Evaluation of On-chip Networks Using Deflection Routing

Zhonghai Lu
zhonghai@imit.kth.se

Mingchen Zhong
mingchen@kth.se

Axel Jantsch
axel@imit.kth.se

Department of Electronic, Computer and Software Systems
Royal Institute of Technology in Sweden

ABSTRACT

Deflection routing is being proposed for networks on chips since it is simple and adaptive. A deflection switch can be much smaller and faster than a wormhole or virtual cut-through switch. A deflection-routed network has three orthogonal characteristics: *topology*, *routing algorithm* and *deflection policy*. In this paper we evaluate deflection networks with different topologies such as *mesh*, *torus* and *Manhattan Street Network*, different routing algorithms such as *random*, *dimension XY*, *delta XY* and *minimum deflection*, as well as different deflection policies such as *non-priority*, *weighted priority* and *straight-through* policies. Our results suggest that the performance of a deflection network is more sensitive to its topology than the other two parameters. It is less sensitive to its routing algorithm, but a routing algorithm should be minimal. A priority-based deflection policy that uses global and history-related criterion can achieve both better average-case and worst-case performance than a non-priority or priority policy that uses local and stateless criterion. These findings are important since they can guide designers to make right decisions on the deflection network architecture, for instance, selecting a routing algorithm or deflection policy which has potentially low cost and high speed for hardware implementation.

Categories and Subject Descriptors: B.7.2 [Integrated Circuits]: Design Aids — Simulation; C.4.1 [Performance of Systems]: Design studies — Deflection routing

General Terms: Design, Performance

Keywords: Network-on-Chip, System-on-Chip communication network, Performance evaluation

1. INTRODUCTION

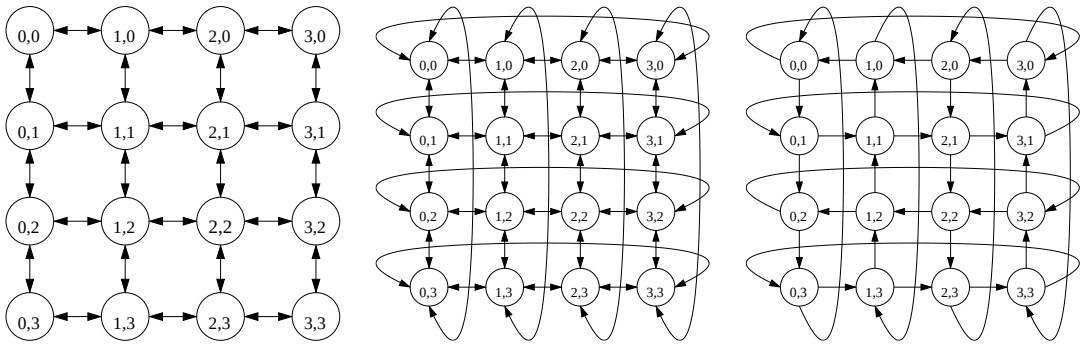
During the past five years, Network-on-Chip (NoC) [4, 10, 11, 13] has been suggested as a systematic approach to cope with the future System-on-Chip (SoC) design challenges such as interconnect difficulty, design productivity and stringent power constraints. Instead of using dedicated wires like bus interconnects, on-chip networks route packets to communicate data. First, by allowing concurrent transactions, NoC can potentially overcome the bandwidth limitation of buses to deal with the alarming design complexity enabled by the steady technology scaling [4]. Second, a network with a well-defined interface could serve as a communication platform to provide various services with diverse guarantees to upper-layer IP blocks. The possibility of the architectural reuse may reduce the Non-Recurring Engineering (NRE) cost and shrink time-to-market since an efficient domain-specific platform may be shared across many applications [8]. Third, the concurrent computation-communication structure and localized clock synchronization can efficiently reduce power consumption and thus satisfy power constraints, which are increasingly becoming design bottleneck and have to trade off with performance [12].

On-chip network is the core of network-on-chip. In general, a network has a much larger design space than a bus. A packet-switched network may be characterized by its topology, flow control and routing algorithm. For on-chip networks, a regular topology is favored against an irregular topology since it simplifies routing and layouting, and enables to modularize switches. Deflection routing [1, 2, 3, 6, 7, 10, 11] is a decentralized and adaptive routing mechanism. The distinguishing feature of a deflection switch lies in that it has no buffer queues. Packets are always on the run cycle by cycle, routing towards their destinations. Upon contending for links, packets with a lower priority will be *misrouted* to unfavored links according to a *deflection policy*. Since it has no buffer and flow management, a deflection switch can be designed with higher speed and lower cost than a wormhole or virtual cut-through switch. Thanks to its fully adaptive nature, it is also possible to avoid hot spots and provide fault-tolerance in the network.

The performance of a deflection network is the function of three parameters, namely, the topology, routing algorithm and deflection policy. In this paper, we explore the design space by means of cycle-true simulation. It is crucial to explore these design alternatives since they are implemented in hardware and may not be dynamically configurable or too costly to permit dynamic configuration. Therefore identifying the significance of each factor and evaluating their alternatives play a vital role in helping designers to make right decisions on the network architecture.

In the sequel, we brief the related work in Section 2. Then we describe and exemplify the characteristics of a deflection network in Section 3. Experimental results are reported in Section 4. Finally we draw conclusions in Section 5.

GLSVLSI'06, April 30–May 2, 2006, Philadelphia, PA, USA.



(a) A mesh (b) A torus (c) A MSN

Figure 1: Three 4×4 network topologies

2. RELATED WORK

Deflection routing, which is also called *hot potato* routing, has its root in [1]. It has been widely used in optical networks where buffering optical signals is too expensive. Because of its simplicity and adaptivity, it is also adopted and implemented in communication networks embedded in massively parallel computers such as the Connection machine [7]. It was initially proposed for on-chip networks in the Nostrum NoC [10, 11]. As projected in [11], a deflection switch can run 2.38 GHz with a gate count of 19370 in 65 nm technology.

Both average-case and worst-case performance in deflection networks have been analytically studied in [2, 3, 6]. Greenberg and Goodman [6] presented two approximate performance models to estimate the steady state throughput and average packet latency in the Manhattan Street Network (MSN). A deflection network is deadlock free but it has to avoid livelock, i.e., a packet continues routing in the network but never reaches its destination. A lot of work was focusing on deriving a performance bound based on assumptions of network traffic. Brassil and Cruz [3] derived upper bounds on the evacuation time of batch admissions [1] on an arbitrary topology and bounds on worst-case transit delay for hypercube networks admitting packets continuously. Borodin *et al.* presented bounds of deterministic algorithms for many-to-many traffic patterns in hypercube, mesh and torus networks [2]. A performance comparison between wormhole networks and deflection networks on chip can be found in [13].

Our work evaluates different kinds of deflection networks by simulation. The evaluation results are aimed to help make architectural decisions for on-chip deflection networks.

3. DEFLECTION ROUTING

A packet-switched deflection network is characterized by its topology, routing algorithm and deflection policy. We describe them with examples in this section.

3.1 Topology

The topology of a network defines how the network nodes are physically connected. It generally influences *network diameter*, *switch degree*, *link capacity* and *layout & wiring* for any kinds of networks. The network diameter is the length of the maximum shortest path between any two nodes. The switch degree is the number of input/output ports of a switch. The link capacity represents the number of links of the network. The layout & wiring refer to how the switches may be laid out and how links between switches may be wired. For deflection networks, the network topology has implications on two additional network properties, *deflection index* [3] and *don't-care density* [6].

- *Deflection index* is the largest number of hops that a single deflection adds to a packet's shortest path.
- *Don't-care density* is the percentage of destination nodes to which a source node has more than one non-overlapped shortest path to send packets. As a packet has multiple preferred links to take out of a node, whichever preferred link to take is regarded as a favorable choice. Hence the choice of link at this node for the packet is don't-care.

We consider two dimensional regular topologies since lower dimension and regular structure have advantages in simplifying the routing controller of switches, wiring, as well as potentially refraining Deep SubMicron (DSM) effects over wires [4, 10]. Examples are the 2D torus network proposed in [4] and the 2D mesh network suggested in [10]. Specifically we consider three 2D topologies with the same number of nodes, namely, 2D mesh, 2D torus and the Manhattan Street Network (MSN) [6]. Along each of the two dimensions, there are K nodes. The mesh network has bi-directional links between nodes. But it has no toroidal connections. The torus network can be viewed as a mesh network with wrap-around connections. The MSN has a uni-directional 2D torus structure. But on a MSN, horizontal and vertical paths alternate in direction. Similarly to [6], for a MSN, we constrain K is a multiple of 4. The three topologies are depicted in Figure 1. We qualitatively compare them in Table 1.

Note that the switches in the torus and MSN networks can be connected with equal-length of wires [4]. The don't-care density in Table 1 is calculated as follows: The three topologies are all node-symmetric. We can pick up any node

[1]Batch admission refers to that packets are admitted at a time slot in a batch without subsequent admissions.

	Mesh	Torus	MSN
Diameter	$2(K\text{-}1)$	K	$K+1$
Switch degree	2,3,4	4	2
Deflection index	2	2	4
Link capacity	$4K(K\text{-}1)$	$4K^2$	$2K^2$
Don't-care density (K=4)	60%	73%	60%

Table 1: The topological factors of three networks

to calculate the don't-care density. For example, on the $K \times K$ mesh, any node has K^2-1 destination nodes. Among them, $2(K-1)$ nodes lie either on the same row or column as the source node. The source node sending packets to the $2(K-1)$ nodes has only one shortest path, but sending packets to the other $(K-1)^2$ nodes has two non-overlapped shortest paths. Therefore the don't-care density of the mesh is $(K-1)^2/(K^2-1)$. When $k=4$, it equals 60% (9/15).

3.2 Routing Algorithm

A routing algorithm determines the path and thus link a packet is to be delivered. Obviously a packet should be delivered along its shortest path whenever possible to reduce latency and increase throughput. For deflection routing, a routing algorithm determines a packet's favorable path and link. A deflection occurs only when a packet has to deviate from its shortest path, no matter whether a routing algorithm is minimal or not.

We consider four routing algorithms: Random, Dimension XY, Delta XY and Minimum deflection. Using random algorithms for on-chip network communication is beneficial for fault-tolerance, as discussed in [5].

- Random: A switch randomly chooses an available path to send packets.
- Dimension XY: A packet tries to first route along the X axis and then the Y axis.
- Delta XY: it routes packets by the minimal number of hops along the X (Δ_x) and Y (Δ_y) axis a packet has to travel. Δ_x/Δ_y is the difference along the X/Y axis between a packet's source and destination address. Δ can be negative. If both desired links are available, it randomly chooses one. It differs from Dimension XY in that it does not prefer the X against the Y axis.
- Minimum deflection: it minimizes the occurrence of deflection at each hop. When a switch prepares to send packets, the switch calculates all possible permutations of packets' emission. Then it chooses the arrangement that the minimum number of packets will be deflected.

Among the above routing algorithms, the Dimension XY and Delta XY route packets with best-effort along their shortest paths, and thus are minimal. The others are not minimal. The routing by the Random algorithm is probability-based. While making routing decisions, the Minimum deflection algorithm tries to minimize the number of deflections by the local and oblivious [2] criterion. Note that it is not appropriate to separate a deflection policy from this routing algorithm. Since it makes routing and deflection decisions all at once, we can view Minimum deflection as both a routing algorithm and a deflection policy.

[2]Oblivious means stateless, i.e., do not consider the history of packet delivery, such as age and deflection times.

3.3 Deflection Policy

A deflection policy resolves packet contentions for links. Together with a routing algorithm, it determines packet-to-link assignment rules according to a pre-determined criterion. In addition, it can be used to design a livelock-free network. Since a higher priority packet wins link arbitration, the packet tends to reach its destination step-by-step deterministically. Nevertheless, it is difficult to derive an upper bound for arbitrary traffic patterns analytically [2].

In addition to the Minimum deflection, we consider both non-priority and priority-based deflection policies. For priority-based policies, we consider the straight-through [6] and a weight-based priority policy. With a priority policy, packet-to-link resolution is performed in favor of packets with a higher priority. A tie is resolved randomly.

- Non-priority: Misrouting decisions are made randomly. Packets have equal probability to be misrouted.
- Straight-through: A straight-through direction has a higher priority than a turn. The packet that arrives on the incoming row/column link is emitted on the outgoing row/column link.
- Weighted priority: The priority of a packet is based on multiple properties of the packet. It is explained in detail as follows.

The weighted priority takes into account a packet's multiple properties, such as *age*, *distance*, *deflection times*, and *default value*. Age indicates how long the packet has been alive in the network. A packet has a hop count field that records the number of hops (age) the packet has been routed. A higher hop count implies an elder age. The distance refers to the minimal number of links the packet has to travel from current node to its destination. Each packet also has an overhead field of deflection count, which bookkeeps the times of deflection during its delivery. In addition, packets may be of a different purpose. A default priority may be assigned to a packet from application by the source node. As all these properties can be meaningful for the packet priority, we use a weighted expression to calculate the priority. Each property i is associated with a weight w_i, and $\sum |w_i| = 1$.

$$P = w_a \times A + w_h \times H + w_d \times D + w_f \times F \qquad (1)$$

where P indicates the value of priority; A is the *age* of packet; H is the distance; D represents the times of deflection; F is the initial priority level; w_a, w_h, w_d and w_f refer to the weight of age, distance, deflection times and initial level, respectively. A weight embodies the impact that a certain packet property exerts on the packet priority. For example, if w_a=*0*, it implies the packet priority has nothing to do with its age; if $w_a > 0$, it means that an elder packet has higher priority than a younger one; if $w_a < 0$, a younger packet will result in a higher priority than an elder one. The absolute value of a weight $|w_i|$ reveals the significance of the property i on the packet priority.

A deflection policy can take advantage of don't-care packets. For example, on the mesh or torus, if packet G_1 goes from node (1,1) to (2,2), and packet G_2 routes via node (1,1) to (2,1), contention for the eastern link occurs by Dimension XY routing. But G_1 is a don't-care packet at node (1,1), it can be routed to the southern link without deflection and G_2 takes the eastern link. If a deflection policy does not

consider G_1's don't-care preference, G_2 may be misrouted if $P_{G_1} \geq P_{G_2}$. In implementation, we can use one extra bit in a packet to denote if it is don't-care. This attribute has to be checked and set at each hop. If it is asserted, the packet priority is temporarily negated for the local priority comparison, causing the don't-care packet to lose arbitration.

4. SIMULATION

In this section, we report results of simulations experimenting topology, routing algorithm and deflection policy.

4.1 Experimental setting

In order to evaluate the alternatives on network topology, routing algorithm and deflection policy, we construct 4×4 networks in our network-on-chip simulation environment [9]. This tool has a cycle-true NoC simulation kernel developed in SystemC. Besides it features a Graphical User Interface (GUI) which allows one to conveniently configure network parameters, traffic parameters, and then invoke kernel simulation. The deflection switch is a single-cycle packet-level model. Delivering packets through a switch takes exactly one cycle. The traffic pattern is uniformly-distributed random traffic. Each node sends traffic to other nodes with equal probability at a constant rate. The highest injection rate is one packet per cycle per node. Each node is equipped with a packet source queue. Packets are injected into the network through the source queue. The queue has conceptually infinite depth since it does not drop packets. The packet ejection model is ideal. Whenever a packet reaches its destination, it is ejected from the network immediately. In the case of multiple packets reaching destinations, all of them will be immediately sunk. Each simulation runs to the steady state, meaning that increasing simulation cycles does not change the results appreciably. Performance statistics are then collected at the steady state.

We measure both time-related and volume-related performance. For the time-related metrics, we consider *latency* T and *network delivery time* T_{net}. Latency T is counted from the instant a packet is injected into the source queue until that it reaches destination. The delivery time T_{net} is the time a packet routes in the network after leaving its source queue until reaching destination. Therefore latency T comprises network delivery time T_{net} and source queuing time T_{src} which is the time a packet waits in the source queue. The hop count of a packet gives its network delivery time. It is incremented by one for each hop. For the volume-related measurement, we consider *throughput*. It is defined as the average number of packets received per cycle in normalization with the number of nodes or links. We also measure *link utilization*, which is the average percentage of active/utilized links. It is important because the links are valuable network resources besides switches, and therefore should be efficiently used. The link capacity of a network gives an absolute constraint on network performance. An over-dimensioned network may use more links than necessary to improve performance. In addition, the number of active links directly relates to power consumption.

4.2 Topology

For this set of experiments, we set the routing algorithm to Dimension XY; the deflection policy is the Weighted priority policy with weight for age $w_a = 0.3$, for distance $w_h = 0.2$, for deflection count $w_d = 0.5$ and for initial priority $w_f = 0$. The deflection does not take advantage of don't-care packets. The performance results are shown in Figure 2.

As the packet injection rate r increases, the network delivery time increases (Figure 2(b)). The increase is not linear but rather exponentially. This trend sustains until the network is saturated. For the MSN and mesh, they saturate at $r = 0.35$ and $r = 0.6$, respectively. The torus does not saturate at even the highest rate $r = 1$, since it has twice ideal throughput as much as the mesh and MSN (The bisection bandwidth of the torus, mesh and MSN is 16, 8, and 8 packets/cycle, thus the ideal throughput under the uniform traffic is 2, 1 and 1 packet/cycle/node, respectively).

With minimal routing, the network delivery time T_{net} is related to the deflection count D and deflection index I by

$$T_{net} = D \cdot I + H_{min} \tag{2}$$

where H_{min} is the average shortest distance of traffic. For example, as can be seen in Figure 2(a), when $r = 1$, $D(mesh) = 1.15$, $D(MSN) = 0.66$ and $D(torus) = 0.9$. As $H_{min}(mesh) = 2.67$, $H_{min}(MSN) = 2.93$ and $H_{min}(torus) = 2$, we have $T_{net}(mesh) = 4.97$, $T_{net}(MSN) = 5.6$, and $T_{net}(torus) = 3.8$. These figures match those on Figure 2(b). At the very low injection rate ($r = 0.025$), the deflection count is not zero. This is because the nodes inject traffic into the network synchronously, leading to contentions even under low load. When $r > 0.58$, the mesh incurrs the highest number of deflections (Figure 2(a)). This is because the traffic tends to conjest in the center, not well-balanced like the torus and MSN, since it has no toroidal connections. Figure 2(c) implies that the source queuing time T_{src} ($T_{src} = T - T_{net}$) becomes extremely high when the network is saturated. From Figure 2(e), we can see that the link utilization in a deflection network can reach 100% upon network saturation.

The torus performs best because it has the highest link bandwidth and wrap-around connections to balance traffic. If we normalize the throughput in Figure 2(d) with the link capacity, it turns out that the MSN has the highest throughput per link before the network saturation (Figure 2(f)).

4.3 Routing Algorithm

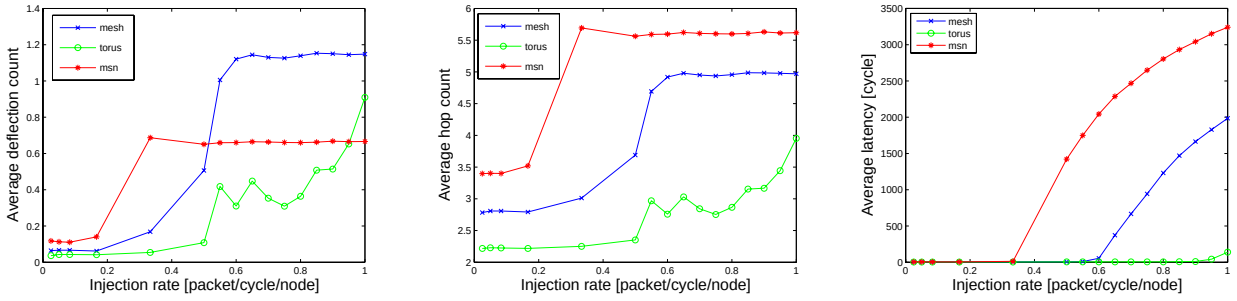
In this set of experiments, the topology is the mesh; the deflection policy is the Weighted priority policy used in section 4.2, but utilizes packets' don't-care preference. The results are depicted in Figure 3.

As can be seen, the Random algorithm performs worst in terms of latency and throughput since this algorithm does not guarantee packets to progress towards their destinations at each hop even there is no contentions for links. Even at the lower injection rate $r \approx 0.2$, the network is saturated and link is utilized 100%. The Dimension XY and Delta XY perform equivalently in latency, throughput and link utilization. The Minimum deflection algorithm performs a bit worse than Dimension XY and Delta XY, because its routing is not minimal and its deflection policy does not take packet delivery history into account. Although it minimizes deflection at each switch, the resulting overall performance is inferior when the network contention is high ($r > 0.7$).

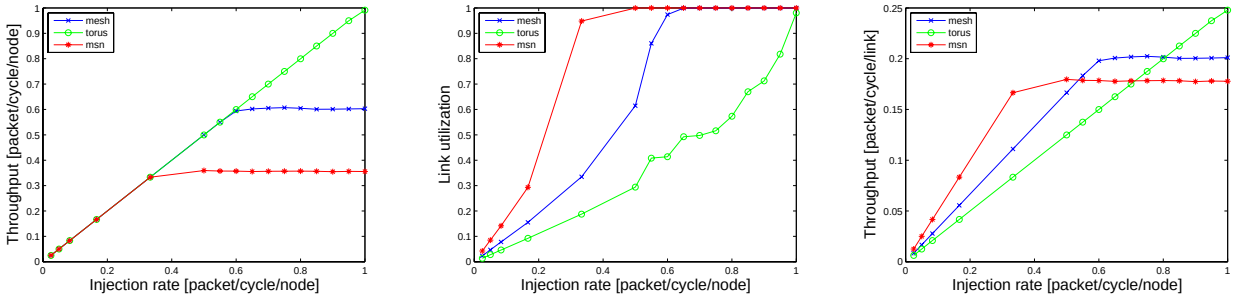
4.4 Deflection Policy

In this group of experiments, we use the mesh network. The routing algorithm is Dimension XY.

We compare the performance with the three deflection policies (Non-priority, Weighted priority and Straight-through)

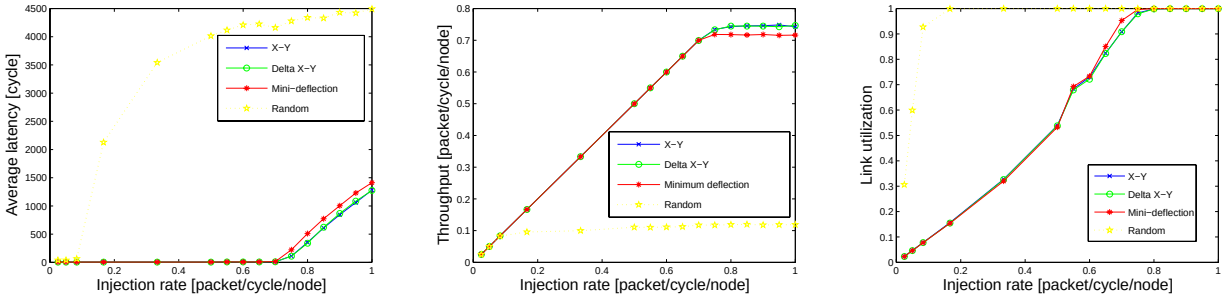


(a) Deflection count (b) Network delivery time (c) Latency



(d) Throughput (e) Link utilization (f) Link-normalized throughput

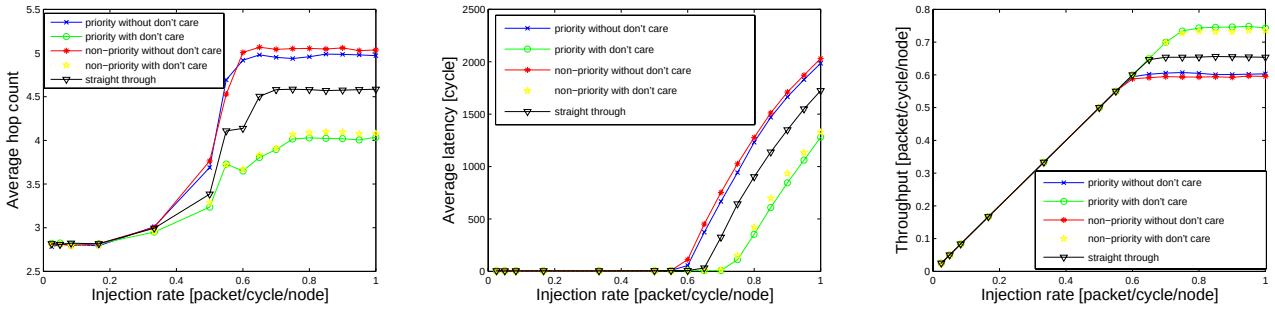
Figure 2: Performance of different topologies



(a) Latency (b) Throughput (c) Link utilization

Figure 3: Performance of different routing algorithms

in Figure 4. The priority weights use the same values as before. For the non-priority and priority policies, we consider both cases. One is with don't-care, the other without don't-care. As can be observed, the Weighted priority policy achieves better performance than the Non-priority policy. This is because distinguishing priority ensures higher-priority packets not misrouted, hence progressing faster to destinations. The Straight-through policy performs the midway between those with priority and those without priority, since it uses a sort of constant policy in favor of either the X or Y dimension to resolve contentions. More importantly, those policies considering don't-care achieves much better performance than those without considering don't-care. The saturation throughput is improved about 24% from 0.6 to 0.745 (Figure 4(c)). Meanwhile the network delivery time decrements about 1 cycle by 19% (Figure 4(a)). Since considering don't-care packets reduces deflections, packets are delivered faster and more. As a result, the reduction of source queuing time T_{src} is also significant (45%), as shown in Figure 4(b). We performed other experiments with different weighted values, for example, $w_a = 1$ and $w_d = 1$. Their performance are close to each other.



(a) Network delivery time (b) Latency (c) Throughput

Figure 4: Performance of different deflection policies

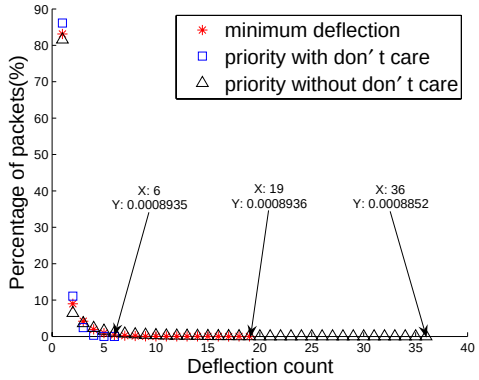


Figure 5: Distribution of deflection count

Given a topology, the deflection count D is the most crucial factor for network performance (Equation 2). We compare the distribution graphs of deflection count when $r = 0.7$ for the Priority without don't-care, Priority with don't-care and Minimum deflection policies in Figure 5. The priority is given to deflection count in this case, i.e., $w_d = 1$. The distribution graphs are subject to the similar envelope. As can be seen, a very large percentage of packets are deflected only once. However, there exist packets experiencing much more deflections. The observed maximum deflection count is 36, 6, 19, respectively. Clearly, considering the don't-care preference of packets can improve the worst-case performance.

5. CONCLUSION

Determining the architecture of a deflection network is not a simple task because there exist many alternatives concerning network topology, routing algorithm and deflection policy. In this paper we have evaluated those alternatives that are currently being proposed for on-chip networks. We can conclude from our results that the network topology is the most significant factor, since it directly determines the deflection index and don't-care density. As long as a routing algorithm chooses the shortest path, the performance of different algorithms is close to each other. To improve the average-case, worst-case behavior and resolve livelock, packets should be prioritized, and a deflection rule should use the priority information and take packets' don't-care attribute into account. We believe these conclusions can be guidelines to select a deflection network architecture.

Our future work is to include application-specific traffic in the evaluation framework so that we can make guidelines for deflection networks targeting domain-specific applications.

6. REFERENCES

[1] P. Baran. On distributed computing networks. *IEEE Transactions on Communication Systems*, March 1964.

[2] A. Borodin, Y. Rabani, and B. Schieber. Deterministic many-to-many hot potato routing. *IEEE Transactions on Parallel and Distributed Systems*, 8(6):587–596, 1997.

[3] J. T. Brassil and R. L. Cruz. Bounds on maximum delay in networks with deflection routing. *IEEE Transactions on Parallel and Distributed Systems*, 6(7):724–732, July 1995.

[4] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the 38th Design Automation Conference*, 2001.

[5] T. Dumitras and R. Marculescu. On-chip stochastic communication,. In *Proc. of Design, Automation and Test in Europe Conference*, March 2003.

[6] A. G. Greenberg and J. Goodman. Sharp approximate models of deflection routing in mesh networks. *IEEE Transactions on Communications*, 41(1), January 1993.

[7] W. D. Hills. The Connection machine. *Scientific American*, 256(6), June 1987.

[8] K. Keutzer, S. Malik, A. R. Newton, J. M. Rabaey, and A. Sangiovanni-Vincentelli. System-level design: Orthogonalization of concerns and platform-based design. *IEEE Transaction on Computer-Aided Design of Integrated Circuits*, 19(12):1523–1543, December 2000.

[9] Z. Lu and A. Jantsch. Traffic configuration for evaluating networks on chips. In *Proceedings of the 5th International Workshop on System on Chip for Real-time applications*, July 2005.

[10] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch. Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip. In *Proceedings of the DATE Conference*, February 2004.

[11] E. Nilsson and J. Öberg. Reducing peak power and latency in 2D mesh NoCs using globally pseudochronous locally synchronous clocking. In *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*, September 2004.

[12] D. Pamunuwa, J. Öberg, L.-R. Zheng, M. Millberg, A. Jantsch, and H. Tenhunen. A study on the implementation of 2D mesh based networks on chip in the nanoregime. *Integration - The VLSI Journal*, 38(2):3–17, October 2004.

[13] T. T. Ye, L. Benini, and G. D. Micheli. Packetization and routing analysis of on-chip multiprocessor networks. *Journal of Systems Architecture*, 50:81–104, February 2004.