Cluster Comput (2013) 16:679–691
DOI 10.1007/s10586-012-0238-3

Asymmetrical topology and entropy-based heterogeneous link for many-core massive data communication

Yu-hang Liu · Ming-fa Zhu · Li-min Xiao · Jue Wang

Received: 14 April 2012 / Accepted: 30 October 2012 / Published online: 5 January 2013
© Springer Science+Business Media New York 2013

Abstract As the need for data processing and communication increases, and likewise, as the number of processing cores placed on a given single chip increases, improving the performance of interconnection networks is vital. In the present work, traditional topologies are re-examined. *Torus* is shown to be a good structure in terms of average latency and symmetry. When using *torus* in combination with high process levels, it is possible to design new, yet asymmetrical topologies that can meet the high communication performance requirements of many-core processors and also suit a large variety of traffic patterns. Firstly, this paper presents two novel and torus-like topologies called *xtorus* and *xxtorus*, which are evaluated by using both theoretical analysis and experimental simulation methods. For theoretical analysis, an algorithm for computing link path diversity and link entropy is given. The analysis shows that, compared with *mesh*, *xmesh* and *torus*, the proposed topologies have better properties in terms of diameter, average latency, throughput, and path diversity. Although more links are added, the number of links is of the same order of magnitude with that of *mesh*, *xmesh*, and *torus*. Proposed topologies also take advantage of increasingly higher levels of the VLSI process. Simulations on GEM5 reveal that *xtorus* has better scalability, and that its average latency is less than that of *mesh*, *xmesh* and *torus* by significant proportions respectively, particularly when the network scale is larger. Moreover, for different traffic patterns, its performance swing is less than that of *mesh*. Furthermore, in the present work, the proposed topologies are both asymmetrical and based on the entropy difference of the links in the topology. A strategy for heterogeneous link design is presented, which enables designers to trade off between delay, power and area according to a concrete integrated circuit design scene.

Keywords Many-core · Asymmetrical topology · Traffic pattern · Performance analysis · Link entropy · Heterogeneous links

Y.-h. Liu (✉) · M.-f. Zhu · L.-m. Xiao
Laboratory of Software Development Environment, Beihang University, Beijing 100191, China
e-mail: liuyuhang@cse.buaa.edu.cn

J. Wang
Supercomputing Center of Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China

1 Introduction

Network architecture is affected by both application requirement and physical process level.

On the application requirement side, data generated by scientific activities as well as commercial applications from a diverse range of fields has been increasing exponentially. Hence, data needs to be processed and communicated even faster. The petascale computing era has come, and future exascale technologies are also being researched. The real performance of the first ten supercomputers in TOP500 (published in June 2012) all attained more than one Petaflops, while the top-ranked IBM Sequoia achieved 16.32 and Fujitsu K computer achieved 10.51 [1].

On the physical process level side, the number of transistors on a single chip is in the billions (e.g. Intel 48-core SCC processor includes more than 13 billion transistors [2]), and the physical process level is going to become even higher. The improvement of process level leads to more available on-chip resources, as well as more power and area saving for the same resources.

Network-on-Chip (NoC) has a significant impact on the scalability and performance of multi/many-core processors

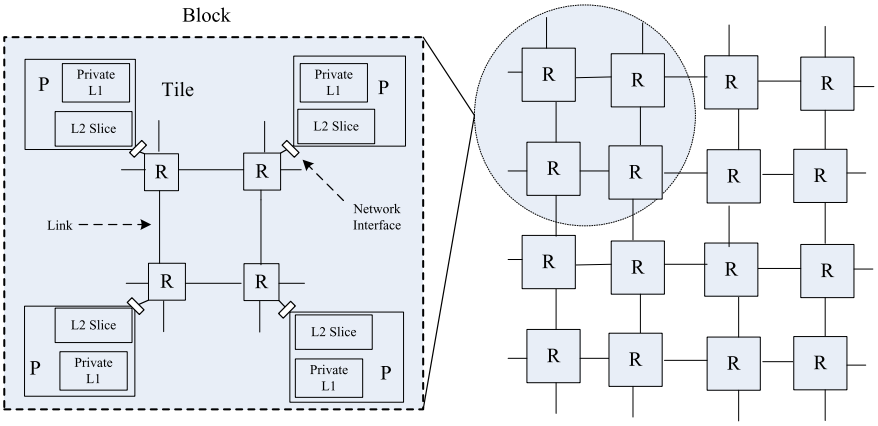


Fig. 1 Mainstream tiled multi-core architecture

[3–5]. Most of the existing single-chip multiprocessors use classical topologies such as *mesh*, *torus*, etc. However, as one of the current processor technology research hotspots and development trends, many-core processor has evolved to include dozens of cores. To collaboratively performing a same program, more cores need to communicate with each other,while more message traffic are needed for synchronization and cache coherence between the cores. As a result, communication problem becomes more acute, which in turn presents higher requirements for corresponding interconnection networks.

Therefore, traditional topologies need to be re-examined. By taking advantage of the increasing process level and adapting to the characteristics of multi/many-core processors, a moderately aggressive interconnection topology and corresponding link design strategy may be a promising solution to alleviate current communication problems.

The contributions of this work are as follows:

First, in the analysis of topologies, a metric that evaluates performance and cost is presented and is shown to outperform a former metric while being consistent with the EDP (production of latency and energy) obtained from the closed-loop network simulation in detail.

Second, an algorithm for computing path diversity is presented, and the concept of link entropy and corresponding calculation method are proposed.

Third, adapting to massive data communication that arised from applications and multi/many-core scale, the design of two torus-like topologies, *xtorus* and *xxtorus*, are discussed. *Xtorus* is shown to inherit good symmetry propriety from *torus*, further reducing average latency, increasing throughput, and also keeping some degree of path-diversity.

Fourth, compared with *xtorus*, *xxtorus* is shown with diminishing marginal benefit, which can be regarded as the upper-bound expansion of *xtorus*.

Finally, based on the entropy difference for links in the topology, a method for heterogeneous link design is presented. The method enables designers to trade off between delay, power and area according to a concrete integrated circuit design scene.

The rest of the paper is organized into four sections. In Sect. 2, related work is reviewed. Section 3 presents *xtorus* topology, and compares it with mainstream classical topologies in terms of five aspects: network diameter, average delay, throughput, path diversity, and difficulty of physical realization. In Sect. 4, from the perspective of link entropy, a method for entropy-based heterogeneous link design is proposed. And finally, some research conclusions are presented.

2 Review of related work

The design of on-chip and on-board interconnection networks has an important influence on efficiency, router energy consumption, and scalability of system.

In 2000, Hemani et al. [6–8] proposed an on-chip interconnection network as a new design paradigm for communication within a single chip. As shown in Fig. 1, the interconnection structure of mainstream many-core processors consists of three basic components: (1) Network Interfaces: through which processor cores are connected to the NoC. (2) Routers: which are responsible for selecting the transmission path from the source node to destination node and forwarding flits. (3) Links: which connect the routers and provide communication bandwidth.

NoC-based system-on-chip (SoC) implements a relative separation between computing and communications. As a subsystem, NoC is responsible for high-speed communication between the processor cores. Therefore, it becomes a research hotspot for current CMPs (Chip Multiprocessors) and MPSoCs (Multiprocessor System-on-Chip). There have been several novel NoC architectures, such as TRIPS [9], Xpipes [10], QNoC [11], Cell [12, 13].

In recent years, on-chip network research has mostly focused on low-power design, router design, implementation, testing methods and fault tolerance mechanisms. In terms of research on topology, there has been relatively less [14].

In 2007, Intel released TeraFLOPS [15, 16] which contains 80 processor cores within a single chip. TeraFLOPS uses a 65 nm process, and is only 275 mm^2. The processing engine (PE) in each tile is connected with a router through the network interface, and all routers form an 8×10 *mesh* topology.

In 2008, Tilera introduced the 64-core chip TILE64, which is oriented for high-performance embedded applications [17]. In TILE64, 64 tiles share memory, and the processor cores communicate with each other through an 8×8 *mesh* topology.

In 2010, Intel presented the 48-core SCC (Single Chip Cloud Computer) [2]. SCC uses a 6×4 *mesh* tiled interconnection structure, and each tile includes two processor cores.

It should be noted that the Intel 80-core TeraFLOPS and 48-core SCC both use *mesh* topology as an interconnection structure. However, they are still research chips and are not included in any production roadmap [2, 18]. Therefore, which topology will be ultimately used for future production-level many-core chips remains a question.

Torus is a good interconnection structure in terms of average latency and symmetry [19]. Based on it and taking advantage of high process level, it is possible to design a new topology to meet high communication performance requirements that many-core processors present, and to suit a great variety of traffic patterns. As a result, this paper proposes a novel and torus-like topology for massive data communication, and a corresponding heterogeneous link design strategy is also presented.

John Kim et al. [20] proposed a flattened butterfly topology, which uses more links than classical topologies under the same network scale. However, since multiple processor cores are linked to the same router (the so-called concentration) and the number of channels on a link is reduced, average delay, throughput and power consumption are improved to some extent. For *xtorus*, it is also possible to use the concentration technology, which can scale up system size with a fixed number of routers. Due to space constraints, it will not be discussed further in this paper.

One of the basic differences between *xtorus* and flattened butterfly is that the former is characterized by having fewer but wider links, while the latter is characterized by having more but narrower ones.

Zhu Xiaojing [14] proposed the *xmesh* topology, the performance of which has been improved to some extent compared with *mesh*. But in terms of hot-spot workload, there is a great fluctuation on average latency, depending on whether the hot-spots are far away from the network center or the diagonals.

The design of fully customized communication architecture, for a given application, is addressed in literature [21]. Ogras and Marculescu presented a method for generating hybrid topologies by the insertion of long-range links into regular topologies [4]. These works focused on the algorithms for topology generation or mapping. But the aim of this paper is to design a single topology adapting to different communication patterns.

Moreover, L. Cheng et al. [22] suggested that due to different delay and bandwidth requirements of cache coherence protocol messages on a chip multiprocessor, it is necessary to design heterogeneous links. At the cost of increasing hardware complexity, the aim is to make some improvements on power consumption and performance. In the scheme, each link has three configuration sets, which are oriented for latency, bandwidth, and power consumption respectively. However, for the proposed strategy in this paper, there is only one configuration set for each link, while different links may have different configurations according to their entropies.

Many network simulators have been developed, such as OCCN [23] and Garnet [24] (which has been integrated into GEM5 [25]). Also, Hanjoon Kim [26] discussed open-loop and closed-loop simulations for on-chip networks. However, while previous research has presented accurate features, simulation in detail requires one to have good understanding of the simulator itself. Additional programming work may be required if the target network architecture is not supported. This raises the question of whether a rough evaluation can be done in an efficient way before simulation.

Dingxing Wang and Guoliang Chen [27] raised five technical requirements. However, these requirements are descriptive rather than quantitative.

Although some analytic models such as Group-Theoretic Model [28] are quantitative and concise, difficulties remain in terms of lengthy applying the model that using multiple graph theories.

Dally [29] addressed three key metrics of performance: throughput, latency, and path diversity. However, the work of Dally and previous researchers differs from the present study in that it fails to provide a fast analysis, which is needed, despite some details being abstracted.

3 The proposed asymmetrical topologies

Using 16 routers scale as an example, classical *mesh* and *torus* topologies are shown in Figs. 2(a) and 2(b). Note, the *xmesh* shown in Fig. 2(c) is an extended *mesh*.

Proposed *xtorus* and *xxtorus* topologies are shown in Figs. 2(d) and 2(e). Compared with *torus*, *xtorus* increases links in both principal diagonal and counter-diagonal directions, but unlike *xxtorus*, it doesn't connect the upper left and lower right vertices of the principal diagonal as well as the upper right and lower left vertices of the counter-diagonal.

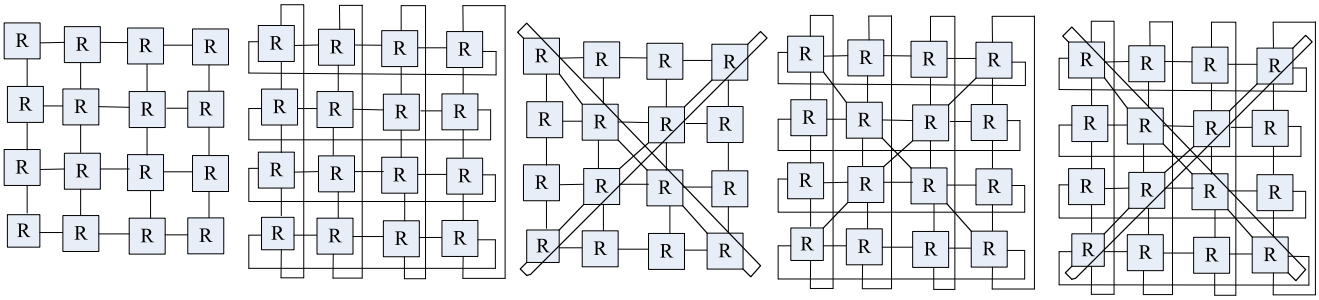


Fig. 2 *mesh* and *torus* topologies and their extensions

In the following, firstly, an indicator (as a general metric) for the pre-simulation evaluation of network performance and cost is presented. Then, structural properties of the topologies are analysed, and for further verification, performance evaluation is also conducted using an execution-driven simulator.

3.1 Metric for evaluating performance and cost

Degree and diameter are two important parameters for interconnection network structures. Degree is related to system price; generally, greater the degree, greater the complexity of the processor interface, and higher the cost; cost also increases with the number of system links. Degree and diameter influence each other, while both affect system scalability and fault tolerance capacity. Usually, greater the degree of nodes, shorter the diameter of network, and higher the system fault tolerance capacity.

Accordingly, for a given network, a metric that measures performance to cost ratio (abbreviated as P_c^I) is the reciprocal of the product of degree and diameter [11], as shown in formula (1). Usually, larger the metric, better the performance to cost ratio.

$$P_c^I(N, Topo) = 1/(L * M) \tag{1}$$

Here, $L = \max\{d_{ij}, 0 \le i \le N-1, 0 \le j \le N-1\}$, $M = \max\{\lambda_k, 0 \le k \le N-1\}$, d_{ij} is the length of the shortest path between any node pair (i, j), N is the number of nodes, and λ_k is the degree of node k.

From Table 1, the metric is shown with coarse property. On one hand, the metrics for different interconnection structures are close to zero, the trend of which becomes more evident as N increases. As a result, the metric poses difficulties in terms of differentiating variable interconnection structures from the perspective of performance and cost. On other hand, own to L and M are the maximum values, that the metrics are not conducive for comparing the scalability of different topologies either.

In this study a new metric is presented, as shown in formula (2).

Table 1 Metric values for the network topologies when $N = 16$ and 64

Topology	*mesh*	*torus*	*xtorus*	*xxtorus*
$P_c^I(N=16)$	0.042	0.083	0.056	0.056
$P_c^{II}(N=16)$	7.500	8.000	8.237	8.280
$P_c^I(N=64)$	0.018	0.031	0.024	0.024
$P_c^{II}(N=64)$	18.375	16.000	15.400	15.260

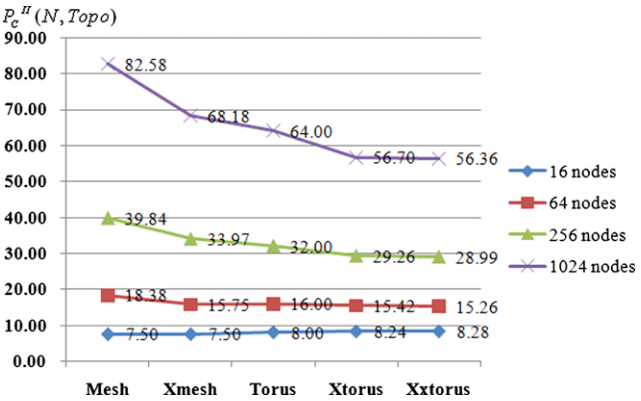


Fig. 3 $P_c^{II}(N, Topo)$ values for different N and topology types

$$P_c^{II}(t, N, Topo) = L(t, N, Topo) * M(t, N, Topo) \tag{2}$$

Here, $L(t, N, Topo) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} p_{ij}(t) d_{ij}(t)$ is the average number of hops for any node pair, where $p_{ij}(t)$ is the probability that communication is carried out between any node pair (i, j) at t moment and $d_{ij}(t)$ is the length of the shortest path between any node pair (i, j) at t moment. $M(t, N, Topo) = \frac{1}{N} \sum_{k=0}^{N-1} \lambda_k$ is the average node degree.

The reason that $d_{ij}(t)$ varies with t, is that there are always multi-source situations during the running of real applications. That is, the shortest paths of different node pairs may have routing conflicts, for which the length of the real-time shortest path may be greater than that of the static case.

Figure 3 shows the scalability of the topologies as the metric varies with N. Compared with $P_c^I(N, Topo)$,

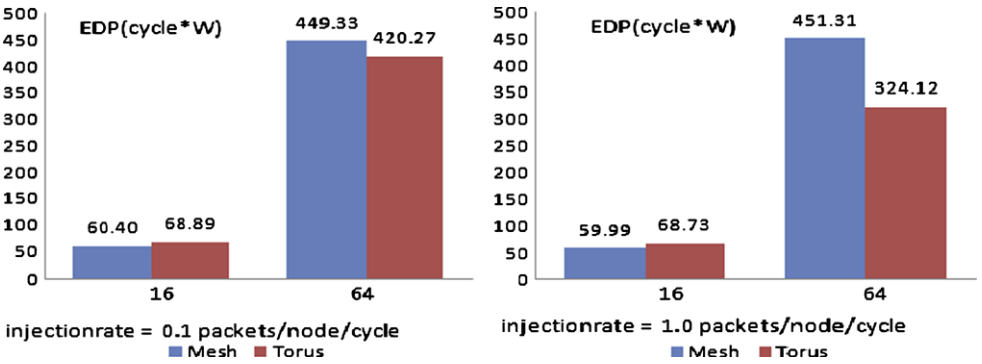


Fig. 4 EDP for *mesh* and *torus* under different system scales, 16 and 64. The *left sub-graph* represents EDP when inject rate equals 0.1 packets/node/cycle, while the *right sub-graph* illustrates EDP when inject rate equals 1.0 packets/node/cycle

Table 2 Experiment environment

Parameters	Values
System scale	16 and 64
Router pipeline	BW + RC, VA, SA, ST, LT

$P_c^{II}(N, Topo)$ is superior in terms of differentiating topologies, especially at large scales.

To allow for a rough yet efficient estimation, the present study utilizes a random flow pattern. That is, node pairs within the entire system have the same probability for communication. For instance, in this case, the metric that evaluates performance and cost of two-dimensional *mesh* can be calculated using formula (3). Metric values when N equals 16 and 64 for the typical network topologies are listed in Table 1.

$$P_c^{II}(t, N, Mesh) = 4 * \frac{\sqrt{N}-1}{(\sqrt{N})^3} \sum_{i_d, j_d, i_s, j_s=0}^{\sqrt{N}-1} \left(|i_s - i_d| + |j_s - j_d|\right) \tag{3}$$

EDP [16] is obtained by the production of latency and energy. Note that a decrease in energy may lead to an increase in latency. For on-chip or on-board networks, lower EDP is preferred, which indicates it is better in terms of performance and energy consumption. As a main source of energy, static router power consumption is partly determined by the number of ports for each router, which directly depends on the node degree.

Figure 4 indicates that when the system scale is 16, EDP of *torus* is larger than that of *mesh*; when the system scale is 64 the opposite situation occurs. Therefore, compared with Table 1, one can conclude that the metric P_c^{II} is consistent with EDP obtained from the simulation using Garnet in the GEM5 simulator (the experiment environment is shown in Table 2).

3.2 Structural properties

3.2.1 Network diameter

Network diameter refers to the largest, minimal hop count over all pairs of terminal nodes in the network. Reducing network diameter is helpful in reducing communication overhead. A network is a graph $G = (V(G), E(G))$, while $V(G)$ and $E(G)$ are corresponding vertex set and edge set respectively.

Given $P_G(u, v) = \{p : p \text{ is a simple path between } u \text{ and } v\}$ and $P(G) = \{P_G(u, v) : u, v \in V(G)\}$, the network diameter is $d(G) = \max\{|p| : p \in P \in P(G)\}$.

In a $n \times n$ network, the diameter of *mesh* is $2(n-1)$; for *xmesh*, it is $n-1$. When n is odd, the diameter of *torus* is $n-1$. When n is even, the diameter of *torus* is n. Whether n is odd or even, the network diameter of *xtorus* remains $n-1$. Therefore, compared with *mesh* and *torus*, *xtorus* has the shortest network diameter.

3.2.2 Ideal average latency

For a given topology, assuming there is no congestion in routing, the average routing delay between all nodes in the topology is the so-called ideal average latency. It can be estimated in formula (4), which was proposed by William James Dally et al. [29].

$$T = H * Tr + D/v + L/b \tag{4}$$

Here, H is the average hop number from source node to destination node. Tr is the delay in routing on the router, and the unit is cycle/hop. D is the average distance from the source node to destination node, which usually equals to H, and the unit is hop. v is the wire transmission speed, and the unit is hop/cycle. L is the packet length, and the unit is flit. b is the bandwidth, and the unit is flit/cycle. Generally, different topologies correspond to different H and D values, while Tr is dependent on the routing algorithm and

Table 3 Calculation of the average distance of *xtorus*

	00	01	11
Nodes from which distance equals 1	01, 03, 10, 11, 30	00, 02, 11, 31	00, 01, 10, 12, 21, 22
Nodes from which distance equals 2	02, 12, 13, 20, 21, 22, 31, 33	03, 10, 12, 21, 22, 30, 32	02, 03, 13, 20, 23, 30, 31, 32, 33
Nodes from which distance equals 3	23, 32	13, 20, 23, 33	

the physical implementation of the router. To calculate the average latency, the first step is to calculate H, that is, D.

For a 4×4 network scale, $D_{Mesh} = H_{Mesh} = 2.5$, $D_{Torus} = H_{Torus} = 2$. For corresponding values of *xtorus*, due to its left-right, upper-lower and diagonal symmetry, the sixteen nodes can be divided into three categories:

$$Class1 = \{00, 03, 30, 33\}$$

$$Class2 = \{01, 02, 31, 32, 10, 20, 13, 23\}$$

$$Class3 = \{11, 12, 21, 22\}$$

The number of the nodes is coded by vertical coordinate and horizontal coordinate. Taking node 00, 01, 11 as the representative of the three classes, the distances between nodes are shown in Table 3.

In terms of routing, in order to be adaptive to different communication patterns and for all possible workload patterns, it is expected that the distances between nodes should show a good balance. That is, the number of equivalence classes and the difference between equivalence classes should both be as small as possible.

$$D(00) = 1 * 5 + 2 * 8 + 3 * 2 = 27$$

$$D(01) = 1 * 4 + 2 * 7 + 3 * 4 = 30$$

$$D(11) = 1 * 6 + 2 * 9 = 24$$

$$D_{xtorus} = \big[D(00) * 4 + D(01) * 8 + D(11) * 4\big]/256 = 1.734$$

If each router uses a typical 5-stage pipeline ($BW + RC, VA, SA, ST, LT$) [9], and the interval time between an adjacent pipeline stage is 1 *cycle*, then the processing time of the router is 4 *cycles*, that is, $Tr = 4$ *cycles*/*hop*. It is also assumed that $v = 1$ *flit*/*cycle*, $L = 2$ *flits* and $b = 1$ *flit*/*cycle*.

Therefore, the ideal average latencies of these topologies are as follows:

$$Tmesh = 2.54 + 2.5/1 + 2/1 = 14.5$$

$$Ttorus = 24 + 2/1 + 2/1 = 12$$

$$Txmesh = 1.8754 + 1.875/1 + 2/1 = 11.375$$

$$Txtorus = 1.7344 + 1.734/1 + 2/1 = 10.67$$

$$Txxtorus = 1.6564 + 1.656/1 + 2/1 = 10.28$$

At 16-node scale, the ideal average latency of *xtorus* is less than that of *xmesh*, *torus* and *mesh* by 6.2 %, 11.08 % and 26.4 % respectively, while the ideal average latency of *xxtorus* is less than that of *xtorus* by 3.65 %.

Moreover, at 64-node scale, the ideal average latency of *xtorus* is less than that of *xmesh*, *torus* and *mesh* by 10.6 %, 11.8 % and 31.33 % respectively, while the ideal average latency of *xxtorus* is slightly less than that of *xtorus* by 2.32 %.

3.2.3 Ideal throughput

For a given topology, ideal throughput is the maximum network throughput in a perfect flow control and routing mechanism. William James Dally, et al. [29] gave a formula as follows for calculating the ideal throughput (TH). Bc is the number of channels required to divide the whole network into two equal halves; b is the width of each channel; N is the number of nodes.

$$TH \leq 2b * Bc/N \tag{5}$$

Taking a 4×4 node scale for example, for *mesh*, it takes four links to divide it into two halves. And each link is bidirectional, so $Bc = 8$. In the same way, Bc for *torus*, *xtorus* and *xtorus* equals to 16, 16 and 20 respectively.

Therefore,

$$THmesh \leq b, \qquad THtorus \leq 2b,$$

$$THxmesh \leq 2b, \qquad THxtorus \leq 2.5b.$$

The ideal throughput of *xtorus* increases by 150 % compared with that of *mesh*, and increases by 25 % compared with that of *torus* and *xmesh*.

3.2.4 Path diversity

If there is more than one shortest path between most of the node pairs, then the topology is referred to as possessing path diversity [29]. With better path diversity, it is possible to select between more multiple shortest paths, which can effectively reduce network congestion and enhance network fault tolerance.

For any node pair (u, v), the path diversity can be expressed as $|P_G(u, v)|$. And the path diversity of graph can summed up as $\sum_{u,v \in V(G)} w_{(u,v)} * |P_G(u, v)|$.

Communication patterns can be shown as a $N \times N$ matrix Λ. Each element $w_{(u,v)}$ in Λ refers to the probability of communication corresponding to (u, v) at a given moment.

Dijkstra [30], Goldberg et al. [31] proposed methods for computing the shortest path between two nodes. As shown in Fig. 5, this paper presents an algorithm for calculating path diversity between any node pair for a connected graph.

```
Input: Edge[n][n] as the adjacency matrix for a graph with n nodes
Output: RoadNum[n][n] the element of which is the number of the shortest paths between any two nodes; LinkRoadNum[n][n] that records the number of the shortest path through any link
Begin
// RoadNum[i][j] is initialized as Edge[i][j]
// LinkRoadNum[i][j] is initialized as 0
for(int i = 0; i < n; i++)
  for (int j = 0; j < n; j++){
     RoadNum[i][j] = Edge[i][j];
     LinkRoadNum[i][j] = 0;
   }
for( i = 0; i < n; i++)
     {
     temp = Find(i, {i}); // InputSet initially only has the source node
     while( temp != ϕ )
     temp = Find(i, temp)
     }
Find(Source, InputSet){
//  Source is the source node
//  InputSet is the input node set
//  OutputSet: its members are the adjacent nodes (of which the serial number is greater than the input nodes) of input nodes in InputSet
     OutputSet = ϕ ;
     for any i∈InputSet
        for ( j = i + 1; j < n; j++){
          if (Edge[i][j] == 1)
          {
// Here set merger operation does not merge the same elements, but counts their numbers
          OutputSet = OutputSet ∪ { j };
// calculate the number of the shortest path through any link
          LinkRoadNum[i][j]++;
          }
     }
     for any k ∈ OutputSet
     RoadNum[Source][k] = the number of k in OutputSet;
     return OutputSet;
}
End
```

Fig. 5 Pseudocode of the path diversity algorithm for any node pair in an undirected connected graph with equal weights. It is also an algorithm for calculating entropy of any link

Note that the algorithm also presents the number of the shortest path through each link, which will be used in Sect. 4 for the entropy-based design of heterogeneous links.

To illustrate, for *mesh*, the process of calculating the path diversity and link entropy is shown in Fig. 6.

Compared with *mesh*, *xmesh* loses some path diversity. Therefore, in terms of this aspect *xmesh* has some relative disadvantage [7]. As shown in Table 4, *xtorus* has in average more than two shortest paths for any node pair, which is more than that of *xmesh*, and thus has better multi-routing and fault tolerance capability.

3.2.5 Physical implementation difficulty

The physical implementation difficulty of different topologies depends on the symmetry of the structure, the number of dimensions, the number of long edges, the number of ports of each router, and the complexity of routing algorithms.

For *mesh*, the links in the structure are short; the number of router ports is no more than 4; and the routing algorithm is simple (such as XY). Based on all of those, the difficulty of physical realization is the lowest.

For *xmesh*, most of the links in the structure are short except for the two long diagonals. The number of ports of each diagonal router is 6, while the number of ports of other routers is no more than 4. Its routing algorithm [14] is more complex than dimension-ordered routing.

For *torus*, the routing algorithm is relatively simpler. Although the number of links is more than that of *mesh* and *xmesh*, the number of ports of each router is 4. Some of the links is longer than that of *mesh* and *xmesh*.

For *xtorus*, every router except the one at the corners on each diagonal has 6 ports, and it also has long links as *torus*. But due to the higher processing level of VLSI, especially the even richer resources in manycore processors and optimized hardware layout, the physical realization problem would be effectively alleviated.

For *xxtorus*, the routers on each diagonal all have 6 ports, and more importantly it has two long diagonal edges, which would bring greater difficulty of wiring, while the long wires would cause long delay and signal integrity problem.

Meanwhile, *xtorus* is symmetrical from top to bottom, from left to right. And in relation to the diagonals, it also has good path diversity. In terms of number of links shown in Table 5, with respect to *mesh*, *xmesh* and *torus*, *xtorus* has no increase in order of magnitude, while the number of links of Flattened Butterfly is of a higher order of magnitude. Overall, the complexity of the physical realization of *xtorus* increases slightly as compared to *torus* and *xmesh*, but is of the same order of magnitude.

3.3 Performance evaluation

Performance evaluations were completed by using the Garnet in the execution-driven simulator GEM5 [24, 25]. The simulation environment is shown in Table 6. Each router has five pipeline stages. Note that if the terminal node has to generate a new packet, it computes the destination for the new packet based on the synthetic traffic type (- -synthetic).

0 (Uniform Random): send to a destination directory chosen at random from the available directories.

1 (Tornado): send to the destination directory half-way across X dimension.

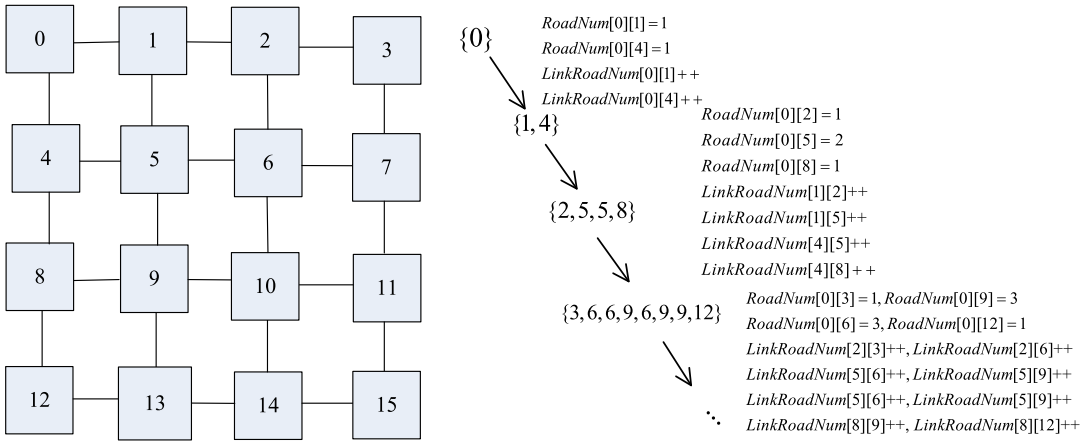


Fig. 6 The process of calculating the path diversity and the link entropy for mesh topology

Table 4 Average path diversity at 16-node scale

mesh	*torus*	*xmesh*	*xtorus*	*xxtorus*
2.96	4.06	1.40	2.08	1.75

Table 5 Number of links corresponding to different topologies

Pt2Pt	$n^2(n^2-1)/2$
mesh	$2n(n-1)$
torus	$2n^2$
xmesh	$2n^2$
xtorus	$2n^2+2(n-1)$
xxtorus	$2n^2+2n$
Flattened Butterfly	$n^2(n-1)$

2 (Bit-Complement): send to the destination directory whose location is the bit-complement of the source terminal node location.

The aim was to evaluate whether the average latency for *xtorus* can be significantly reduced compared with that of *mesh* and *torus*, and whether *xtorus* is able to adapt to a variety of communication patterns among different processor cores.

For simulating massive data communication, a range of injection rates were configured, while 1 packet/node/cycle implied intensive communication.

For each topology, the average latency are shown in Figs. 7, 8 and 9, corresponding to the injection rate of 0.01, 0.1, 1.0 (packet/node/cycle) respectively. For each figure, the sub-graph (a), (b), (c) denotes the average latency of such topologies corresponding to traffic patterns 0, 1 and 2.

By analyzing these figures and Table 7, the following conclusions can be drawn.

Table 6 Experiment environment

Parameters	Values
System scale	16 and 64
Router pipeline	BW + RC,VA,SA,ST,LT
Routing scheme	To choose the route with minimum number of link traversals, rinks can be given equal weights.
Simulation time	100000 cycles
Traffic patterns	0 (Uniform Random); 1 (Tornado); 2 (Bit-Complement).

Table 7 The power consumption (Unit: Watt) of *xtorus* at different injection rates and system scales

Injection rate	0.01	0.10	1.00
16 routers	3.94	4.06	4.07
64 routers	15.34	15.44	15.44

– For *xtorus* topology, when the system scale, injection rate and routing algorithm are kept constant, the average latency gradually increases for traffic patterns 1, 2, and 0. But for *mesh* topology, the average delay ascends in sequence of traffic patterns 1, 0, and 2. It reflects the differences among different topologies in terms of their adaptability to different traffic patterns.
– *Mesh* has the highest latency for traffic pattern 2 among all the traffic patterns. In this case, the advantage of *xtorus* to *mesh* is most obvious.
– If the sub-graphs in Figs. 7, 8, and 9 are compared vertically, it can be seen that the average latency is not sensitive to the injection rate. As the injection rate increases,

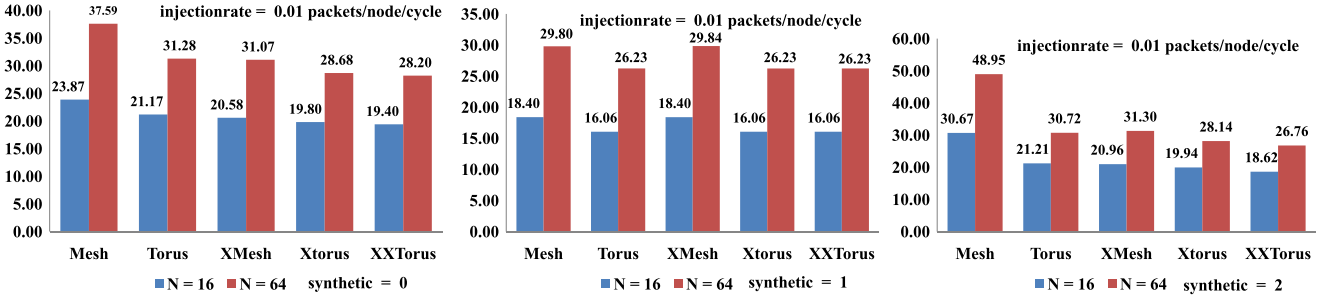


Fig. 7 Case that the injection rate is 0.01 packet/node/cycle

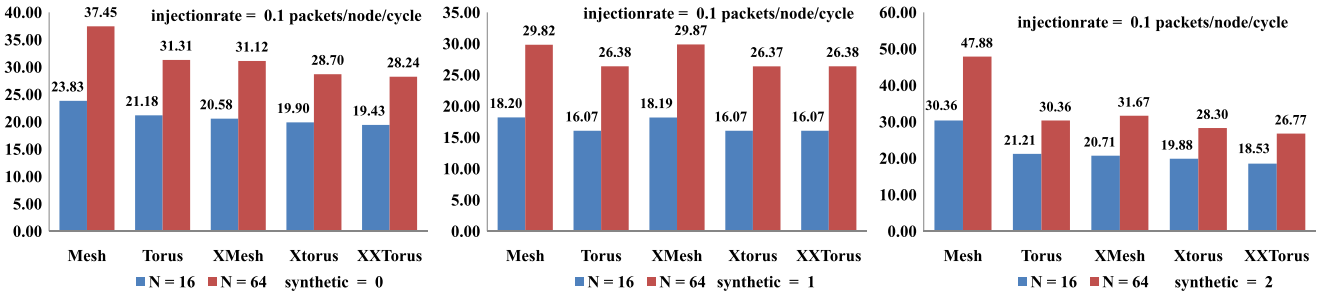


Fig. 8 Case that the injection rate is 0.1 packet/node/cycle

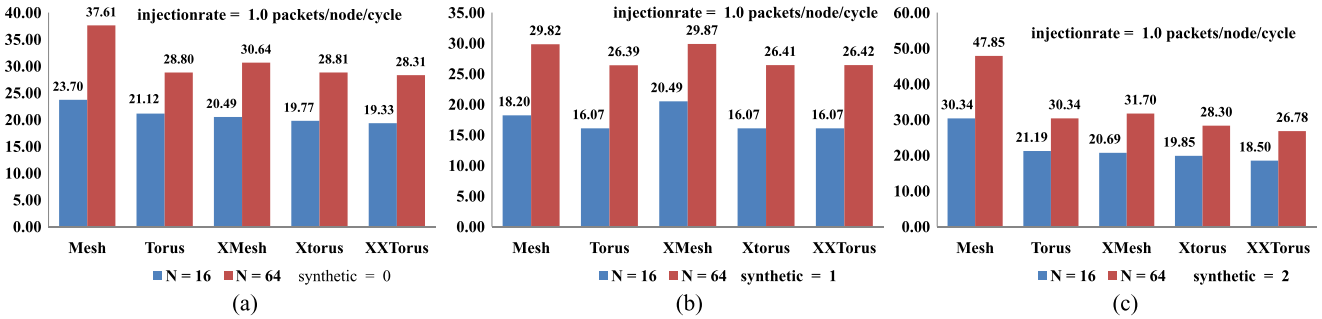


Fig. 9 Case that the injection rate is 1.0 packet/node/cycle

there is only a very small amount of change. Even when the injection rate increases 10-fold, the corresponding change of the average latency is still within 2 %.

- As shown in Table 7, for *xtorus* topology at 16-node and 64-node scale, power consumption of the routers is initially sensitive to the injection rate. However, after the injection rate increases to a certain point (i.e. 0.1 packets/node/cycle), there is almost no further increase in power consumption of the routers.
- For different traffic patterns, the performance swing for *xtorus* is less than that for *mesh*, showing a well-balanced feature.
- Through a combination of simulation-based evaluations regarding performance and power consumption, it can be concluded that relative to *xtorus*, *xxtorus* has more power consumption and two additional long diagonals, but the performance improvement is small. Therefore, compared with *xtorus*, it is concluded that *xxtorus* is shown with diminishing marginal benefit, which can be regarded as the upper-bound expansion of *xtorus*.

The logical topology is implemented in the physical integrated circuits and is composed of physical links, the design of which is presented in the next section.

4 Entropy-based design of heterogeneous links

Depending on the symmetry of the topology, different links of a topology may show different level of hotness. That is,

there are more paths are through some links, while there are fewer paths are through others.

Assuming the routing algorithm is R,

$$R = \{(u, v) \to p' : u, v \in V(G),\ p' \in P_G(u, v)\},$$

then the result of routing is

$$R_{I_m} = \{p' : \exists u, v \in V(G),\ p' \in P_G(u, v)\}.$$

$\forall p \in P_G \in P(G)$ and $\forall l \in E(G)$, in terms of the number of paths through a link (denoted by l), there are two categories.

First, with expression (6), when multiple paths between a node pair go through a given link, they are counted respectively, and are regarded as the potential number of paths through the given link.

Second, with expression (7), when multiple paths between a node pair go through a given link, they are counted no more than one time (if and only if is on the path selected by the routing algorithm), and are regarded as the actual number of paths through the given link.

Both of the above are referred to as the entropy of a link l.

$$\sum_{P \in P(G)} \sum_{p \in P} (l \in p) \tag{6}$$

$l \in p$ indicates link l is on path p.

$$\sum_{P \in P(G)} \sum_{p \in P} \big((l \in p) \wedge (p \in R_{I_m})\big) \tag{7}$$

Since R_{I_m} denotes the path set determined by the routing algorithm for any node pair, $p \in R_{I_m}$ indicates that p is selected by the routing algorithm.

As shown in Fig. 2(a), for uniform communication load and the shortest path routing, the number of paths through the inner ring is 2 to 4 times the number of paths through the outer ring, so it is more likely for the inner ring of the *mesh* structure to be congested. As shown in Fig. 2(b), the entropy of each link in the *torus* structure remains the same (16 and 1588 at 16-node and 64-node scale respectively), hence for uniform load and the shortest path routing, the possibility of congestion is lower. In Fig. 2(c), for *mesh*, there may be larger traffic through its two diagonals. Compared with *torus*, as shown in Fig. 2(d), *xtorus* has two additional diagonals, but no greater number of paths is travelled through.

Avoiding or reducing congestion is helpful for reducing the communication latency and thus improving performance. As an indicator, the number of paths through a link is closely related to network congestion. As the indicator becomes larger, potential congestions increases.

Keeping in mind the reduction of network cost and congestion, this study has designed the bandwidth of each link in the structure as a function of its entropy.

The latency of the wire is related to the production of R and C, which are the resistance and capacitance of the wire respectively. The resistance per unit length is (approximately) inversely proportional to the width of the wire [32]. The capacitance per unit length is partly inversely proportional to the spacing s of the wire, and is partly proportional to the width w of the wire [22].

If the amount of coverage metal (e.g. copper) on each wire or at least one of the w and s increases, it is possible to reduce the production value of R and C, thereby reducing the latency of the link. However, if the bit width of each link is kept the same, a single link will occupy more area.

For a long link, it is common to use some repeaters to connect multiple short segments. If more as well as larger repeaters are used, the latency is generally smaller, but the power consumption is generally greater. By utilizing the non-uniform electrical properties of different links, it is possible to trade off between delay, power and area. That is, for each link, according to the specific IC physical circumstances, different wire widths, wire spacing as well as the number and size of repeaters could be configured in a flexible manner. Therefore, it would allow for a certain degree of freedom in the IC design of the links.

Based on link entropy, heterogeneous link design can be conducted. That is, for a link with higher entropy, it is necessary to consider using more coverage metal, or larger w and s, as well as more and larger repeaters. For a link with lower entropy, it is necessary to consider using less coverage metal, or smaller w and s, as well as fewer and smaller repeaters. That is, actual design should be chosen according to the specific situation.

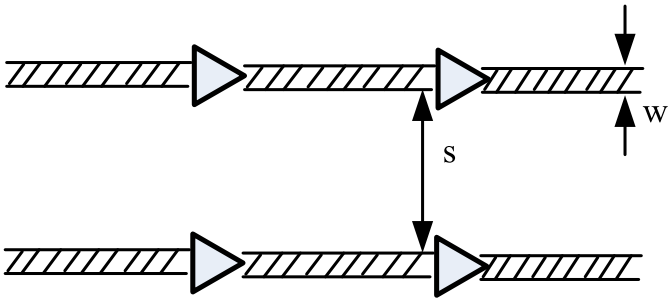
Heterogeneous and homogeneous designs are evaluated using Scheme 1 and Scheme 2 respectively. The experiment environment is same as that of Sect. 3.3 as shown in Table 6.

Scheme 1: As shown in Fig. 10, it employs entropy-based heterogeneous links. In this scheme, when the entropy is larger, the link is designed as the first type, that is, the w and s of the link are made larger, while more and larger repeaters are used. However, when the entropy is smaller, the link is designed as the second type. That is, the w and s of the link are made smaller, while fewer and smaller repeaters are used.

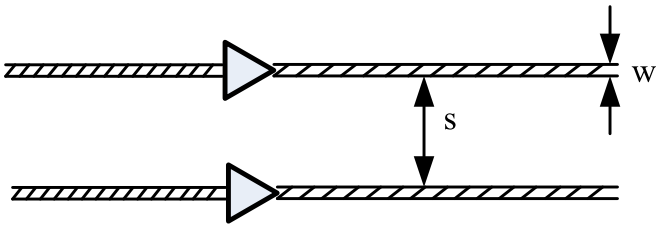
Scheme 2: All links are of the same structure as the first type, that is, both w and s are smaller, and repeaters for each link are fewer and smaller.

One can see in Fig. 11 (for space constraints, only data for cases when the injection rate is 1.0 packets/node/cycle and system scale is 64 is given) that compared with homogeneous links, when area and power consumption are kept the same, there will be an approximate $3 \sim 11$ % performance improvement when using heterogeneous links. The rate of change depends on traffic patterns. For traffic pattern 0 (Uniform Random), there is a roughly 2.8 percent benefit, while for traffic pattern 2 (Bit-complement), there is about 10.7 % benefit.

Fig. 10 Heterogeneous links



(a) Link type1: Larger width *w* and spacing *s*, more and larger repeaters



(b) Link type2: Smaller width *w* and spacing *s*, fewer and smaller repeaters

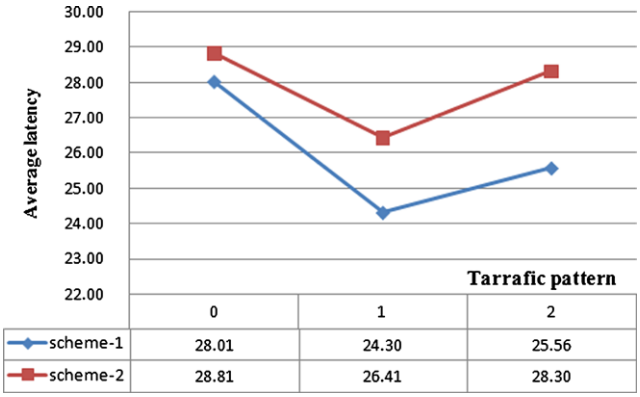


Fig. 11 Average latency vs. traffic patterns under different link schemes

If performance remains the same, there will be a $2 \sim 6$ % improvement in area and power consumption savings when using heterogeneous links. Both application traffic patterns and physical process levels affected the rate of change.

5 Conclusions

To adapt to the need for massive data communication and the increase in the number of on chip/board processing units, the present work has taken advantage of increased area and lower power in order to provide a new topology called *xtorus* and an entropy-based method for heterogeneous link design. *Xtorus* not only inherits good symmetry from *torus*, but also further reduces the average latency and increases throughput. Moreover, it also keeps some degree of path-diversity. Finally, an entropy-based heterogeneous link methodology allows designers to trade off between delay, power consumption and area in different concrete integrated circuit design cases.

Acknowledgements This work is supported by the State Key Laboratory of Software Development Environment (SKLSDE-2012ZX-07); Beijing Natural Science Foundation (4122042); the National Natural Science Foundation of China (60973008 and 61232009); the Doctoral Fund of Ministry of Education of China (20101102110018); the Hi-Tech Research and Development Program (863) of China (2011AA01A205). Part of this work is the refinement and extension of our work in 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum. The authors would like to express sincere gratitude to the Supercomputing Research Center of Beihang University and Computer Network Information Center, Chinese Academy of Sciences, for providing quality experiment facilities. Appreciation should also be expressed to the anonymous referees for their detailed comments and valuable suggestions.

References

1. Site, T.S.: Top500 list–June 2012 (1-100). Top500 Supercomputing Site [Website], http://www.top500.org/list/2012/06/100 (2012)
2. Mattson, T.G., Riepen, M., Lehnig, T., Brett, P., Haas, W., Kennedy, P., Howard, J., Vangal, S., Borkar, N., Ruhl, G., et al.: The 48-core scc processor: the programmer's view. In: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–11. IEEE Computer Society, Washington (2010)

3. Bartic, T.A., Mignolet, J.Y., Nollet, V., Marescaux, T., Verkest, D., Vernalde, S., Lauwereins, R.: Highly scalable network on chip for reconfigurable systems. In: Proceedings International Symposium on System-on-Chip, 2003, pp. 79–82. IEEE Press, New York (2003)
4. Ogras, U.Y., Marculescu, R.: It's a small world after all: Noc performance optimization via long-range link insertion. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **14**(7), 693–706 (2006)
5. Nickray, M., Dehyadgari, M., Afzali-kusha, A.: Power and delay optimization for network on chip. In: Proceedings of the 2005 European Conference on Circuit Theory and Design, 2005, vol. 3, pp. III–273. IEEE Press, New York (2005)
6. Dally, W.J., Towles, B.: Route packets, not wires: on-chip interconnection networks. In: Proceedings Design Automation Conference, 2001, pp. 684–689. IEEE Press, New York (2001)
7. Hemani, A., Jantsch, A., Kumar, S., Postula, A., Oberg, J., Millberg, M., Lindqvist, D.: Network on chip: an architecture for billion transistor era. In: Proceeding of the IEEE NorChip Conference, pp. 166–173, Citeseer (2000)
8. Benini, L., De Micheli, G.: Powering networks on chips: energy-efficient and reliable interconnect design for socs. In: Proceedings of the 14th International Symposium on Systems Synthesis, pp. 33–38. ACM Press, New York (2001)
9. Sankaralingam, K., Nagarajan, R., McDonald, R., Desikan, R., Drolia, S., Govindan, M.S., Gratz, P., Gulati, D., Hanson, H., Kim, C., et al.: Distributed microarchitectural protocols in the trips prototype processor. In: 39th Annual IEEE/ACM International Symposium on Microarchitecture, 2006, MICRO-39, pp. 480–491. IEEE Press, New York (2006)
10. Dall'Osso, M., Biccari, G., Giovannini, L., Bertozzi, D., Benini, L.: Xpipes: a latency insensitive parameterized network-on-chip architecture for multiprocessor socs. In: Proceedings 21st International Conference on Computer Design, 2003, pp. 536–539. IEEE Press, New York (2003)
11. Bolotin, E., Cidon, I., Ginosar, R., Kolodny, A.: Qnoc:Qos architecture and design process for network on chip. J. Syst. Archit. **50**(2), 105–128 (2004)
12. Hofstee, H.P.: Power efficient processor architecture and the cell processor. In: 11th International Symposium on High-Performance Computer Architecture, 2005, HPCA-11, pp. 258–262. IEEE Press, New York (2005)
13. Gschwind, M., D'Amora, B.D., O'Brien, J.K., O'Brien, K., Eichenberger, A.E., Wu, P.: Cell broadband engine enabling density computing for data-rich environments. In: Proceedings of the Annual International Symposium on Computer Architecture (2006)
14. Zhu, X.J., Hu, W.W., Ma, K., Zhang, L.B.: Xmesh: a mesh-like topology for network on chip. J. Softw. **18**(9), 2194–2204 (2007)
15. Hoskote, Y., Vangal, S., Singh, A., Borkar, N., Borkar, S.: A 5-GHz mesh interconnect for a teraflops processor. IEEE MICRO **27**(5), 51–61 (2007)
16. Vangal, S.R., Howard, J., Ruhl, G., Dighe, S., Wilson, H., Tschanz, J., Finan, D., Singh, A., Jacob, T., Jain, S., et al.: An 80-tile sub-100-W teraflops processor in 65-nm CMOS. IEEE J. Solid-State Circuits **43**(1), 29–41 (2008)
17. Wentzlaff, D., Griffin, P., Hoffmann, H., Bao, L., Edwards, B., Ramey, C., Mattina, M., Miao, C.C., Brown, J.F., Agarwal, A.: On-chip interconnection architecture of the tile processor. IEEE MICRO **27**(5), 15–31 (2007)
18. Mattson, T., van der Wijngaart, R.: Rcce: a small library for many-core communication. Intel Corp. **May** (2010)
19. Wei, W., Lin, Q., Guangwen, Y., Zhizhong, T.: Performance analysis of the 2-d networks-on-chip. J. Comput. Res. Devel. **46** (2009)
20. Kim, J., Balfour, J., Dally, W.: Flattened butterfly topology for on-chip networks. In: Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 172–182. IEEE Computer Society, Washington (2007)
21. Ogras, U.Y., Marculescu, R.: Energy-and performance-driven noc communication architecture synthesis using a decomposition approach. In: Proceedings of the Conference on Design, Automation and Test in Europe, vol. 1, pp. 352–357. IEEE Computer Society, Washington (2005)
22. Cheng, L., Muralimanohar, N., Ramani, K., Balasubramonian, R., Carter, J.B.: Interconnect-aware coherence protocols for chip multiprocessors. In: ACM SIGARCH Computer Architecture News, vol. 34, pp. 339–351. IEEE Computer Society, Washington (2006)
23. Coppola, M., Curaba, S., Grammatikakis, M.D., Maruccia, G., Papariello, F.: Occn: a network-on-chip modeling and simulation framework. In: Proceedings of the Conference on Design, Automation and Test in Europe, vol. 3, p. 30174. IEEE Computer Society, Washington (2004)
24. Agarwal, N., Krishna, T., Peh, L.S., Garnet, N.K.J.: A detailed on-chip network model inside a full-system simulator. In: IEEE International Symposium on Performance Analysis of Systems and Software, 2009, ISPASS 2009, pp. 33–42. IEEE Press, New York (2009)
25. Binkert, N., Beckmann, B., Black, G., Reinhardt, S.K., Saidi, A., Basu, A., Hestness, J., Hower, D.R., Krishna, T., Sardashti, S., et al.: The gem5 simulator. Comput. Archit. News **39**(2), 1–7 (2011)
26. Kim, H., Heo, S., Lee, J., Huh, J., Kim, J.: On-chip network evaluation framework. In: 2010 International Conference for High Performance Computing, Networking, Storage and Analysis (SC), p. 10. IEEE Press, New York (2010)
27. Chen, G., Wang, D.-x: Interconnection Network Analysis. Science Press, Marrickville (1990)
28. Akers, S.B., Krishnamurthy, B.: A group-theoretic model for symmetric interconnection networks. IEEE Trans. Comput. **38**(4), 555–566 (1989)
29. Dally, W.J., Towles, B.: Principles and Practices of Interconnection Networks. Morgan Kaufmann, San Mateo (2004)
30. Chen, J.C.: Dijkstras shortest path algorithm. J. Formal. Math. **15** (2003). http://markun.cs.shinshu-u.ac.jp/Mirror/mizar.org/JFM/Vol15/graphsp.html
31. Goldberg, A.V., Harrelson, C.: Computing the shortest path: a search meets graph theory. In: Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 156–165. Society for Industrial and Applied Mathematics, Philadelphia (2005)
32. Ho, R., Mai, K.W., Horowitz, M.A.: The future of wires. Proc. IEEE **89**(4), 490–504 (2001)

Yu-hang Liu Ph.D. candidate, Student membership of China Computer Federation. His main research areas are high performance computer architecture, and parallel processing.

Ming-fa Zhu Ph.D., Professor, Senior membership of China Computer Federation. His main research areas are computer architecture, parallel computing, high performance computer system and network, artificial intelligence.

Li-min Xiao Ph.D., Professor, Senior membership of China Computer Federation. His main research areas are computer architecture, computer system software, high performance computing, virtualization and cloud computing.

Jue Wang Ph.D., Associate professor, membership of China Computer Federation. His main research area is software optimizations for modern architectures including many cores clusters, GPU, etc.