DyAD – Smart Routing for Networks-on-Chip[*]

Jingcao Hu Radu Marculescu

Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213-3890, USA
e-mail: {jingcao, radum}@ece.cmu.edu

April 25, 2004

The performance of Networks-on-Chip (NoC) is highly dependent on the throughput and latency properties of the on-chip routers. Deterministic routing gives better latency at low packet injection, however, the performance degrades quickly when the network becomes congested. Compared to deterministic routing, adaptive routing alleviates the congestion problem at higher packet injection rates by distributing packets across multiple routing paths. However, it suffers from higher latency at low injection rates due to the extra logic needed to perform the routing path selection.

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.

1 Introduction

With the advances in the semiconductor technology, the huge number of transistors available on a single chip allows designers to integrate tens of IP blocks together with large amounts of embedded memory. These IPs can be CPU or DSP cores, video stream processors, high-bandwidth I/O, *etc.* This richness of the computational resources places

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

tremendous demands on the communication resources as well. Additionally, the shrinking feature size in the *deep-submicron* (DSM) domain makes interconnect delay and power consumption the dominant factors in the optimization of modern systems. Another consequence of the DSM effects is the difficulty in optimizing the interconnect because of the worsening effects such as crosstalk, electro magnetic interference, *etc* [13].

The regular tile-based NoC architecture was recently proposed as a solution to the complex on-chip communication problems [3][6][8]. As shown in Fig. 1, 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.

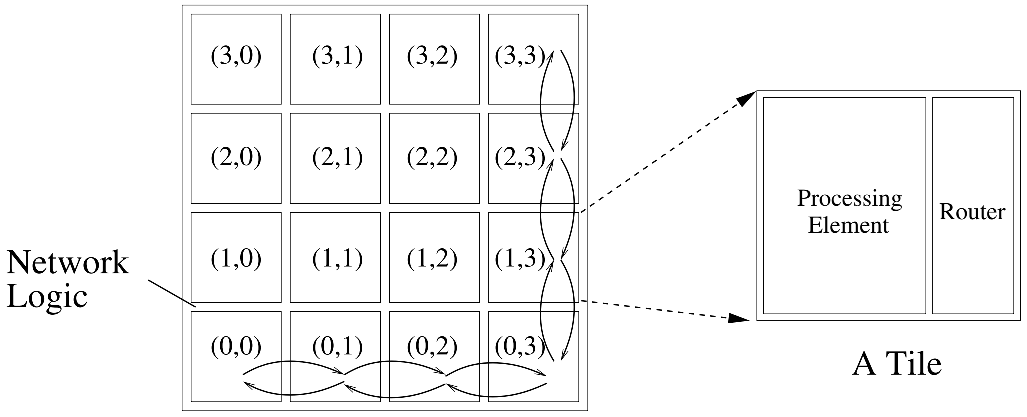


Figure 1: Regular tile-base NoC architecture

The performance and the efficiency of the NoC highly depends on the underlying communication infrastructure; this, in turn, relies on the performance (in terms of latency and throughput) of the on-chip routers. Thus, the design of efficient, high performance, on-chip routers represents a critical issue for the success of the NoC approach.

Depending on the routing mode, routers can be generally classified into two types: *deterministic* and *adaptive* [10]. 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 a particular packet depends on dynamic network conditions (e.g. congested links due to traffic variability).

One main advantage of using deterministic routing is its simplicity in terms of routers design. Because of the simplified logic, the deterministic routing provides low routing latency when the network is not congested. However, as the packet injection rate increases, the deterministic routers are likely to suffer from throughput degradation as they can not dynamically respond to network congestion. In contrast, adaptive routers increase the chances of packets to avoid congested links by using alternative routing paths; this leads to higher throughput. However, because of the extra logic needed in order to decide on a good routing path, adaptive routing has a higher routing latency compared to the deterministic routing, 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, which is dubbed as *DyAD*, from **Dy**namically switching between **A**daptive and **D**eterministic modes, is based on the current network congestion situation. More precisely, in *DyAD* 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 and thus enjoys 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 a NoC.

In order to propose a valid approach, we also show how the freedom from deadlock and livelock [10] 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. By designing a prototype *DyAD* router based on a typical adaptive router, we show that the overhead in terms of chip area is marginal (typically less than 7%), while its performance consistently outperforms the 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 first give a brief review of the related work. Following that, in Section 3, we present the *DyAD* router architecture and an implementation which fulfills the *DyAD* concept. Experimental results in Section 4 validate the performance improvement which can be achieved with *DyAD* routers. More than this, the prototype design (Section 5) shows that the implementation overhead compared to a traditional adaptive router is negligible.

2 Related Work

There has been significant work published on efficient routing schemes in parallel and distributed computing areas. Glass and Ni in [5] propose a turn-model for designing wormhole routing algorithms that are deadlock and livelock free. This model has been later utilized by Chiu in [1] to develop an odd-even adaptive routing algorithm for meshes without virtual channels. However, to the best of our knowledge, there is no work which tries to exploit the combined advantages of deterministic and adaptive routing when implementing NoCs. Because of limited space, the reader is referred to [10] for a survey on routing techniques developed for direct networks.

In [14], Shin *et al.* propose a hybrid switching scheme that dynamically combines both virtual cut-through [7] 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 [14]. Interestingly enough, they can be combined, if needed.

From another perspective, several on-chip routers have been proposed for NoC (e.g. [12][9][11]). 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, our proposed routing scheme achieves significant better performance compared to purely adaptive routers only with negligible implementation overhead.

3 The DyAD Router Architecture

Instead of proposing a detailed implementation, *DyAD* proposes 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). Nevertheless, in what follows, we focus on discussing an implementation instance which targets the NoC domain. Similar ideas can be extended to other routers (e.g. routers for multi-computer networks) as well.

3.1 Platform Description

The platform under consideration is composed of a $n \times n$ array of tiles which are interconnected by a $2D$ mesh network. We choose the $2D$ mesh network mainly because of two reasons: it naturally fits the tile-based architecture and because it has been frequently discussed in other NoC work (e.g. [3][9][11]). However, we emphasize that our algorithm can be easily extended for other topologies.

Fig. 2 shows an abstract view of a tile in this architecture. As shown, 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.

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. This is especially important for those architectures composed of tiles at fine-level of granularity. Thus, instead of having huge memories (e.g. SRAM or DRAM) as buffering space for routers/switches in the macro-network, it's more reasonable to use registers. For the architecture in Fig. 2,

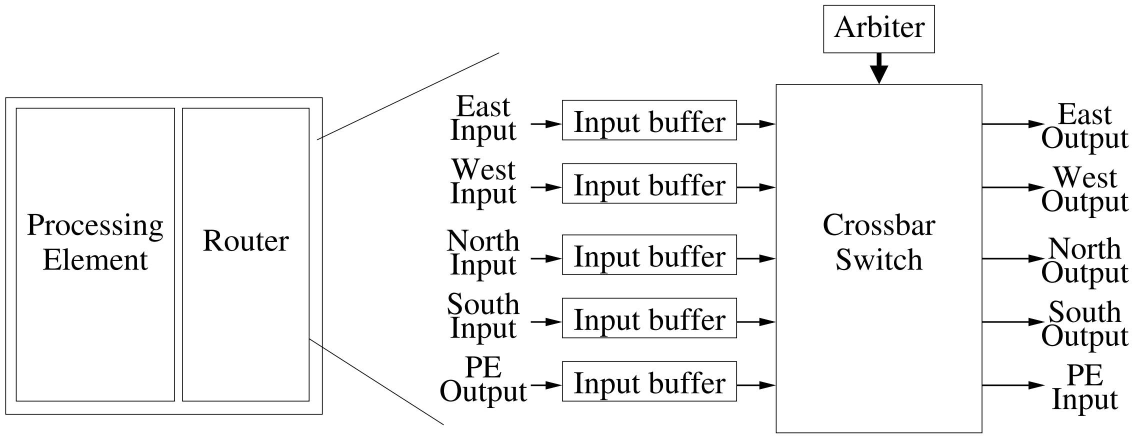


Figure 2: The typical structure of a tile

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.

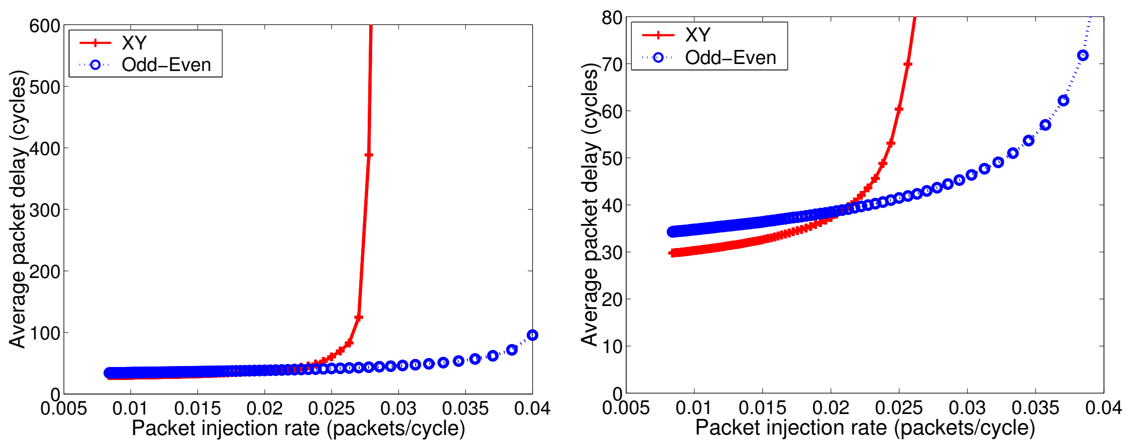
In our experiments, the XY routing scheme is picked up as a representative deterministic routing scheme because of its simplicity and wide popularity. In short, for $2D$ mesh networks, the XY routing first routes packets along the X-axis. Once the packets reach the column wherein lies the destination tile, they are then routed along the Y-axis. Obviously, XY routing is a *minimal* path routing algorithm and is *free* of deadlock and livelock [10].

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. One way to achieve deadlock-free adaptive routing is by using virtual channels (e.g. [4]). However, adding virtual channels does not come for free, as it requires extra buffering spaces and complex control logic to routers. These pose serious problems on applying the virtual channel approach to NoCs.

Starting from these observations, we instead 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 o the possible routing cycles. In addition, in order to preserve the adaptiveness, it should *not* prohibit more turns than necessary. Several deadlock-free adaptive routing algorithms have been proposed [5], including *west-first*, *north-last* and *negative-first*. In [1], Chiu 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. [5]), 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 to keeping the network free from the livelock.

3.2 Motivation for the DyAD Approach

Intuitively, because of its simplicity, the deterministic routing provides lower routing latency compared to adaptive routing when the network is not really congested. On the other hand, the deterministic routing is more susceptible to congested links as it provides no choice for alternative routes when network becomes congested. As a motivational example, Fig. 3 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).



(a) (b)

Figure 3: Performance Comparison between XY and Odd-even

In Fig. 3 , 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. Compared to Fig. 3(a), Fig. 3(b) uses a smaller Y-axis scale to show 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 a low network work load (below 0.023 packets/cycle in this case), XY beats the *odd-even* routing in terms of average packet latency[2]. This is exactly the trade off that motivated us to develop the *DyAD* routing, which tries to combine

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

[2] Of course, in order to fairly compare different routing schemes, more traffic load/patterns and network configurations need to be tested, which will be shown in Section 4.

the advantages of both deterministic and adaptive routing by judiciously choosing the right routing mode under different traffic load.

3.3 DyAD-OE: a DyAD Implementation of Adaptive Odd-even Routing

We would like to emphasize again that *DyAD* stands for a routing concept rather than a particular combination of a deterministic and an adaptive routing mode. In principle, any combination of deterministic and adaptive routing scheme can be used to construct a *DyAD* router (providing that the resulted network meets other constraints, such as freedom from deadlock, *etc.*). The choice of the "best" combination depends on factors such as resources available and the application requirements/characteristics, *etc.*

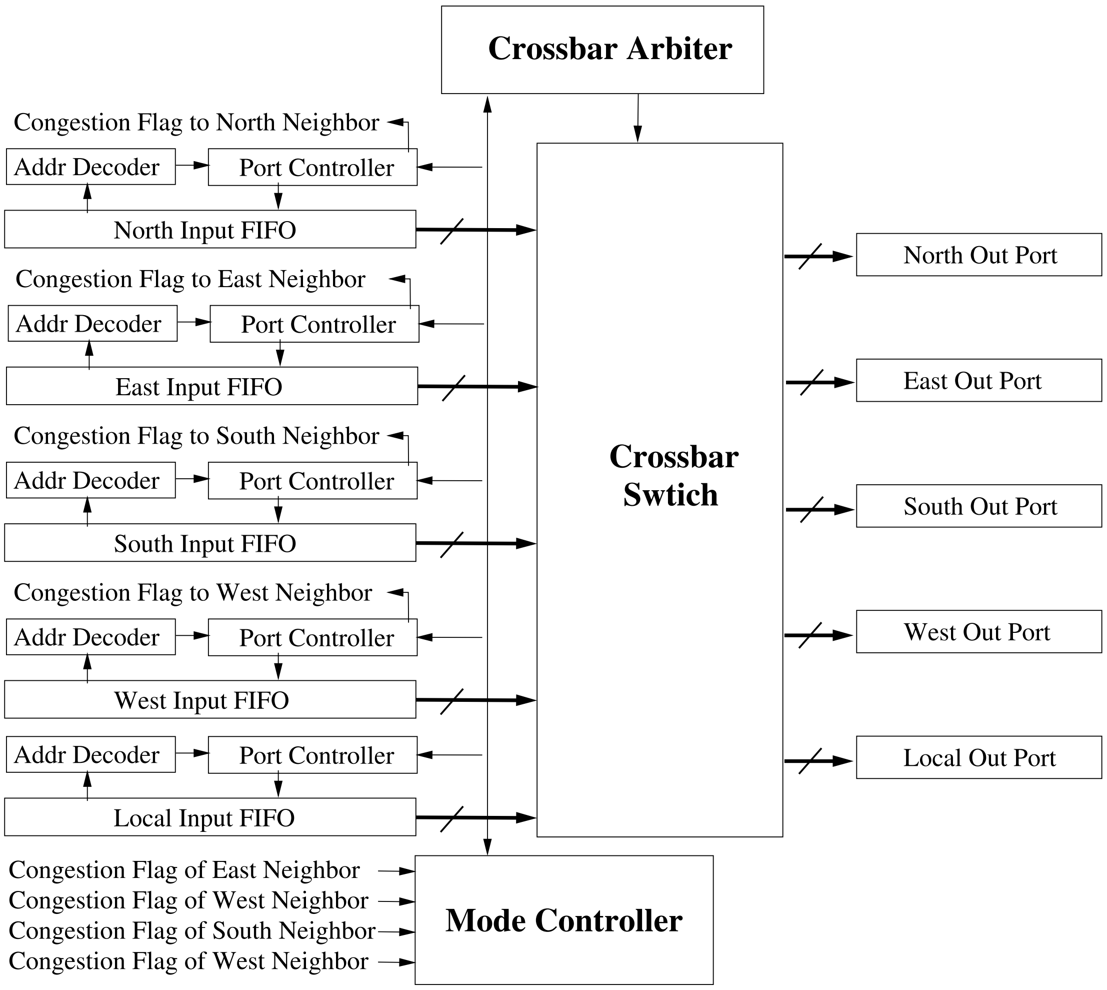


Figure 4: The *DyAD-OE* Router Architecture

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 problem. 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. 4 illustrates the architecture of the *DyAD-OE* implementation.

Each input controller in Fig. 4 has a separate FIFO (typically the size of several flits and 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. When the router works 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. For instance, let us suppose the router located in the position of $(1, 1)$ as in Fig. 1 receives a packet from router $(1, 0)$ with the destination address $(3, 3)$. Under *odd-even* node, the packet can be routed either to north or to east in this router. In this case, the port controller compares the occupancy of the south input FIFO at router $(2, 1)$ with that of the west input FIFO at router $(1, 2)$. If the former has more flits in the FIFO, then the packet will be routed to the east to router $(1, 2)$; otherwise, it will be routed to the north to router $(2, 1)$. 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 specified *congestion threshold*, a value 1 will be asserted on the corresponding congestion flag wire. Otherwise, a value of 0 will be asserted. Intuitively, a value of 1 in the congestion flag indicates to the upstream router that the downstream router is congested, and it is better to use adaptive routing in this case in order to avoid possible congested links. On the other hand, a value of 0 tells the upstream router that congestion is not an issue and it should dump the packets out as fast as possible for minimum latency.

The use of the FIFO is regulated by back-pressure mechanism. Under this scheme, a flit will be held in the buffer until the downstream router has empty space in the corresponding input FIFO. Thus, the network will not drop any packet in transit. This is extremely important for NoC architectures which may not implement very advanced end-to-end protocol.

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 (FCFS) policy to decide which input port to grant the access, such that the starvation on a particular port can be avoided.

The mode controller continuously monitors its neighboring congestion to determine whether the deterministic or the adaptive routing mode should be used. Although more advanced techniques can be used to determine the optimal routing mode, we use the following simple policy to demonstrate the idea of *DyAD*. If any congestion flag from its neighboring routers are asserted, then the mode controller commands all the input port controllers to work at the adaptive (*odd-even*) mode; otherwise, it switches the port controllers to deterministic (*oe-fixed*) mode.

4 Experimental Results

To evaluate the performance gains that can be achieved with *DyAD*, we simulate several square mesh networks with different routing schemes and design parameters under different traffic patterns. Under each load and configuration, four types of mesh networks are simulated, 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 other work in the literature, we assume that the packet latency spans the instant when the first flit of the packet is created, to the time when last flit is ejected to the destination node, including the queuing time at the source. We also 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 are collected after 20,000 packets are sent.

4.1 Worm_sim Simulator

A cycle-accurate interconnection network simulator (named as *worm_sim*) was implemented in *C++* based on standard template libraries (STL). *Worm_sim* supports *2D* mesh networks with wormhole switching and is designed with flexibility in mind so that it can be 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 all these factors accurately with their actual values taken from our prototype router designs[3]. More specifically, for different types of routers (e.g. *XY*, *odd-even*, *etc.*), their performance related delays are modeled differently as parameters in *worm_sim* by accurately mimicking our prototype circuit design.

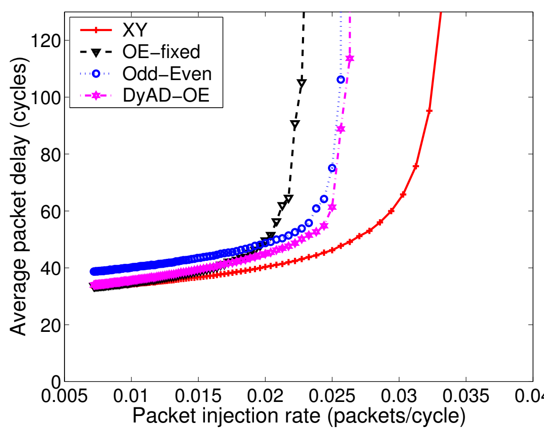
4.2 Performance Evaluation under Random Traffic

In this set of experiments, we used the random traffic model to simulate the performance of the networks which use different routing strategies. In this case, the processing elements (PEs) generate messages at time intervals chosen with exponential distribution, as commonly reported in the literature. Each message is assumed to be a 5-flit packet. Similar to the work presented in [1], three traffic patterns, namely *uniform*, *transpose* and *hot spot* are used in the simulation. Under the *uniform* traffic pattern, a PE sends a message to any other node with equal probability. Two types of the transpose patterns are used

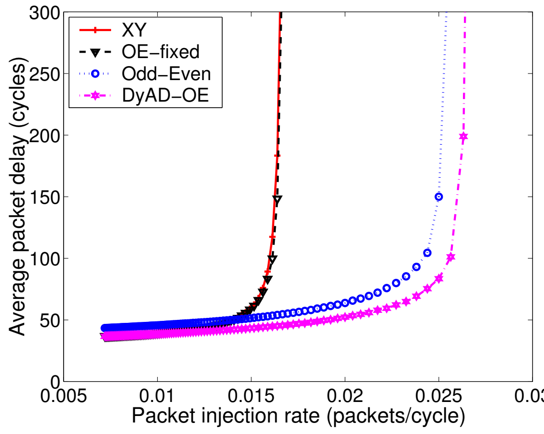
[3]Details of these designs of the routers will be presented in Section 5.

for the transpose traffic. Let E to be the edge size of the square mesh under simulation. Under the pattern *transpose1* (also called *matrix-transpose* in some publications), a PE at (i, j) $(i, j \in [0, E))$ only sends messages to node $(E - 1 - i, E - 1 - j)$; under the pattern *transpose2*, a node (i, j) only sends messages to node (j, i). *Transpose1* and *transpose2* correspond to reflections of the source about the lines $y = -x$ and $y = x$, respectively, given a coordinate system through the center of the network. Finally, under the *hot spot* traffic pattern, one or more nodes are chosen as *hot spots* which receive an extra proportion of traffic in addition to the regular uniform traffic.

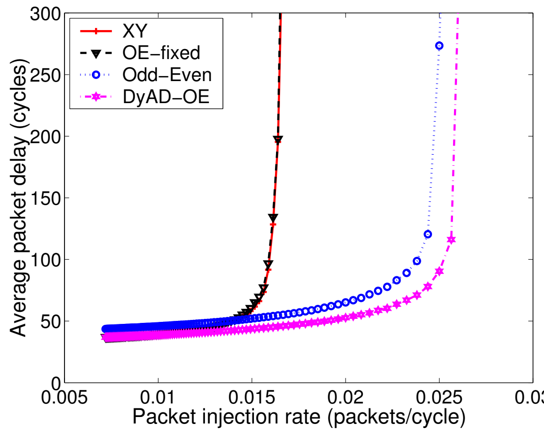
Fig. 5(a-d) show the latency/throughput figures under *uniform*, *transpose1*, *transpose2* and *hot spot* traffic patterns, respectively. The network size during simulation is fixed to be 6×6 tiles. All of the input ports have a FIFO size of 5 flits, with the congestion threshold set at 60% of the total FIFO capacity.



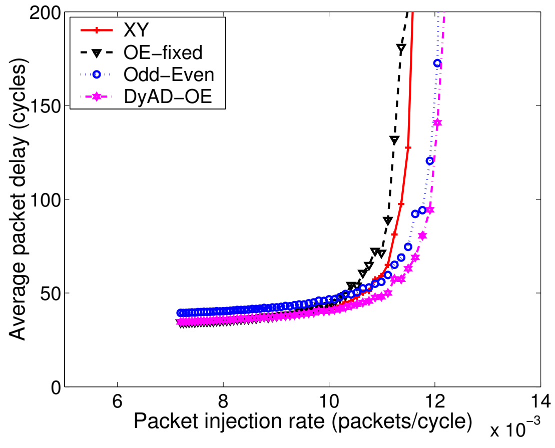
(a) Uniform Traffic



(b) Transpose1 Traffic



(c) Transpose2 Traffic



(d) Hot Spot Traffic

Figure 5: Performance Evaluation under Random Traffic Patterns

As shown in Fig. 5(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. [5][1]). The reason why XY performs best under *uniform* traffic is that it embodies global, long-term information about this traffic pattern. From a global, long-term point of view, the *uniform* traffic pattern starts with message traffic spread evenly across the mesh; later on the XY routing strategy maintains that evenness. On the other hand, the adaptive algorithms select the routing paths based on *local*, short-term information. The 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 [5].

However, for most of the applications in real world, each node will communicate with some nodes much more compared to others. XY routing has serious deficiency 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. 5 (b-d) shows that XY routing is clearly outperformed by *odd-even* and *DyAD-OE*. Taking the results using *transpose1* traffic for instance (Fig. 5(c)), the network using XY saturates at an injection rate of 0.0167 packets/cycle. On the other hand, *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.

The effectiveness of *DyAD-OE* is confirmed by the fact that it continuously outperforms *odd-even* in terms of sustainable throughput in all these experiments. In fact, for the same traffic pattern and the injection rate, *DyAD-OE* achieves *shorter* average packet latency compared to *odd-even* throughout the experiments.

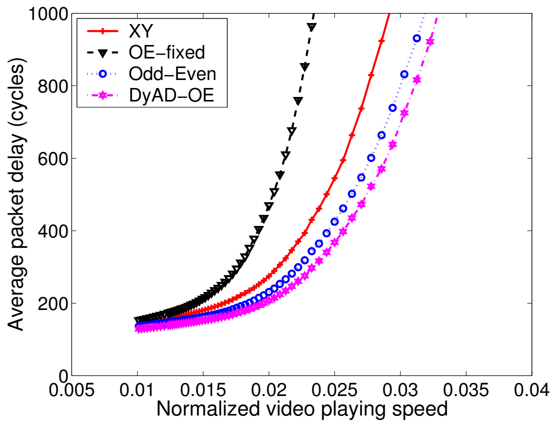
Another interesting fact to observe is that *DyAD-OE* does keep the advantage of deterministic routing when network is not congested. As shown in Fig. 5, *DyAD-OE* has the same average packet latency when network is not congested. On the other hand, compared to *DyAD-OE*, the average latency a packet experiences in *odd-even* is 14% higher compared to that in *DyAD-OE*, when the network is lightly loaded.

We also simulated the network under 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. 5 and are not included here due to space limitation.

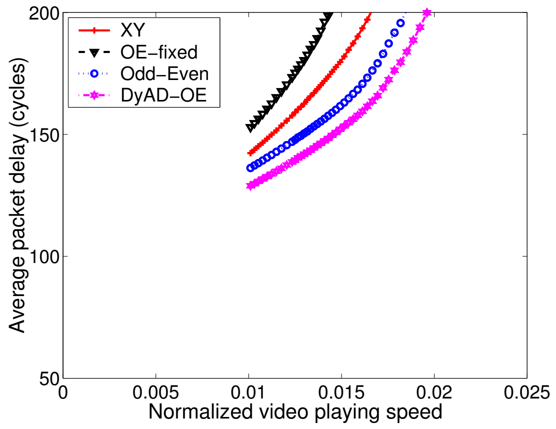
4.3 Performance Evaluation under Multimedia Traffic

Real world traffic (both in macro and on-chip networks) frequently exhibits patterns with self-similarity and long-range dependencies [15][16]. This can be quite a different scenario compared to the traffic patterns used in subsection 4.2. 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 using different video clips to retrieve 9 traces and recorded the arrival data at the motion compensation module in the decoder. 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 use uniform traffic which sends packets to other PEs with equal probability. We incrementally



(a)



(b)

Figure 6: Performance Evaluation under Multimedia Traffic

increase the packet injection rates to mimic the case when the system increase its speed of playing the video clips. The results are shown in Fig. 6.

Plotted in Fig. 6(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 4.2. *DyAD-OE* always performs the best under all playing rates, while *odd-even* performs second best.

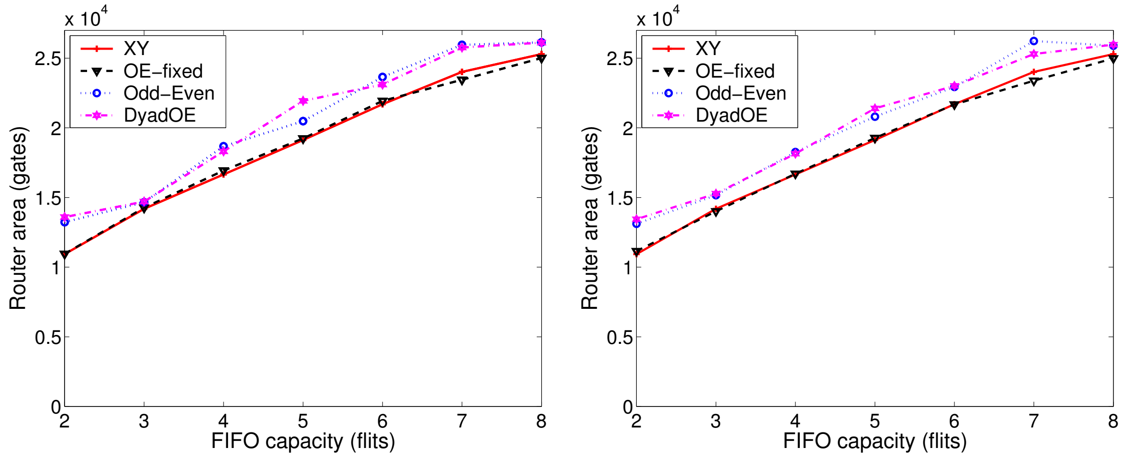
Fig. 6(b) shows a magnified view of Fig. 6(a) for the low speed region (that is, the region corresponding to 0.01 to 0.025 playing speed). It is interesting to note that unlike the simulation results using random traffic patterns where *DyAD-OE* has the same latency as *XY* and *oe-fixed*, *DyAD-OE* enjoys now a such shorter latency compared to that of *XY* and *oe-fixed*, even at very low playing speeds. Indeed, even *odd-even* has a better latency than *XY* and *oe-fixed*. This behavior is due to the bursty nature of the multimedia traces.

5 Prototype Router Designs

Since a *DyAD-OE* router actually combines *oe-fixed* and *odd-even* routing schemes, the silicon resources needed to construct it are larger than those needed by *oe-fixed* and *odd-even* alone. To evaluate the overhead of *DyAD-OE*, we decided to implement several designs to check the actual performance/area trade off.

The overhead mainly comes from two source: the mode controller and the port controller. The *mode controller* is extremely simple as it can be simply implemented by OR-ing the congestion flags. Compared to those in *odd-even* routers, the *port controllers* in *DyAD-OE* need not only to generate the congestion flag, but also to perform the routing decision in the *oe-fixed* mode. Given the fact the output directions of the *oe-fixed* are just a subset of those in the *odd-even*, it is not necessary to duplicate the logic for the *oe-fixed* mode routing decision, as it has already been implemented by the *odd-even* and only one mode will be active at any time. What remains is the logic inside the port controller which generates the congestion flag. The cost of implementing it is actually also very small, as only a comparator is really needed to check whether or not the number of the flits in the FIFO has reached the specified threshold; this cost is almost negligible.

Based on the above discussion, it appears that implementing a *DyAD-OE* router requires an almost negligible additional cost compared to an *odd-even* router. To justify our claim, we actually implemented all four versions of routers (*XY*, *oe-fixed*, *odd-even* and *DyAD-OE*) using a 0.16μm technology, with a clock rate of 333MHz. In our design, FIFOs are implemented using registers in order to achieve better performance/power efficiency. Their respective area (in gates) is shown in Fig. 7 for different FIFO capacity. In all the designs, each input port has a fixed link width of 32 bits. The flit size is set to be 32 bits as well.

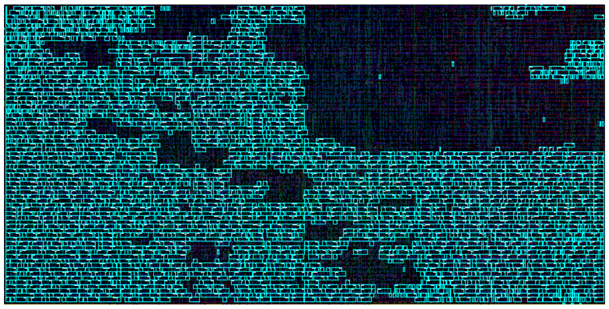


(a) Even Column Router Size (b) Odd Column Router Size

Figure 7: Router Sizes with Different FIFO Capacity

Except for *XY* routing, the routers in odd and even columns need to perform different routing decisions. Thus, the implementations are divided into even-column routers and odd-column routers, with their sizes shown in Fig. 7(a) and Fig. 7(b), respectively. Fig. 8(a) and Fig. 8(b) give the actual layout for an even-column and odd-column routers, respectively, when a FIFO capacity of 8 flits is used. The area occupied by FIFO is highlighted in Fig. 8; this clearly shows that the FIFO area takes the majority of the resources.

As shown in Fig. 7, 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, the overhead is indeed negligible (less than 1%). For all the configurations shown in Fig. 7, the overhead compared to *odd-even* is below 7%.


(a) Even Column Router Layout

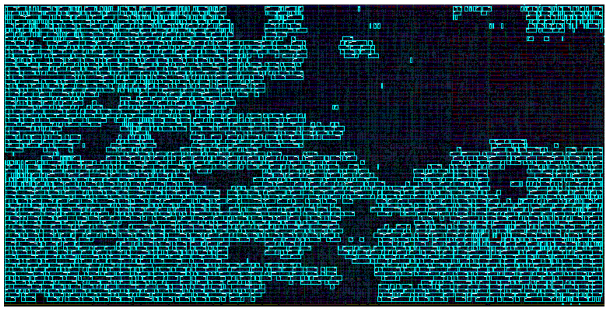

(b) Odd Column Router Layout

Figure 8: Layout of the *DyAD-OE* Router

6 Conclusion and Future Work

We presented a novel NoC routing concept (called *DyAD*) which combines the low latency of the deterministic routing (at low network load) and the high throughput of the adaptive routing. An instance of the *DyAD-OE* based on this new concept 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.

The preference for *DyAD-OE* over *odd-even* is justified by our prototype *DyAD-OE* router which adds less than 7% overhead to the corresponding *odd-even* router design in terms of used gates but performs much better in handling various traffic characteristics.

As explained in the paper, *DyAD* routing is a new concept rather than a particular design or implementation choice. In order 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 carefully customized to match the given application traffic characteristics. This remains to be done as future work. We also plan to prototype other applications in order to evaluate *DyAD* with various real world traffic patterns.

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] J. Duato. New theory of deadlock-free adaptive routing in wormhole networks. *IEEE Tran. on Parallel and Distributed Systems*, 4(12):1320–1331, Dec. 1993.

[5] C. J. Glass and L. M. Ni. The turn model for adaptive routing. In *25 Years ISCA: Retrospectives and Reprint*, pages 441–450, 1998.

[6] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Oberg, M. Millberg, and D. Lindqvist. Network on a chip: an architecture for billion transistor era. In *Proc. of the IEEE NorChip Conf.*, Nov. 2000.

[7] P. Kermani and L. Kleinrock. Virtual cut-through: a new computer communication switching technique. In *Computer Networks*, volume 3, pages 267–286, Sept. 1979.

[8] S. Kumar, A. Jantsch, M. Millberg, J. Oberg, J. Soininen, M. Forsell, K. Tiensyrj, and A. Hemani. A network on chip architecture and design methodology. In *Proc. Symposium on VLSI*, pages 117–124, April 2002.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] Semiconductor Association. *The International Technology Roadmap for Semiconductors (ITRS)*, 2001.

[14] K. G. Shin and S. W. Daniel. Analysis and implementation of hybrid switching. *IEEE Tran. on Computers*, 45(6):684–692, 1996.

[15] G. Varatkar and R. Marculescu. On-chip traffic modeling and synthesis for MPEG-2 video application. *IEEE Tran. on VLSI*, 12(1), Jan. 2004.

[16] 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.