieve a throughput of 0.0256 packets/cycle and 0.027 packets/cycle, respectively. This gives a 53.3% and 61.7% improvement in terms

of sustainable throughput. In fact, for the same traffic pattern and injection rate, *DyAD-OE* achieves *shorter* average packet latency compared to *odd-even* throughout the experiments.

Another interesting fact is that *DyAD-OE* does keep the advantage of deterministic routing when the network is not congested. As shown in Fig. 3, *DyAD-OE* has the same average packet latency when network is not congested. On the other hand, the average latency a packet experiences in *odd-even* is 14% higher compared to that in *DyAD-OE*, when the network is lightly loaded.

Other non-uniform traffic patterns (like *transpose2* and *hot spot*) have been simulated as well and the results were similar to that under *transpose1* traffic pattern. We also simulated different network sizes (ranging from 4×4 to 8×8 tiles) and different FIFO sizes (ranging from 3 to 8 flits). All the results reflect the same characteristic as in Fig. 3. For complete results, please refer to [5].

4.2 Evaluation under multimedia traffic

Real world traffic (both in macro and on-chip networks) frequently exhibits patterns with self-similarity and long-range dependencies [12][13]. This can be quite a different scenario compared to the traffic patterns used in subsection 4.1. In what follows, we present some experimental results when the network is simulated under realistic traces which exhibit self-similarity.

We first profiled an H263 video decoder application using different video clips to retrieve 9 real traces and recorded the arrival data for the motion compensation module. A 4×4 network is then constructed in which nine PEs are randomly picked to generate the packets according to the corresponding input trace files. The remaining PEs in the network generate uniform traffic. We incrementally increase the packet injection rates to mimic the case when the system increases its speed of playing the video clips. The results are shown in Fig. 4.

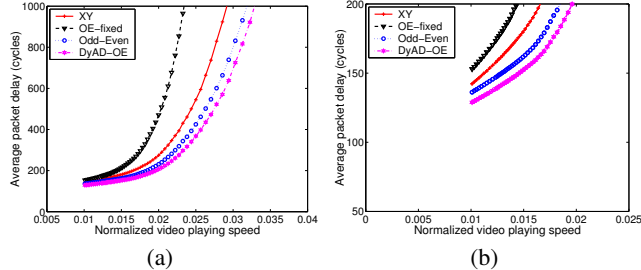


Figure 4: Performance evaluation under multimedia traffic

Plotted in Fig. 4(a) is the measured average communication latency of the packets and video playing speed. As we can see, the results show the same trend as those derived under the non-uniform traffic patterns in subsection 4.1. *DyAD-OE* always performs the best under all playing rates, while *odd-even* performs second best.

Fig. 4(b) shows a magnified view of Fig. 4(a) for the low speed region (that is, the region corresponding to 0.01 to 0.025 playing speed). It is interesting to note that unlike the simulation results using random traffic patterns where *DyAD-OE* has the same latency as *XY* and *oe-fixed*, *DyAD-OE* enjoys now a shorter latency compared to that of *XY* and *oe-fixed*, even at very low playing speeds. This behavior is due to the bursty nature of the multimedia traces.

5. PROTOTYPE ROUTER DESIGNS

To evaluate the overhead of *DyAD-OE*, we have implemented several designs and checked the actual performance/area trade offs. More precisely, we want to see whether or not implementing a *DyAD-OE* router requires an almost negligible additional cost compared to an *odd-even* router. We implemented all four versions

of routers (*XY*, *oe-fixed*, *odd-even* and *DyAD-OE*) using a $0.16\mu\text{m}$ technology, with a clock rate of 333MHz. In our designs, the FIFOs are implemented using registers in order to achieve better performance/power efficiency. Each input port has a fixed link width of 32 bits. The flit size is set to be 32 bits as well. For each version of routers, several design instances were synthesized with different input FIFO capacities, starting from 2 flits per input FIFO to 8 flits per input FIFO.

As shown by our design results, the overhead of implementing the extra logic for *DyAD-OE* is indeed negligible compared with *odd-even* implementation. For instance, for odd column routers with FIFO size of 8 flits, *DyAD-OE* requires 25,971 gates, while *odd-even* router requires 25,891 gates (less than 1% overhead). In fact, for all those designs (starting from FIFO size of 2 flits to FIFO size of 8 flits), the overhead compared to *odd-even* is below 7%, with an average overhead as small as 0.54%. (Please refer to [5] for detailed comparison and layout plots.)

6. CONCLUSION AND FUTURE WORK

We presented a novel NoC routing idea (called *DyAD*) which combines the low latency of the deterministic routing (at low network load) and the high throughput of the adaptive routing. Based on this new concept, an instance of the *DyAD-OE* was designed based on *minimum odd-even* routing. The simulation results show that *DyAD-OE* consistently outperforms *odd-even* under different traffic loads/patterns and different network configurations. At the same time, *DyAD-OE* enjoys the same low packet latency as deterministic routing when the network is not heavily loaded.

As explained in the paper, *DyAD* routing is a new concept rather than a particular design or implementation choice. To achieve the best performance, the configuration of *DyAD* (e.g. which adaptive/deterministic routing should be used, the mode switching policy, etc.) should be customized to match the given application traffic characteristics. This remains to be done as future work.

References

- [1] G.-M. Chiu. The odd-even turn model for adaptive routing. *IEEE Tran. on Parallel and Distributed Systems*, 11(7):729–738, July 2000.
- [2] W. J. Dally and C. L. Seitz. The torus routing chip. *Distributed Computing*, 1(3):187–196, 1986.
- [3] W. J. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. In *Proc. DAC*, pages 684–689, June 2001.
- [4] C. J. Glass and L. M. Ni. The turn model for adaptive routing. *Journal of ACM*, 41(5):874–902, Sept. 1994.
- [5] J. Hu and R. Marculescu. Smart routing for networks-on-chip. Technical report, ECE Department, Carnegie Mellon University, March 2004. Available at <http://www.ece.cmu.edu/~sld/pubs>.
- [6] P. Kermani and L. Kleinrock. Virtual cut-through: a new computer communication switching technique. In *Computer Networks*, volume 3, pages 267–286, Sept. 1979.
- [7] J. Liang, S. Swaminathan, and R. Tessier. aSOC: a scalable, single-chip communication architectures. In *IEEE Int. Conf. on Parallel Architectures and Compilation Techniques*, pages 37–46, Oct. 2000.
- [8] L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in direct networks. *IEEE Tran. on Computers*, 26:62–76, Feb. 1993.
- [9] E. Nilsson, M. Millberg, J. Oberg, and A. Jantsch. Load distribution with the proximity congestion awareness in a network on chip. In *Proc. DATE*, pages 1126–1127, March 2003.
- [10] E. Rijpkema, K. G. Gossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In *Proc. DATE*, March 2003.
- [11] K. G. Shin and S. W. Daniel. Analysis and implementation of hybrid switching. *IEEE Tran. on Computers*, 45(6):684–692, 1996.
- [12] G. Varatkar and R. Marculescu. On-chip traffic modeling and synthesis for MPEG-2 video application. *IEEE Tran. on VLSI*, 12(1), Jan. 2004.
- [13] W. Willinger, V. Paxson, and M. S. Taqqu. *A Practical Guide to Heavy Tails: Statistical Techniques and Applications*, chapter Self-similarity and Heavy Tails: Structural Modeling of Network Traffic. Birkhauser Verlag, 1998.