Evaluation of Deflection Routing on Various NoC Topologies

Chaochao Feng[1,2*], Jinwen Li[1], Zhonghai Lu[2], Axel Jantsch[2], Minxuan Zhang[1]

[1] School of Computer, National University of Defense Technology, Changsha 410073, China
[2] Department of Electronic Systems, Royal Institute of Technology, Stockholm, Sweden
* Email: fengchaochao@nudt.edu.cn

Abstract

In this paper, we propose two novel deflection routing algorithms for de Bruijn and Spidergon NoCs and evaluate the performance of the deflection routing on 5 NoC topologies with different synthetic traffic patterns. We also synthesize the routers in various NoC topologies with TSMC 65nm technology. The evaluation results illustrate that the performance of deflection routing is susceptible to the network topology and traffic pattern. The results can also guide the NoC architect to choose the suitable NoC topology for the specific application.

1. Introduction

Network-on-Chip (NoC) has already become a potential solution for the interconnection issue of the Multi-Processors System-on-Chip (MPSoC). Recently, deflection routing has been used in NoC due to the fact that no buffer is needed in the router, which can reduce the area cost significantly [1][2]. Deflection routing is a full adaptive routing algorithm, which can be used in any topologies with the same number of input and output ports. In this paper, we propose two novel deflection routing algorithms for de Bruijn and Spidergon NoCs and evaluate the performance of the deflection routing on 5 NoC topologies with different synthetic traffic patterns. We also synthesize the routers in various NoC topologies with TSMC 65nm technology. The evaluation results illustrate that the performance of deflection routing is susceptible to the network topology and traffic pattern. The results can also guide the NoC architect to choose the suitable NoC topology for the specific application.

2. Related work

Deflection routing was first proposed in [3] and widely used in optical network, which can reduce the cost of optical buffer [4]. Lu et al. [1] have evaluated the deflection routing on Mesh, Torus and MSN on-chip networks with different routing and deflection policies under only uniform random traffic pattern. Moscibroda and Mutlu [2] have proposed two bufferless routing schemes (FLIT-BLESS and WORM-BLESS) for NoC. In the FLIT-BLESS scheme, each flit contains a header and is routed independently. The WORM-BLESS scheme combines the deflection routing with wormhole routing. It can be concluded from the simulation results that FLIT-BLESS is slightly better than WORM-BLESS. In this paper, we also use the flit-level deflection routing as the baseline routing algorithm.

3. Deflection routing for various NoC topologies

3.1 Various NoC topologies

Five NoC topologies are considered in this paper to evaluate the performance of the deflection routing. The 2D Mesh is the most popular topology used for NoC, such as Tile64 [5] and Teroflops [6]. Torus can be denoted as k-ary 2-cube with k nodes along each dimension [7]. The difference between the Mesh and Torus networks is that the routers at the edges in Torus are connected to the routers at the opposite edge through wrap-around links. Fig. 1(a) and (b) show the 4×4 Mesh and Torus respectively. The Manhattan Street Network (MSN) is a degree-2 network, in which the odd rows/columns have links in one direction and the even rows/columns have links in the opposite direction [8]. A 4×4 MSN network is shown in Fig. 1(c). The de Bruijn graph [9] denoted as DB(r,k) has $N=r^k$ nodes with diameter k and degree $2r$. A node I in DB(r,k) can be represented by ($i_{k-1}i_{k-2}\ldots i_1i_0$), where $i_j \in \{0,1,...,r-1\}$, $0 \le j \le (k-1)$. The nodes connected to I are represented by ($i_{k-2}i_{k-3}\ldots i_0p$) and ($pi_{k-1}i_{k-2}\ldots i_1$), where p=0,1,...,(r-1). Here, we consider DB(2,4), shown in Fig. 1(d), as an example. The Spidergon NoC topology is a degree-3 network proposed by ST Microelectronics [10]. The Spidergon network connects an even number of nodes $N=2n$, n=1,2,..., which consists of a bidirectional ring in both clockwise and anti-clockwise directions. In addition, a cross link connects each node i ($0 \le i \le N$) to node ($i+n$) *mod* N. Fig. 1(e) shows a Spidergon NoC with 16 nodes.

3.2 Routing algorithm

In deflection routing, the routing process can be divided into two parts: routing computation and output allocation. Routing computation gets the productive direction(s) to the destination. Incoming packets are prioritized by the hop counts the packet has been routed in the network. The router makes output allocation for each packet from the highest priority to the lowest. A load-aware technique [11], which considers the load information of neighboring routers, is introduced in the output allocation to balance the network load. Each router transmits the number of packets processed in the last 4

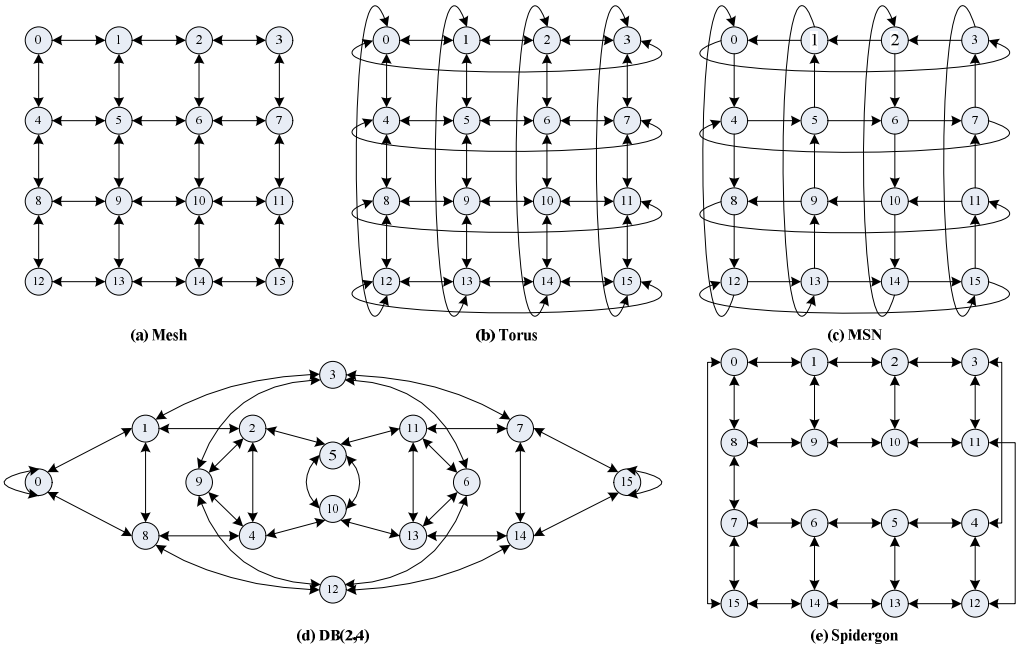

Figure 1. Various NoC topologies

cycles as the load information to its four neighbors. If there is more than one productive output available, the router will choose one of them with the smallest traffic loads. If two or more packets contend for one productive output port, the one with higher priority will be routed through the productive output port and other packet(s) will be deflected to a free output port with the smallest traffic loads.

For 2D Mesh and Torus networks, the productive direction(s) are calculated based on the minimum number of hops to the destination along the *X* and/or *Y* dimension ($\Delta x/\Delta y$). If both desired directions are available, the router will choose the one output with the minimum traffic loads to route the packet. For MSN network, the packet is first routed along *X* dimension and then *Y* dimension. In the case of contention, the packet with higher priority will take the preferred output port, while the other packet will be deflected to the other output port.

We propose novel routing computation algorithms for de Bruijn and Spidergon NoCs. Source routing has been proposed in de Bruijn NoC to perform the routing computation at the source node [9]. However, it cannot be used in deflection routing, since the routing path may be changed dynamically due to the contention in deflection routing. We propose a distributed routing computation algorithm for de Bruijn NoC. For DB(2,4), each router ($i_3i_2i_1i_0$) has at most 4 links (*L-0, L-1, R-0* and *R-1*) connecting 4 neighboring routers ($i_2i_1i_0$0), ($i_2i_1i_0$1), (0$i_3i_2i_1$) and (1$i_3i_2i_1$) respectively. The routing algorithm calculates the productive direction based on the length of the *L-path* or *R-path*. The *L-path*/*R-path* is defined as left/right-shift the current node address with 0 or 1 to get the destination address. It can be calculated based on finding the common *L*/*R-string* between the current and destination node addresses. For example, from node (1011) to (0110), the *L-string* and *R-string* are 011 and 10 respectively. The length of the *L*/*R-path* is the length of the address minus the length of the *L*/*R-string*. Fig. 2 shows the pseudo code of the routing computation function. When a packet reaches a router, the *L-path* and *R-path* are first calculated. If the length of *L-path* is smaller than that of the *R-path*, the productive direction is set to be *L-0* or *L-1*, otherwise the productive direction is set to be *R-0* or *R-1*. If the length of *L-path* is equal to the length of *R-path*, both *L-direction* and *R-direction* can be productive directions.

For Spidergon NoC with $N=2n$ nodes, the pseudo code of routing computation function is shown in Fig. 3. Each router in Spidergon NoC has three ports (*Left*, *Right* and *Across*). The productive direction is calculated according to the distance between the current and destination nodes.

3.3 Deadlock and livelock avoidance

Deflection routing is deadlock-free due to the fact that packets never have to wait in a router. Because of the non-minimal routing characteristic, deflection routing must avoid livelock by limiting the number of misroutings. The age-based prioritization rule considering the hop count of the packet as the priority is used to avoid unlimited misroutings effectively. The oldest packet can always be routed through the productive direction to the destination. Once the oldest packet reaches the destination, another packets become the oldest, thus livelock can be avoided.

Routing computation function for de Bruijn NoC
Input: *cur_addr*, *dst_addr* (current and destination address)
Output: *productive_direction*

```
1: |L_path| ← N; L_bit ← dst_addr[N-1]
2: |R_path| ← N; R_bit ← dst_addr[0]
3: for i in N-1 downto 1 loop //calculate the length of the L_path
4:   if the highest i bits of dst_addr = the lowest i bits of cur_addr then
5:     |L_path| ← N - i
6:     L_bit ← dst_addr[N-i-1]
7:     break;
8:   end if
9: end loop
10: for i in N-1 downto 1 loop //calculate the length of the R_path
11:   if the lowest i bits of dst_addr = the highest i bits of cur_addr then
12:     |R_path| ← N - i
13:     R_bit ← dst_addr[i+1]
14:     break;
15:   end if
16: end loop
17: end if
18: if |L_path| < |R_path| then
19:   productive_direction ← L[L_bit]
20: else if |L_path| > |R_path| then
21:   productive_direction ← R[R_bit]
22: else
23:   {productive_direction} ← {L[L_bit], R[R_bit]}
24: end if
```

Figure 2. Routing computation for de Bruijn NoC

Routing computation function for Spidergon NoC
Input: *distance* between current and destination nodes
$|distance| \le N/2$
Output: *productive_direction*

```
1: if distance = 0 then
2:   productive_direction ← Local
3: else
4:   if 0 < distance ≤ n/2 then
5:     productive_direction ← Right
6:   else if -n/2 ≤ distance < 0 then
7:     productive_direction ← Left
8:   else if n/2 < distance ≤ N/2 or -N/2 < distance ≤ -n/2 then
9:     productive_direction ← Across
10:  end if
11: end if
```

Figure 3. Routing computation for Spidergon NoC

4. Evaluation

In this section, we evaluate the performance of the deflection routing algorithm on 5 NoC topologies in terms of throughput and average latency under different synthetic traffic patterns.

4.1 Experimental setup

The simulations are performed on a cycle-accurate NoC simulator which can be configured with different topologies. Six synthetic traffic patterns [7] are used in the simulations, as shown in Table 1. For uniform random traffic, each node sends packets randomly to other nodes with an equal probability $\lambda=1/N$, where N is the number of nodes in the network. Transpose, bit complement, bit reverse, bit rotate and shuffle traffic patterns belong to the bit permutation traffic. In bit permutation traffic, the node address is denoted as n bits and the destination address is computed by permuting the bits of the source address.

Table 1. Synthetic traffic patterns

Name	Description
Uniform random	$\lambda=1/N$
Transpose	$d_i=s_{i+n/2 \bmod n}$
Bit complement	$d_i=\neg s_i$
Bit reverse	$d_i= s_{n-i-1}$
Bit rotate	$d_i=s_{i+1 \bmod n}$
Shuffle	$d_i=s_{i-1 \bmod n}$

We measure the throughput and average packet latency of various networks in the simulations. The throughput of the network (*TP*), measured in packets/cycle/node, is defined as the saturation point of the network which means the maximum accepted traffic. It can be calculated as follow:

$$TP = \frac{Total\ received\ number\ of\ packets}{(Number\ of\ nodes) \times (Total\ time)} \quad (1)$$

where *Total received number of packets* refers to the number of packets successfully arriving at their destination nodes, *Number of nodes* is the total number of nodes in the network and *Total time* is the total measurement time (in clock cycles) in the simulation. The packet latency T is calculated by equation (2), where T_{net} is the *network delivery time* which is the hop count the packet being routed (1 hop is defined as the packet has passed a router) and T_{src} is the time a packet waiting in the source queue.

$$T = T_{net} + T_{src} \quad (2)$$

4.2 Throughput

Fig. 4 illustrates the throughput of the five networks with 6 synthetic traffic patterns. For uniform random, transpose and bit complement traffic patterns, the Torus network has the highest throughput. For bit reverse, bit rotate and shuffle traffic patterns, the throughput of the DB(2,4) network can be 1 packet/cycle/node, which means all nodes can receive one packet every cycle. The throughput of the Torus network can also be 1 packet/cycle/node under bit reverse traffic pattern. The MSN network achieves the lowest throughput under all 6 traffic patterns. The Mesh network achieves the medium throughput except for bit complement traffic pattern. For bit complement traffic pattern, the throughput of the Mesh network is slightly lower than that of the Spidergon network.

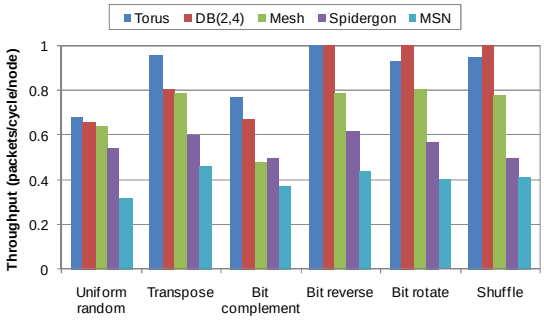

Figure 4. Throughput of various NoC topologies

4.3 Latency

Fig. 5(a)-(f) show the average packet latency of the 5 NoC topologies with 6 synthetic traffic patterns respectively. For uniform random and bit complement traffic patterns, the average packet latency of the Torus network is less than that of the other networks. For other traffic patterns, the DB(2,4) network achieves the smallest packet latency. In addition, the average packet latency of the DB(2,4) network under bit reverse, bit rotate and shuffle traffic patterns is fixed at whatever packet injection rate. The reason lies in that due to the connection nature of the de Bruijn graph, packets can be routed without contention in the three traffic patterns. The Mesh network can achieve medium packet latency except for bit complement and bit reverse traffic patterns. For bit complement traffic pattern, the average packet latency of the Spidergon network is slightly less than that of the Mesh network and for bit reverse traffic pattern, the Spidergon network achieves less average latency than the Mesh network before the network reaches the saturation point.

4.4 Hardware cost

The deflection routers are developed with VHDL. We synthesize the routers with TSMC 65nm technology. The area results for a router with the clock frequency constraint at 500MHz in various NoC topologies are shown in Table 2. The router in the Torus network has the largest area, while the router in the MSN network has the smallest area since it has only three input/output ports. From the view of hardware overhead, de Bruijn network is a better choice for deflection routing, because it can achieve a high throughput and low packet latency with a moderate area cost.

Table 2. Area for a router in various NoC topologies

	Area (μm^2)
MSN	12406
Spidergon	19503
Mesh	34319
DB(2,4)	16054
Torus	34901

5. Conclusion

In this paper, we evaluate the performance of the deflection routing on 5 NoC topologies with different synthetic traffic patterns and also implement the deflection routers with TSMC 65nm technology. We can conclude from the evaluation results that the performance of the deflection routing is more susceptible to the network topology and traffic pattern. A suitable NoC topology can be chosen for specific application based on the evaluation results.

Acknowledgments

The research is partially supported by the National Natural Science Foundation of China under Grant No. 60970036, No. 60873212 and No. 61003301.

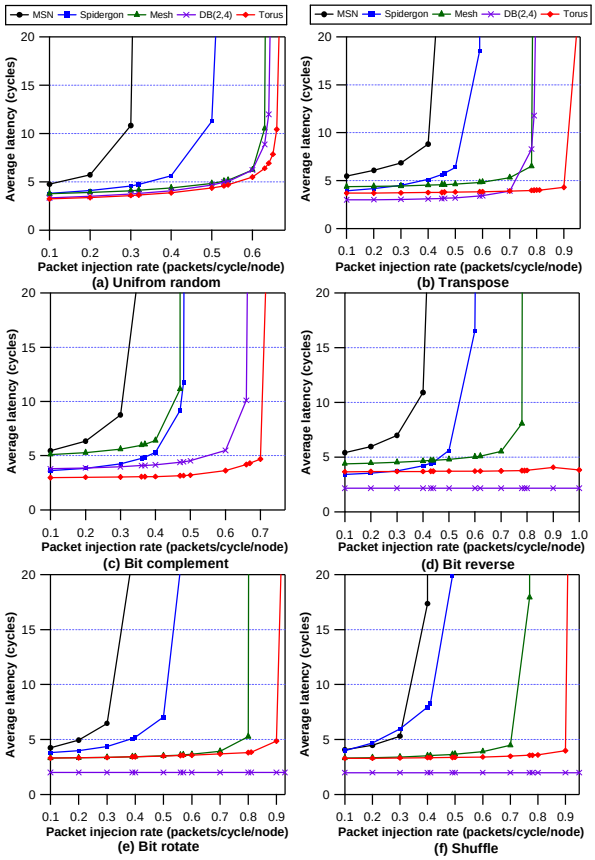

Figure 5. Average latency of various NoC topologies

References

[1] Z. Lu, et al, Evaluation of on-chip networks using deflection routing, GLSVLSI, p.363 (2006).

[2] T. Moscibroda and O. Mutlu, A case for bufferless routing in on-chip networks, ISCA, p.196 (2009).

[3] P. Baran, On distributed communications networks, IEEE Trans. on Comm., 12(1), p.1 (1964).

[4] P. Chich and J. Cohen, Unslotted deflection routing: a practical and efficient protocol for multihop optical networks, IEEE Trans. on Networking, 9(1), p.47 (2001).

[5] D. Wentzlaff, et al, On-chip interconnection architecture of the tile processor, IEEE Micro, 27(5), p.15 (2007).

[6] Y. Hoskote, et al, A 5-ghz mesh interconnect for a teraflops processor, IEEE Micro, 27(5), p.51 (2007).

[7] W. Dally and B. Towles, Principles and practices of interconnection networks, Morgan Kaufmann, (2003).

[8] N. Maxemchuk, Routing in the manhattan street network, IEEE Trans. on Comm., 35(5), p.503 (1987).

[9] M. Hosseinabady, et al, Application of de bruijn graphs to noc design, DSNoC, p.346 (2007).

[10] F. Dubois, et al, Spidergon stnoc design flow, NoCs, p.267 (2011).

[11] E. Nilsson, et al, Load distribution with the proximity congestion awareness in a network on chip, DATE, p.1126 (2003).