Triplet-based Topology for On-chip Networks

WANG ZUO, ZUO QI, LI JIAXIN
School of Computer Science and Technology
Beijing Institute of Technology
Beijing, 100081
China
qiushui@bit.edu.cn, Zqll27@bit.edu.cn, starforce@bit.edu.cn

Abstract: - Most CMPs use on-chip network to connect cores and tend to integrate more simple cores on a single die. As the number of cores increases, on-chip network will play an important role in the performance of future CMPs. Due to the tradeoff between the performance and area constraint in on-chip network designs, we propose the use of triplet-based topology in on-chip interconnection networks and demonstrate how a 9-node triplet-based topology can be mapped to on-chip network. By using group-caching protocol to exploit traffic locality, triplet-based topology achieve lower latency and energy consumption than 2D-MESH. We run multithreaded commercial benchmarks on multi-core simulator GEMS to generate practical traffics and simulate these traffics on network simulator Garnet. Our experiment results show that triplet-based network can increase the work-related throughput by 3%~11% and reduce average network latency by 24%~32% compared with 2D-MESH, with the router energy consumption reduced by 13%~16% and the link energy consumption reduced by 14%~16%.

Key-Words: - On-chip network, Cache protocol, Network latency, Energy consumption, Performance, Mapping

1 Introduction

In order to utilize the increasing number of transistors available in modern VLSI technology, processor designs tend to integrate more simple cores instead of complicated processor on a single die. As the number of cores increases in CMPs, on-chip networks are widely used to provide efficient communication between cores.

There are two different ways to implement on-chip network. One is to use low-radix networks, such as 2D-MESH found in the RAW processor [1], the TRIPS processor [2], Intel's Teraflops [3] and the Tile64 processor [4]. With simple topologies, low-radix networks can be mapped to on-chip network with compact physical layout using conventional Manhattan routing in horizontal and vertical directions. Especially in tiled CMP designs, the longest wire in the system is no greater than the length or width of a tile [1]. Although low-radix networks can be mapped efficiently, they lead to several disadvantages such as high network latency caused by long diameter. The other is to use high-radix networks, such as butterfly network [5]. With shorter network diameters, high-radix networks can achieve the improvement on network latency and energy consumption by reducing hop count between cores. However, wiring requirements of high-radix networks complicate physical layout which leads to higher area cost [6]. We think that using high-radix networks may not be a cost-effective way, while exploiting traffic locality on low-radix network can achieve both goals in performance and area cost. Thus, we propose the use of triplet-based topology in on-chip interconnection networks and describe how triplet-based network can achieve performance improvement and energy consumption degradation with the help of group-caching protocol.

This paper is organized as follows. In Section 2, we describe triplet-based topology and its routing algorithm. Section 3 demonstrates how a 9-node triplet-based topology can be mapped to on-chip network and discusses the impact of various link latencies on routing algorithms. Average network latency model as well as static link load model is proposed for triplet-based network in Section 4. Group-caching protocol is presented in Section 5. We evaluate triplet-based network and compare it with 2D-MESH in Section 6. Related work is discussed in Section 7 and we conclude in Section 8.

2 Triplet-based Topology and its New Routing Algorithm

The basic component of triplet-based topology [7] is a full-connected triplet which consists of three nodes connecting with each other directly. As shown in Fig.1, three full-connected triplets form a

9-node network while larger networks can be implemented by replacing each node with a full-connected triplet iteratively. The node degree of triplet-based topology is 3 while that of 2D-MESH is 4. As buffers occupy approximately 60~80% of router area [4] [6], triplet-based network consumes less router area than 2D-MESH due to fewer buffers needed by its lower node degree. Besides, the router energy consumption of triplet-based network also benefits from low node degree.

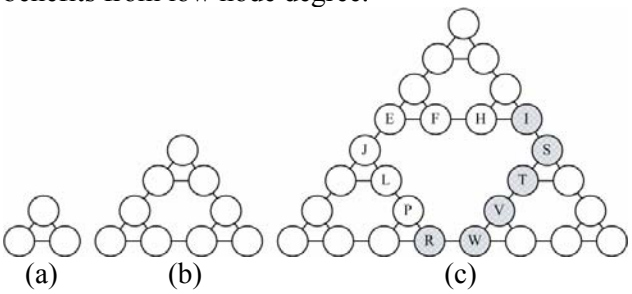


Fig.1 Triplet-based topology

The basic routing algorithm presented in paper [7] is described as follows:

1. If node X and node Y locate in the same full-connected triplet, the minimal path from node X to node Y is X—Y.

2. If node X and node Y locate in the same sub-network consisting of 3^{K-1} full-connected triplets and locate in the different sub-networks consisting of 3^{K-2} full-connected triplets, packets from node X to node Y chooses the minimal path, referred as MIN(X—Y_1), to enter the sub-network which node Y locates, then choose the minimal path, referred as MIN(Y_1—Y), to traverse to node Y. Y_1 is the first node that packets from the sub-network which node X locates to the sub-network which node Y locates will pass.

3. Repeat step 2 iteratively to implement routing for MIN(X—Y_1) and MIN(Y_1—Y).

Although the basic routing algorithm is efficient, there are still some disadvantages it. First, the basic routing algorithm is not a minimal routing algorithm due to its non-global decision. For example, because node I and node R locate in the two neighbouring 9-node sub-networks, the basic routing algorithm will choose I—H—F—E—J—L—P—R as the path from node I to node R while the minimal path from node I to node R is I—S—T—V—W—R. Although the difference existed between the basic routing algorithm and minimal routing algorithm is very tiny [7], it still has negative impact on network performance. Second, the basic routing algorithm does not take deadlock into account.

In order to solve the non-minimal problem of the basic routing algorithm, we introduce the use of the transition node. When node X need send a packet to node Y, we insert the transition node Z, which locates in the shortest path, into the packet (the transition node is calculated beforehand and stored in each node). The routing process from node X to Y is divided into two parts, node X to Z and node Z to Y. For example, as shown in Fig.1, when node I send packets to node R, we choose node S as the transition node and insert it into packets. Packets from node I to node R are first transferred to node S with the help of the basic routing algorithm and then transferred to node R. The benefits of these changes are obvious: it almost inherits all the advantages of the basic algorithm, such as the low complexity of router design and the fast routing process (normal table-based algorithm need read rout table in each node and the basic routing algorithm will not), while implementing the minimal routing.

In order to solve the deadlock problem, we assign different weights to the links in different directions and change the basic routing algorithm as follows: during the route computation phase, if there are two minimal paths, the output link with the minimal weight is selected. As shown in Fig.2, the links in 0 degree direction are assigned weight 3, links in 60 degree direction are assigned weight 4 and links in 120 degree direction are assigned weight 5. Although there are two shortest paths F—D—B—C and F—H—G—C from node F to node C, packets from node F to node C will be first sent to node H because the weight of F—H is lighter than F—D, then choose path H—G—C to arrive node C. According to the same analysis, packets from node C to node F will be first sent to node B, then choose path B—D—F to arrive node F. By combining the above rules and virtual channel technique, deadlock will not occur in the ring B—C—G—H—G—D—B.

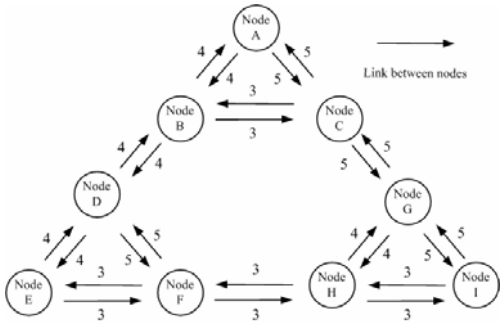


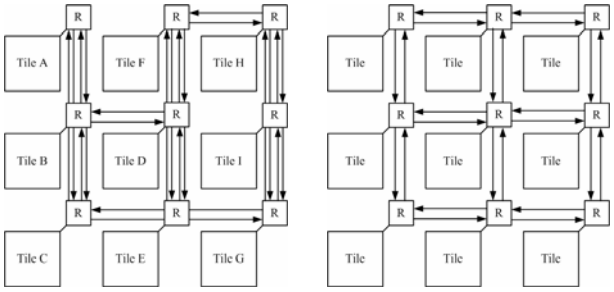
Fig.2 Link weight assignment of the 9-node triplet-based topology

We should note that the basic routing algorithm is based on the assumption that all links have the same latency. However, when mapping triplet-based topology to on-chip network using conventional routing in horizontal and vertical directions, some

links will have higher latencies than others thus the minimal paths which we define according to topology will be changed. We discuss the impact of various link latencies on routing algorithm in the next Section.

3 Mapping Triplet-based Topology to On-chip Network

Although there are diagonal routing techniques [8] [9] which can be used to map triplet-based topology more efficiently, we still adopt conventional routing in horizontal and vertical directions since physical implement as well as area cost evaluation of triplet-based network is not the topic concerned by this paper. Fig.3 (a) describes how a 9-node triplet-based topology can be mapped to on-chip network in tiled CMPs. The nodes locating in a full-connected triplet are placed vertically in a row, while the links connecting full-connected triplets are placed horizontally. Fig.3 (b) describes the physical layout of 2D-MESH. The area costs of triplet-based network is close to that of 2D-MESH though the former's layout is a little more complicated than the latter's.



(a) (b)

Fig.3 Physical layout comparison between triplet-based network and 2D-MESH

As shown in Fig.3, the wires between tile A and tile C have almost twice the length of those between tile A and tile B. Although we can reduce the adverse impact of long wires on performance by inserting repeaters or pipeline registers into the longer wires [6], the impact of various wire latencies on routing algorithm still exists. We made the following changes to the basic routing algorithm presented in Section 2. The first is that the minimal path from tile F to tile C is changed from path F—H—G—C to path F—D—B—C because long wires between tile G and tile C add more latency to path F—H—G—C. The second is that the minimal path from tile B to tile H is changed from B—C—G—H to B—D—F—H.

4 Traffic Models for Triplet-based Network

We assume that each node has the opportunity to communicate with any other node in our model. Parameter p ($0<p<1$) is used to denote acceptance probability with which a node will receive the packet from other nodes in the same full-connected network. In order to analyze the impact of locality on network traffic, we introduce locality factor α ($0<\alpha\leq 1$). If node X and node Y locate in the same sub-network which consists of 3^{K-1} full-connected triplets and locate in the different sub-networks which consist of 3^{K-2} full-connected triplets, the acceptance probability of node Y to receive the packet from node X is $p*\alpha^{K-1}$.

In most CMP designs, the network traffic is used to maintain the cache coherence, such as the cache coherence between L1 and L1 (protocol dependent), between L1 and L2, between L2 and L2 (protocol dependent) and the cache coherence between L2 and memory. As the interleaving technique is applied to both shared L2 banks and shared memory banks, the traffic frequencies for nodes are almost the same in the case of the same load for each node. $\alpha=1$ means all nodes in the network have the same acceptance probability which probably matches the traffic pattern under banked and shared L2 cache design.

4.1 Average Network Latency Model

According to the physical layout described by Fig.3 (a), we set the latency of short wires to 1 cycle and set the latency of long wires to 2 cycles. According to the configuration of network simulator present in Section 6, router latency is set to 4 cycles. Table 1 lists the zero-load latency between nodes according to the updated basic routing algorithm presented in Section 3.

Table 1 Zero-load latency between nodes according to the updated basic routing algorithm

	A	B	C	D	E	F	G	H	I
A	XX	9	10	14	19	19	16	22	21
B	9	XX	9	9	14	14	15	19	20
C	10	9	XX	14	19	19	10	16	15
D	14	9	14	XX	9	9	20	14	19
E	19	14	19	9	XX	10	21	15	20
F	15	14	19	9	10	XX	15	9	14
G	16	15	10	20	21	15	XX	10	9
H	22	19	16	14	15	9	10	XX	9
I	21	20	15	19	20	14	9	9	XX

Based on the assumption that each node probably rejects the same number of flits into network and each node's outgoing messages are equal to its coming messages, the average zero-load network latency of 9-node triplet-based network is depicted as follows:

$$L_{average} = 9 + \frac{3+199\times\alpha}{9+27\times\alpha} \quad (1)$$

According to Equation (1), network latency will benefit from traffic locality. The lowest network latency between nodes is 9 cycles which probably matches the latency between the nodes in the same full-connected triplet. It is worth mentioning that Equation (1) is based on the assumption that each node has the opportunity to communicate with any other node in the network. However, as described in paper [21] that a type of packets may only have a limited selection of receivers among all nodes in reality, thus Equation (1) needs some changes to adapt real-life traffics. For example, according to the group-caching protocol presented in the next Section, each node only communicates with the nodes in the same full-connected triplet and the nodes locating in the same place of different full-connected triplets (node A, node D and node G locate in the same place of different full-connected triplets). Under such condition, Equation (1) needs to be modified as:

$$L_{average} = 9 + \frac{3+65\times\alpha}{9+9\times\alpha} \quad (2)$$

4.2 Link Load Model

In Equation (3), we define the load of a link as a sum of messages passing this link. Table 2 lists the 9-node network link load distribution based on Equation (3) and the updated basic routing algorithm presented in Section 3.

$$T_j = \sum_{i=1}^{n} T_{(N_X \xrightarrow{pass\ link\ j} N_Y)} \quad (3)$$

Taking link C—B as an example, because the minimal path between node C and node F is C—B—D—F, the communications between the two nodes will inject α*t messages to link C—B during time t. By analogy, we calculate the messages injected by the communications between node C and node D, node C and node E, node A and node D, node A and node E, node A and node F during time t, then add up all the messages. As shown in table 2, the traffic load of link C—B is p*(1+5*α)*t.

Table 2 9-node triplet-based network link load distribution

Link ID	load		Link ID	load
A-B B-A A-C C-A	1+3*α		F-D	1+7*α
D-E E-D E-F F-E	1+3*α		G-C	8*α
G-I I-G H-I I-H	1+3*α		C-G F-H	9*α
H-G	1+4*α		B-D H-F	10*α
G-H C-B	1+5*α		D-B	11*α
B-C D-F	1+6*α		—	—

We can see from Table 2 that if there no locally distributed traffics (α = 1), the load of links between full-connect networks will be 2 times higher than that of links in full-connect networks; if traffics are distributed locally in each full-connect network, for example α = 0.5, the link load distribution will be more symmetrical. We can also see from table 2 that the revised minimal routing algorithm discussed in Section 3 makes traffics concentrate on some links. For example, the load of link D—B is a much heavier than that of G—C. The impact of the revised minimal routing algorithm on link load distribution will be also alleviated as α value decreases.

5 Group-Caching for Triplet-based Network

Banked and shared L2 cache design is a good choice to minimize the number of off-chip accesses when the latency of accessing remote L2 banks is not significant [16]. For example, [14] and [15] use low latency crossbar to connect distributed L2 banks. However, if we use low-radix networks, such as 2D-MESH, as the connection substrate, banked and shared L2 cache design seems a poor choice. As data are spread evenly across all banks, only a small fraction of references are satisfied by the local bank while other references may need jump several hops to access remote L2 banks. Hence, the average L2 access latency is heavily influenced by network diameter. For triplet-based network, using banked and shared L2 cache design is worse. Frequent accessing remote L2 banks will block large mount of traffic on the links connecting different full-connected triplets, and the blocked traffic will introduce more latency due to congestion.

We think that there are two policies to reduce the hop count. First, when a thread tries to load a page

into L2 cache, this page should be loaded into the L2 banks which are close to the core running this thread. Second, if two or more threads share the same data, multiple copies should be loaded into different L2 banks with each copy placed close to its corresponding core. Besides, we also need to control the number of data copies properly, otherwise large number of copies will prick up the inefficient use of the aggregate L2 cache capacity so as to increase the number of off-chip accesses.

A valuable idea to reduce the number of remote L2 accesses is cooperative caching [16]. As most L2 accesses are satisfied by local private L2 cache, remote L2 accesses are reduced compared with banked and shared L2 cache design. Moreover, as L2 caches cooperate caching just like a shared L2 cache, off-chip accesses are also significantly reduced compared with private L2 cache design.

In this paper, we use group-caching to improve the performance of 9-node triplet-based network. We extend private L2 cache in [16] to a cache group consisting of 3 interleaved banks belonged to the nodes in the same full-connected triplet. Each cache group behaves like a shared L2 cache independently while the cache coherence among cache groups is maintained by distributed directories implemented by duplicating the tag architecture of L2 caches. As most L2 accesses are served by local cache group, remote L2 accesses are also significantly reduced compared with banked and shared L2 cache design.

When the L2 cache write actions occur, the invalidation messages are also multicasted to the other cache groups to mark other data copies invalid. When a L2 cache read access is missed in local cache group, query messages will be sent to the other two cache groups to check whether the accessing data (dirty or clean) is already loaded into chip. If so, an off-chip memory access is avoided by directly obtaining the data from another cache group. As shown in Fig.4, when a L2 cache read miss occurs in L2 bank C, two query messages will be simultaneously sent to L2 bank F and L2 bank I to judge whether carry an off-chip access or not (because the address of cache group is interleaved across its 3 cache banks, the data, which can be cached in L2 bank C, can only be cached in L2 bank F or L2 bank I).

As cache group provide more cache blocks to hold useful data on-chip than single private L2 cache, implementing cache-to-cache transfer of clean data is enough to reduce off-chip accesses compared with private L2 cache design. Compared to cooperative caching [10], group-caching still has some advantages. Cooperative caching uses Central Coherence Engine (CCE) to backup the tags of all private L2 cache and adopts special point-to-point network to connect CCE and cores. Even if we ignore the extra cost of special network and the possible performance bottle-neck of CCE, placing every core in the same distance to CCE is a hard work.

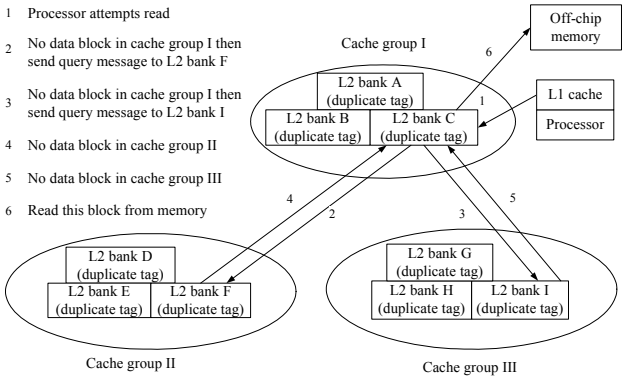


Fig.4. Group-caching design

6 Experiment

6.1 Experiment Methodology

We use Princeton's Garnet network simulator [10] to evaluate the 9-node triplet-based network and compare it with 2D-MESH. Garnet models the detailed features of state-of-the-art on-chip networks and thus is suitable for low-level interconnection network evaluations. We select fixed-pipeline model in our experiment. Table 3 lists parameters used by fixed-pipeline model. Besides, in order to obtain the practical locality factor, source-destination matrix is also introduced to trace the number of flits traversing between nodes.

Table 3 Parameters of Garnet's fixed-pipeline model

Network configuration	Parameters
Flit size	16
Buffer size	4
Pipeline stage	5-stage
VCs per virtual network	4
Number of virtual network	5

Some papers [5] [22] use synthetic batch traffics, such as uniform random (UR), to evaluate network performance. However, these benchmarks do not represent real-life traffics. In this paper, we run multithreaded commercial benchmarks on multi-core simulator GEMS [11] to generate practical traffics. GEMS is a full-system execution-driven

simulator based on Simics [12]. Table 4 lists the processor and cache/memory configurations used in GEMS. These parameters are independent of on-chip network and benchmarks. We create the network configuration file for 9-node triplet-based network via the user-defined network interface provided by GEMS. This configuration file records the triplet-based topology, the various link latencies and the link weights which are used by Garnet as network configuration. We also create the network configuration file for 9-node 2D-MESH.

Table 4 Processor and cache/memory parameters

Component	Parameters
Processor	UtraSPARCIII+, single issue
L1 I/D cache	64KB, 4-way, 128 byte/block, 3 cycles
L2 cache bank	2MB, 4-way, 128 byte/block, 12 cycles
Memory bank	1GB, 4KB/page,158 cycles

We use work-related throughput [19] as the metric to evaluate the network performance and use Orion [20] to evaluate the network power consumption. Orion is a power model integrated with Garnet to evaluate network power consumption using 100nm technology. The commercial benchmarks used in experiment include Apache (a static web server benchmark based on SURGE [13] running on top of Apache web server), SPECJBB2005 (a java server benchmark) and OLTP (a scaled down TPC-C benchmark to capture the performance behavior of OLTP workloads [18]). Besides, we evaluate the performance of multiple web servers on single CMP. These different web servers have the same configurations as Apache benchmark. Workload parameters for benchmarks are reported in Table 5.

Table 5 Workloads parameters

Benchmark (transactions)	Environment
Apache(10000)	2000files, 90 clients
JBB(20000)	SPECjbb2005, 14 warehouses
OLTP(30)	IBM DB2, mbenchkit, 10 clients
Multi-Apache(20000)	3 independent web servers

There are two protocols used in our experiment: MESI_SCMP_bankdirectory protocol (referred as MESI below) and group-caching protocol. The former one, which uses banked, shared L2 caches as directories, is widely used by tiled CMPs while the latter one is presented to exploit the traffic locality of 9-node triplet-based network. In order to exploit traffic locality further, we use process bind API provided by operating system (here we use Solaris's psrset command) to bind different web severs to each cache group. As shown in Fig.5, three web severs are bond to different cache groups. Under such condition, as there is probably no data sharing among different web servers, almost all L2 cache accesses are satisfied by its local cache group thus having more significant traffic locality.

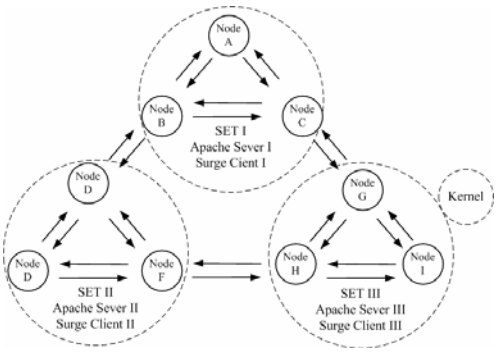


Fig.5 Diagrams of generating traffic locality

6.2 Evaluation

6.2.1 Performance Evaluation

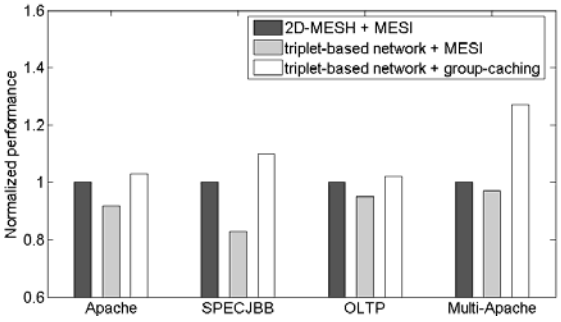


Fig.6 Performance comparisons between triplet-based network and 2D-MESH

Fig.6 compares the performance between 9-node 2D-MESH and triplet-based network. When using MESI protocol, the performance of 9-node triplet-based network is not as good as that of 2D-MESH. The throughout degradation of SPECJBB is much more significant that of other benchmarks, which means SPECJBB is more sensitive to network latency. However, with the help of group-caching to exploit locality, triplet-based network increases the throughout by 3%~11% on Apache, SPECJBB and OLTP benchmarks compared with 2D-MESH. Especially, if top-level applications and low-level architecture can be greatly matched, such as binding different web servers to each cache group in the

environment of multiple web servers, triplet-based network achieves more significant improvement, probably 27% on the throughout compared with 2D-MESH.

6.2.2 Power Evaluation

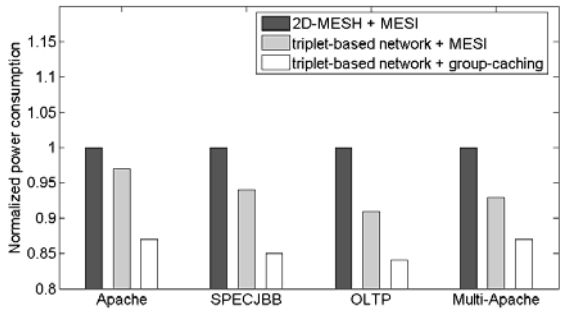


Fig.7 Total router power comparisons between 2D-MESH and triplet-based network

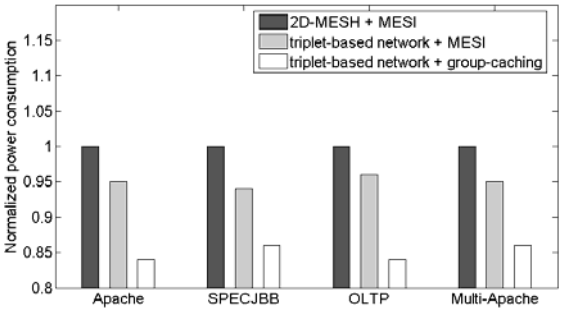
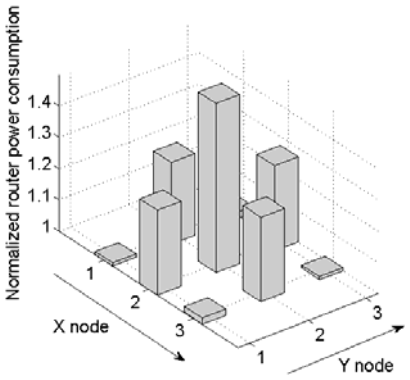


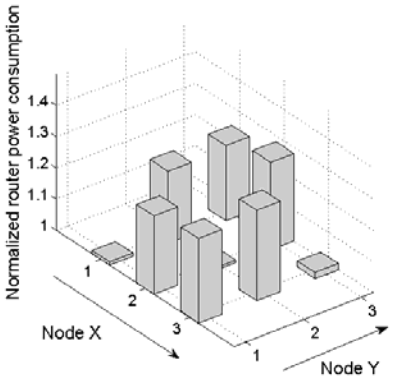
Fig.8 Total link power comparisons between 2D-MESH and triplet-based network

Total router power comparisons and total link power comparisons are shown in Fig.7 and Fig.8. Both of the power consumptions are normalized to that of 2D-MESH using MESI protocol. As shown in the two figures that triplet-based network using MESI protocol reduces the total router power consumption by 3%~9% and the total link power consumption by 4%~6% compared with 2D-MESH. We think the reason is, as more congestion blocks the communications between cores, power decreases as performance decreases. However, the significant degradation in power consumption (router power reduced by 13%~16% and link power reduced by 14%~16%) achieved by group-caching protocol attributes to the reduced hop count.

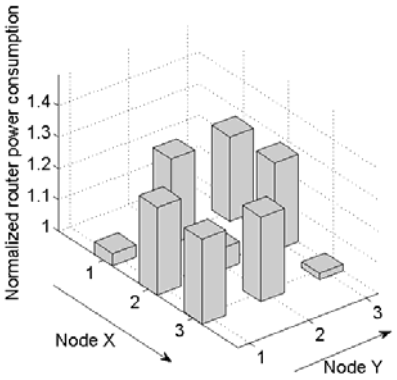
Fig.9 describes the router power distributions of 2D-MESH and triplet-based network. In 2D-MESH, the power distribution of routers is quite different from each other. As shown in Fig.9 (a), the power consumption of the center node is nearly 1.5 times than that of the four nodes in the corners. We can also see from Fig.9 (b) (c) that, the router power consumption of triplet-based network distributes more evenly under either MESI or group-caching protocol. We think that the different router power consumption probably reflects the loads for routers. With router load distributed more evenly, triplet-based network may benefit from the low congestion inside routers.



(a)



(b)



(C)

Fig.9 Router power distribution comparisons: 2D-MESH using MESI protocol (a), triplet-based

network using MESI protocol (b) and triplet-based network group-caching protocol (c)

6.2.3 Latency Model Verification

Table 6 Comparisons between statistical and theoretical average network latencies under MESI protocol

Benchmark	Statistical α	Theoretical latencies	Statistical latencies
Apache	0.9676	14.5673	14.8771
SPECJBB	0.9716	14.5728	14.7773
OLTP	0.9624	14.5601	14.7767
Multi-Apache	1.0579	14.6865	14.8250

Table 7 Comparisons between statistical and theoretical average network latencies under group-caching protocol

Benchmark	Statistical α	Theoretical latencies	Statistical latencies
Apache	0.1206	10.0747	10.1898
SPECJBB	0.3364	11.0674	11.1699
OLTP	0.1704	10.3363	10.3868
Multi-Apache	0.1907	10.4366	10.4751

We obtain practical locality factor α from source-destination matrix. The practical locality factor α probably equals to 1 for all benchmarks under MESI protocol while α varies with different benchmarks under group-caching protocol. We use Equation (1) and Equation (2) to calculate theoretical average network latencies and list them in Table 6 and Table 7. As shown in the two tables that the statistical values approximately match the theoretical values, while the little differences between them attribute to the latency caused by contention. Besides, we can also see from the two tables that, the group-caching reduces average network latency by 24%~32% compared with MESI protocol.

7 Related Works

Most proposed on-chip networks utilize low-radix networks such as 2D-MESH in the RAW processor [1]. In order to reduce network diameter, Balfour and Dally [6] proposed the use of concentrating mesh. James [16] also demonstrated how the flattened butterfly topology can be applied to on-chip networks to reduce network diameter as well as network energy consumption.

Except for improving on-chip network itself, some valuable designs were also presented to reduce hop count by improving cache management. A hybrid of private, per-processor tag arrays and a shared data array was used to implement controlled replication, in-situ communication and capacity steeling in [17]. Jichuan Chang used private L2 cache and all private L2 caches cooperate caching just like a shared L2 cache in [16]. As the frequently-accessed data is held close to the requestor in the two designs, the number of remote L2 accesses is significantly reduced.

Liqun Cheng [23] presented the work that combined efforts on both network design and cache management to improve the performance of CMPs. They proposed an interconnection composed of wires with varying latency, bandwidth, and energy characteristics, and advocate intelligently mapping coherence operations to the appropriate wires. We also try to achieve performance improvement as well as energy degradation through the combination of triplet-based network and group-caching design.
Soteriou [21] presented the work that modeled on-chip network traffic based on real-life traffic traces obtained from full system simulations of three different CMP architectures. In its empirically-derived model, they derive the hop count distribution model on the assumption that an optimal mapping should place communicating nodes as close as possible. However, such optimal mapping is hard to implement in reality.

Diagonal routing technologies were discussed in [8] [9]. Diagonal routing with X-architecture, which uses diagonal routing in 45 and 135 degree directions, was reported in [8]. It was reported that the path delay improved by 19.8% and the area reduced by about 10% by applying diagonal routing in conjunction with conventional Manhattan routing in horizontal and vertical directions. In another routing model, Y-architecture based on 0, 60 and 120 degree directions was proposed [9].

8 Conclusion and Future Work

In this paper, we propose the use of triplet-based topology in on-chip interconnection networks. Like most low-radix networks, triplet-based network has the same advantages such as low area cost and the same disadvantages such as low performance due to long diameters. According to our average network latency model, if most traffic concentrates in each group, the performance of triplet-based network will benefit from the reduced hop count. Thus, we propose group-caching to exploit the traffic locality

of 9-node triplet-based network. Our experiment results show that, with the help of group-caching protocol, triplet-based network increases work-related throughput by 3%~11% and reduces average network latency by 24%~32% compared with 2D-MESH, with total router power consumption reduced by 13%~16% and total link power consumption reduced by 14%~16%. Furthermore, if top-level applications and low-level architecture can be greatly matched, such as binding independent tasks to different cache groups, triplet-based network will achieve more significant improvement on performance and power.

Our future work focuses on the following three aspects:

The average network latency model presented in Section 4 only gives the estimation of static average network latency without taking congestion into account. Thus, when traffic load increase sharply, this model can not work well. Thus, our first work is adjusting the average network latency model to overload environment.

As conventional routing in horizontal and vertical directions leads to various link latencies for triplet-based network, we consider using Y-architecture [9] to do "perfect" mapping for triplet-based topology. Thus, our second work is evaluating the perfect-mapped triplet-based network.

We note the potential of 2D-MESH to exploit traffic locality. Based on the group conception, the basic component constructing 2D-MESH is a 4-node rectangle with one extra hop count required for the traffic between the nodes locating diagonally (neighbor nodes in a rectangle need 2 hops). Our third work is to evaluate the potential of 2D-MESH to exploit traffic locality and compare it with triplet-based network.

Acknowledgement

This paper is partially supported by Beijing Key Discipline Program

References:

[1] Taylor, M.B.; Psota, J.; A.; Shnidman, N.; Strumpen, V.; Frank, M.; Amarasinghe, S.; Agarwal, A.; Lee, W.; Miller, J.; Wentzlaff, D.; Bratt, I.; Greenwald, B.; Hoffmann, H.; "Evaluation of the Raw microprocessor: an exposed-wire-delay architecture for ILP and streams", *31st Annual International Symposium on Computer Architecture,* 2004. Proceedings. June 2004 Page(s):2 – 13.

[2] Gratz, P.; Changkyu Kim; Sankaralingam, K.; Hanson, H.; Shivakumar, P.; Keckler, S.W.; "On-Chip Interconnection Networks of the TRIPS Chip", *Micro, IEEE* Volume 27, Issue 5, Sept.-Oct. 2007 Page(s):41 - 50.

[3] Hoskote, Y.; Vangal, S.; Singh, A.; Borkar, N.; Borkar, S.; "A 5-GHz Mesh Interconnect for a Teraflops Processor *Micro", IEEE* Volume 27, Issue 5, Sept.-Oct. 2007 Page(s):51 – 61.

[4] Wentzlaff, D.; Griffin, P.; Hoffmann, H.; Liewei Bao; Edwards, B.; Ramey, C.; Mattina, M.; Chyi-Chang Miao; Brown, J.F.; Agarwal, A.; "On-Chip Interconnection Architecture of the Tile Processor" *Micro, IEEE* Volume 27, Issue 5, Sept.-Oct. 2007 Page(s):15 – 31.

[5] Kim, John; Balfour, James; Dally, William; "Flattened Butterfly Topology for On-Chip Networks" *Annual IEEE/ACM International Symposium on Micro architecture*, 2007. Page(s):172 – 182.

[6] Balfour, James; Dally, William J.; "Design tradeoffs for tiled CMP on-chip networks", *the International Conference on Supercomputing, Proceedings of the 20th Annual International Conference on Supercomputing*, 2006, p 187-198.

[7] Qiao Baojun; Shi Feng; "A New Routing Algorithm in Triple-Based Hierarchical Interconnection Network", *First International Conference on Innovative Computing, Information and Control, 2006.* Page(s):725 – 728.

[8] I. Mutsunori, T. Mitsuhashi, et al "A Diagonal-Interconnect Architecture and Its Application to RISC Core Deisgn" *ISSCC*, pp. 684-689, 2002.

[9] H. Chen, B. Yao, et al, "The Y-Architecture: Yet Another On-Chip Interconnect Solution" *ASPDAC*, pp. 840-846, 2003.

[10] http://www.princeton.edu/~niketa/publications/garnet-tech-report.pdf.

[11] Milo M.K. Martin, Daniel J. Sorin, Bradford M. Beckmann, Michael R. Marty, Min Xu, Alaa R. Alameldeen, Kevin E. Moore, Mark D. Hill, and David A. Wood "Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset", *Computer Architecture News*, September 2005.

[12] P.S.; Christensson, M.; Eskilson, J.; Forsgren, D.; Hallberg, G.; Hogberg, J.; Larsson, F.; Moestedt, A.; Werner, B.; "Simics: A full system simulation platform Magnusson" *Computer* Volume 35, Issue 2, Feb. 2002 Page(s):50 – 58.

[13] Paul Barford; Mark Crovella; "Generating representative Web workloads for network and server performance evaluation", *the 1998 ACM SIGMETRICS joint international conference on*

Measurement and modeling of computer systems table of contents Madison, Wisconsin, United States Pages: 151 – 160.

[14] Barroso, L.A.; Gharachorloo, K.; McNamara, R.; Nowatzyk, A.; Qadeer, S.; Sano, B.; Smith, S.; "Piranha: a scalable architecture based on single-chip multiprocessing" *27th International Symposium on Computer Architecture*, 2000 Page(s):282 - 293

[15] Kongetira, P.; Aingaran, K.; "Niagara: a 32-way multithreaded Sparc processor" *Micro, IEEE* Volume 25, Issue 2, March-April 2005 Page(s):21 – 29

[16] Jichuan Chang; Sohi, G.S.; "Cooperative Caching for Chip Multiprocessors" *33rd International Symposium on Computer Architecture*, 2006. 2006 Page(s):264 – 276

[17] Chishti, Z.; Powell, M.D.; Vijaykumar, T.N.; "Optimizing replication, communication, and capacity allocation in CMPs" *International Symposium on Computer Architecture*, 2005. Page(s):357 - 368

[18] Minglong Shao, Anastassia Ailamaki, Babak Falsafi. "DBmbench: Fast and Accurate Database Workload Representation on Modern Microarchitecture" *Conference of the Centre for Advanced Studies on Collaborative Research* 2005.

[19] A. R. Alameldeen, M. M. K. Martin, C. J. Mauer, K. E. Moore, M. Xu, D. J. Sorin, M. D. Hill, and D. A. Wood. "Simulating a $2M commercial server on a $2K PC", *IEEE Computer* 36(2):50–57, Feb. 2003.

[20] Hang-Sheng Wang; Xinping Zhu; Li-Shiuan Peh; Malik, S.; "Orion: a power-performance simulator for interconnection networks" *35th Annual IEEE/ACM International Symposium on Microarchitecture*, 2002 Page(s):294 - 305

[21] Soteriou, V.; Hangsheng Wang; "A Statistical Traffic Model for On-Chip Interconnection Networks" *14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, 2006 Page(s):104 – 116.

[22] Guerrier, P.; Greiner, A.; "A generic architecture for on-chip packet-switched interconnections" *Design, Automation and Test in Europe Conference and Exhibition*, 2000, Page(s):250 – 256.

[23] Liqun Cheng; Muralimanohar, N.; Ramani, K.; Balasubramonian, R.; "Interconnect-Aware Coherence Protocols for Chip Multiprocessors" *33rd International Symposium on Computer Architecture*, 2006 Page(s):339 – 351.