Express Link Placement for NoC-Based Many-Core Platforms

Yunfan Li†, Di Zhu*, Lizhong Chen†
†School of Electrical Engineering and Computer Science, Oregon State University, United States
*Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, United States
Email: †{liyunf, chenliz}@oregonstate.edu, *dizhu@usc.edu

ABSTRACT

With the integration of up to hundreds of cores in recent general-purpose processors that can be used in parallel processing systems, it is critical to design scalable and low-latency networks-on-chip (NoCs) to support various on-chip communications. An effective way to reduce on-chip latency and improve network scalability is to add express links between pairs of non-adjacent routers. However, increasing the number of express links may result in smaller bandwidth per link due to the limited total bisection bandwidth on chip, thus leading to higher serialization latency of packets in the network. Unlike previous works on application-specific designs or on fixed placement of express links, this paper aims at finding effective placement of express links for general-purpose processors considering all the possible placement options. We formulate the problem mathematically and propose an efficient algorithm that utilizes an initial solution generation heuristic and enhanced candidate generator in simulated annealing. Evaluation on 4x4, 8x8 and 16x16 networks using multi-threaded PARSEC benchmarks and various synthetic traffic patterns shows significant reduction of average packet latency over previous works.

1 INTRODUCTION

General-purpose multi-core and many-core processors are becoming increasingly important in large-scale, high-performance parallel computing systems and data centers for numerous scientific, economic, and social computing applications. As more and more cores are integrated in recent chip-multiprocessors (CMPs) [10][15][23], it is crucial to design scalable and high-performance networks-on-chip (NoCs) to support various possible communication patterns among the processing cores. Although NoCs are generally more scalable than buses, the conventional hop-by-hop topologies significantly increase packet latency and tend to consume a large amount of power due to repeated packet forwarding at every hop. It is, thus, critical to explore effective use of express links that allow packets to bypass intermediate routers in the network.

While adding express links may provide significant potential benefits particularly for current and future large many-core chips, the total number of express links at the cross-section between routers is limited by physical wiring resources. Consequently, the width of express links may need to be reduced to satisfy network bisection bandwidth constraints. This leads to increased packet serialization latency, which is the time needed for transmitting the remainder of a packet after receiving its first bit. Therefore, adding express links may not necessarily result in reduction of the overall packet latency.

A couple of works have been conducted in the past on this subject, but opportunities for finding effective express link placement in general-purpose processors are still largely unexplored. For example, many schemes have been proposed for application-specific designs that optimize topologies based on prevailing traffic patterns (e.g., [11][14][20]). These application-specific schemes have limited applicability for general-purpose processors where information about inter-core communication may not be available beforehand or may change frequently at runtime. In these situations, it is needed to place express links in a way that benefits the overall or the average case. Limited research has been conducted in this regard including virtual express topology and physical express topology for general-purpose computing. Either approach has its own advantages and disadvantages (more discussion is in the related work in Section 2).

In this paper, we focus on the optimization of physical express topology and address a key limitation in prior works that add physical express links in rather fixed ways (e.g., [8][13][17]), representing only a few design points in a very large design space. In fact, the number of possible valid ways of placing express links is a super-exponential function of the network size, even under the constraints of bisection bandwidth. This not only presents a large opportunity for better express link placement, but also calls for efficient algorithms that are able to find optimal or near-optimal placement under network constraints in acceptable runtime.

This paper explores the opportunities for finding effective express link placement for general-purpose many-core platforms to minimize average packet latency under bisection bandwidth constraints. This problem, however, is very difficult as the solution space containing all possible link placement combinations grows extremely rapidly with respect to network sizes, which makes enumeration-based optimal algorithms impractical. Furthermore, the large number of invalid placements due to bandwidth constraints may also slow down search-based algorithms (e.g., simulated annealing) quite significantly.

To solve this optimization problem, we first propose a way to transform a typical two-dimensional placement problem into a one-dimensional problem while retaining the optimality of potential solutions. We then propose an efficient divide and conquer algorithm to generate the initial input to simulated annealing which greatly increases the effectiveness. To further improve the efficiency of the proposed algorithm, we devise a connection-matrix-based solution

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
ICPP 2019, August 5–8, 2019, Kyoto, Japan
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6295-5/19/08…$15.00
https://doi.org/10.1145/3337821.3337877

space and a candidate generator that cleverly exclude the invalid placement of express links while guaranteeing that all possible solutions are still probabilistically reachable. We evaluate the proposed scheme on 4x4, 8x8 and 16x16 networks using PARSEC benchmarks as well as synthetic traffic patterns. Evaluation results show a significant reduction of packet latency (23.5% to 36.4%) compared with previous schemes while incurring neglibable hardware overhead (less than 0.5%).

The rest of this paper is organized as follows. Section 2 provides the related work and background information on express links. Section 3 formulates the problem of express link placement, and Section 4 elaborates the proposed solution and system implementation. Section 5 evaluates the proposed express link placement scheme, and finally Section 6 concludes this paper.

2 BACKGROUND

2.1 Related Work

With tens to hundreds of cores integrated in a many-core processor, the performance as well as the scalability of on-chip networks have become one of the primary challenges for NoC designers. Mainstream mesh topologies are easy to implement and more scalable than ring or bus topologies, but the average on-chip latency of mesh still increases linearly with the network diameter.

Adding express links to existing mesh-based NoCs is a promising solution to improve the scalability of NoCs. One of these express link technologies is to hybridize the electronic NoCs with photonic interconnects or multi-band radio frequency interconnects (RF-I) to increase NoC performance. For example, Chang *et al.* propose a general-purpose NoC with RF-I to enhance performance [6]. Bahirat *et al.* propose a hybrid photonic NoC with a photonic ring waveguide combined into a mesh NoC [2]. Photonic links and RF-I deployments reduce the on-chip latency and power consumption, but they also need considerable extra technological and hardware support such as waveguides, optical-electronic and electronic-optical converters, signal mixers, etc., which may take quite a while to mature for volume production in on-chip settings.

Alternatively, conventional electronic express channels can be easily added between non-adjacent routers. It reduces the average number of hops traversed by network packets, thereby reducing on-chip communication latency. Researchers have proposed many application-specific designs that improve NoC topology by utilizing application characteristics (e.g., [11][14][20]). For example, Ogras *et al.* enhance mesh networks with additional long range links between frequently-communicating routers determined by traffic patterns of certain applications [20], and Dumitriu *et al.* present a NoC topology generation process to provide high performance for given applications [11]. However, the application-specific nature of these designs may lead to non-optimal solutions in the use of general-purpose processors.

For general-purpose many-core platforms, there are mainly two categories of express link-based approaches that are equally competitive, namely virtual express link approaches and physical express link approaches [6]. The *virtual* approach utilizes virtual express channels to allow certain packets to bypass the first few pipeline stages of intermediate routers [18]. This approach does not require actual additional links but packets cannot fully bypass the intermediate router stages (specifically, packets still need to go through the switch traversal and link traversal stages in most designs). Thus, it yields a limited reduction in packet latency.

In contrast, the *physical* approach deploys physical express links between routers [8][13][17], enabling maximum bypass effects but consuming additional bisection bandwidth. For example, Dally proposes a hierarchical express-link placement technique [8]. Kim *et al.* present flattened butterfly, which adds express links to form full connectivity between the (concentrated) routers in each row and column [17]. The flattened butterfly design successfully achieves low-diameter NoC topology. A variant of flattened butterfly is a multi-drop express channel topology that achieves high connectivity between routers but needs extra multiplexing logic [13]. Although these proposals highlight the promise of adding physical express links, they represent only a few specific examples of placement schemes while neglecting other potentially better placement in the design space.

This paper distinguishes itself from existing studies by aiming at finding optimal or near-optimal express link placement for general-purpose processors by exploring the entire placement design space. We identify the on-chip bisection bandwidth as a key constraint and factor that influences the overall packet latency.

2.2 Impact of Express Links on Latency

The on-chip latency of a packet is comprised of two components [9]:

$$L = L_D + L_S = (H \cdot T_r + D_M \cdot T_l + H \cdot T_c) + (S/b) \qquad (1)$$

where the first component $L_D = H \cdot T_r + D_M \cdot T_l$ is the *head latency*, representing the time required for the first bit of a packet to traverse the network. $H \cdot T_r$ calculates the overall delay in router pipeline stages. H is the number of hops that the packet goes through. Note that a hop can be either a bidirectional local link that connects two adjacent routers or a bidirectional express link that connects non-adjacent routers. Router delay T_r is the number of cycles that a packet takes to pass through a router. The overall link delay $D_M \cdot T_l$ for network links is proportional to their lengths. Express long-range links are segmented into unit-length links by repeaters [20]. Repeater insertion is necessary in long-range express links to maintain the desired data rate on the links [16]. D_M is the total Manhattan distance that a packet traverses from source to destination in the number of unit-length links. Unit-length link delay T_l is the time for a flit[1] to travel on a local link (one cycle). T_c is the average per-hop contention delay, which could become very high in theory, but in practice is usually quite low in general-purpose chip multiprocessors due to the typically low on-chip traffic load of applications, the wide on-chip links, and multiple virtual channels per link to reduce head-of-line blocking, as shown in empirical findings as well as recent studies such as [7].

The second component, L_S, is the *serialization latency*, representing the time for the rest of the packet to complete transmission at the destination after the arrival of the first bit. The serialization

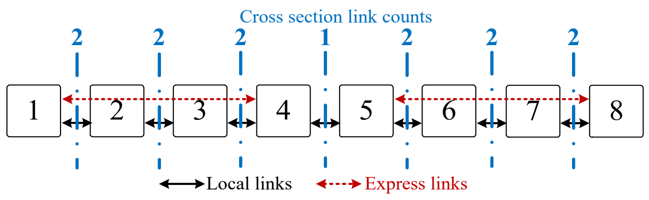


Figure 1: Express Links and bandwidth limit.

latency is calculated by S/b where S is the packet size in bits and b is the link width (or the flit size).

Although deploying express links can reduce the number of hops for a packet, it does not always result in reduced overall packet latency. This is because the total bisection bandwidth B of a NoC is limited by many factors, including chip dimension, manufacturing technology, and energy constraints to name a few [21]. Assume that the link count at the cross-section of two adjacent routers is c. For an $n \times n$ mesh, if we add express links such that each row or column has c links at the cross-section of two adjacent routers, the link width b needs to be adjusted in order to stay within the network bisection bandwidth constraint, i.e., $b \cdot c \cdot n \leq B$.

As an example, Figure 1 depicts the first row of an 8x8 mesh network. Initially, there are only local links and the maximum number of wires at the cross-section of two neighboring routers is 256 (i.e., flit size is 256 bits). For a packet that contains two flits (512 bits), the serialization latency L_S is two cycles. However, if we add express links (the red dotted lines), the link width b needs to be reduced to 128 bits. Consequently, each flit is 128-bit, and the same packet now needs to be transmitted using four flits, resulting in four-cycle serialization latency.

The above analysis indicates that there is a design space in express link-based NoCs that needs further exploration. On the one hand, we need to determine the appropriate link width to balance head latency and serialization latency under bisection bandwidth constraints. On the other hand, we also need to find the optimal placement of express links at a given link width so as to minimize average hop count. In the next section, we formulate this optimization problem mathematically and then present, in Section 4, several algorithms that produce effective placement results.

3 PROBLEM FORMULATION

We focus on the design of an express link-based on-chip network for general-purpose many-core platforms. The goal is to reduce the average NoC packet latency with the presence of multiple packet types with different sizes (e.g., short packets for read requests or write acknowledgements, and long packets for read replies or write requests). To reflect general-purpose computing, packet latency is averaged over all source-destination pairs to avoid unfairness during the optimization process. Various synthetic traffic patterns as well as application traffic are used during evaluation (Section 5).

Problem: Find the optimal number and placement of express links to be added to a $n \times n$ mesh network under a given bisection bandwidth constraint B.

Objective: Minimize the average on-chip packet latency L_{avg},

$$L_{avg} = L_{D,avg} + L_{S,avg} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} L_D(i,j)}{N \cdot N} + \sum_k p_k \frac{S_k}{b}, \quad (2)$$

where $N = n^2$ is the total number of routers in the network, $L_D(i,j)$ is the packet head latency from router i to router j, determined by the express link placement; p_k and S_k are the percentage and size of the k-th type of packets, and b is the link width in bits.

Constraints: The number of links c at the cross section between any two adjacent routers (including both local and express links) in a row or column, must not exceed a link limit C, which is calculated as

$$c \leq C = \frac{B}{b \cdot n}, \forall c \quad (3)$$

Equations (2) and (3) imply that, with a specific value of C and thus fixed b and $L_{S,avg}$, the overall average packet latency L_{avg} is determined by the average head latency $L_{D,avg}$, and $L_{D,avg}$ is determined by the placement of express links.

4 PROPOSED APPROACH

To solve the above problem, the overall approach we take is to first determine all the possible values of C, and for each C, determine the optimal express link placement to minimize $L_{D,avg}$. The minimal average packet latency can then be found by comparing all cases of C values.

4.1 Cross-Section Link Limit

For a regular $n \times n$ mesh network, C has the minimum value of 1, meaning that neighboring routers are connected with one bidirectional link. When all routers on the same row (or column) are fully connected, C has the maximum value C_{full} at the cross-section between the two routers in the middle of a row or column, given by

$$C_{full} = \frac{n}{2} \cdot \frac{n}{2} = \frac{n^2}{4} \quad (4)$$

meaning that each router on one side of the center line is connected bi-directionally to all the routers on the other side. For example, $C_{full} = 4$ for a 4x4 network where the maximum link count occurs between the second and third router, and $C_{full} = 16$ for 8x8 network where the maximum link count occurs between the fourth and fifth router.

Since the possible flit sizes are very limited (the flit size or the link width in bits is typically a divisor of the packet size and is a power of 2), there are only a few possible values of C. For example, the value of C can be 1, 2, or 4 for 4x4 networks and 1, 2, 4, 8, or 16 for 8x8 networks.

In what follows, we use $P(n, C)$ to denote the express link placement problem that minimizes $L_{D,avg}$ on a $n \times n$ mesh NoC for a specific link limit value of C.

[1] A flow control digit or a flit is the basic control unit of a packet. The size of a flit is assumed to be the same as the link width.

4.2 Reduction from 2D to 1D

To tackle the express link placement problem, we start with the following key lemma that transforms the two-dimensional placement problem into a one-dimensional while retaining the optimality of potential solutions.

Lemma: For a specific value of link limit C, if dimension-order routing is deployed, the express link placement problem $P(n, C)$ on a two-dimensional $n \times n$ mesh is reducible to the problem of one-dimensional express link placement on a row (or column) of n routers that minimizes the average head latency L_D among these n routers.

Before presenting the proof, it is important to justify the assumption of dimension-order routing in the targeted problem domain of this paper. It is known that adaptive routing algorithms have higher maximum achievable throughput, and have been adopted in some off-chip networks in high-performance computing systems. However, the difference between adaptive routing and dimension-order routing (e.g., XY routing) is noticeable only when network loads approach saturation. For on-chip networks in many-core CMPs where the traffic load is rarely very high, dimension-order routing is as effective as adaptive routing most of the time. Due to this reason as well as the consideration of implementation cost, most, if not all, taped-out commercial and research many-core chips adopt XY or YX routing (such as Intel Teraflop [23], Intel SCC [15], TRIPS [12], and Scorpio [10]). Our simulation using multi-threaded benchmarks also shows that the average contention per hop is almost always less than 1 cycle, and the overall performance difference between XY and adaptive routing is less than 1%. Thus, to increase the applicability of this work in practical designs, we follow the design choice and assume dimension-order routing in this paper.

Proof: With dimension-order routing, given the source and destination routers, the routing path of a packet is comprised of a horizontal path component and/or a vertical path component. The on-chip traffic is then separated into horizontal traffic and vertical traffic. Express links can be added to directly connect two non-adjacent routers on the same row or column (for any two routers that are not on the same row or column, they can be connected by a horizontal express link plus a vertical express link).

Mathematically, assuming the routing path of a packet from router i to router j consists of a horizontal path and/or a vertical path, we have $L_D(i,j) = L_D(i, v_{ij}) + L_D(v_{ij}, j)$, where v_{ij} is the turning point, i.e., the router on the same row with i and on the same column with j. Since i and v_{ij} are on the same row, $L_D(i, v_{ij})$ is solely determined by the link placement on that row. Similarly, $L_D(v_{ij}, j)$ is determined by the link placement on the column that router j and v_{ij} share. The average packet latency of the head flit is then expressed by

$$L_{D,avg} = \frac{\sum_{r=1}^{n}\left(\sum_{i\in r}\sum_{j=1}^{n\cdot n} L_D(i, v_{ij})\right) + \sum_{c=1}^{n}\left(\sum_{j\in c}\sum_{i=1}^{n\cdot n} L_D(v_{ij}, j)\right)}{N\cdot N}$$
$$= \frac{\sum_{r=1}^{n}(n\sum_{i\in r}\sum_{v\in r} L_D(i,v)) + \sum_{c=1}^{n}\left(n\sum_{j\in c}\sum_{v\in c} L_D(v,j)\right)}{N^2}$$
$$= \frac{n\left(\sum_{r=1}^{n} n^2 L_{D,r} + \sum_{c=1}^{n} n^2 L_{D,c}\right)}{N^2} \quad (5)$$

where $L_{D,r}$ and $L_{D,c}$ denote the average packet latency on the r-th row and the c-th column, respectively. Equation (5) shows that the average latency can be minimized by minimizing the packet latency on each row or column individually. Therefore, the express link placement onto mesh network can be obtained by (i) solving a one-dimensional link placement problem for a row of n routers and (ii) replicating the result of this sub-procedure n times for n rows and another n times for n columns and combine them into a final result.

Hereinafter, we use $\hat{P}(n, C)$ to denote the one-dimensional express link placement problem on an n-router row with the link limit of C. Due to the geometric symmetry of general-purpose CMPs, $\hat{P}(n, C)$ only needs to be solved once to minimize pair-wise average packet latency for the n routers on the same dimension, and the solution can be duplicated for n rows and n columns.

4.3 Large Solution Space and Need for Heuristics

The solution space of $\hat{P}(n, C)$ is of size $O(2^{n\times n})$, which is the combination of any number of links between every router pair. However, note that not all combinations are valid. First, a valid combination must contain all the local links between adjacent routers. Second, at the cross-section between any two routers, the link count cannot exceed C. Nevertheless, the solution space is still a super-exponential function of n, which renders a brute-force solution impractical. Therefore, a properly designed heuristic is required for large networks and/or higher link limit.

4.4 Proposed Scheme

We propose an efficient simulated annealing-based algorithm to solve the one dimensional link placement problem $\hat{P}(n, C)$.

A general simulated annealing procedure starts from an initial solution, and performs sufficient times of probabilistic searches on the solution space. Its basic components include an initial solution, the solution space, a candidate generator, a cooling schedule, and an acceptance probability function. We adopt an exponential function for acceptance probability and a linear function for cooling. Since the number of searches needed for simulated annealing to locate a good solution, i.e., the efficiency of the algorithm, is greatly affected by the quality of both the initial solution and the neighboring states found by the candidate generator in each iteration, we present how to choose a good initial solution and design a good candidate generator in detail as follows.

4.4.1 Initial Solution Based on Divide-and-Conquer

The heuristic of initial solution generation should be efficient and effective. Divide-and-conquer (D&C) is very fast if the problem can be divided into sub-problems directly and its solution can be combined efficiently using the solutions to sub-problems.

As shown in the pseudo code of the initial solution generation procedure above, in the proposed algorithm, we divide $\hat{P}(n, C)$ into two problems of $\hat{P}(\lfloor n/2 \rfloor, C-1)$ and $\hat{P}(\lceil n/2 \rceil, C-1)$. The combination step is to add one express link between the solutions to sub-problems, which is fast to implement and also a good estimation to the optimal solution. When the size of the sub-problems becomes small after multiple divisions (e.g., $n \leq 4$), the local optimal

Procedure $I(n, C)$: generates initial solution for $\hat{P}(n, C)$.

```
1  initialize: label the $n$ routers from left to right as $1, 2, \ldots, n$.
2  if $n$ is small enough
3    call the branch and bound algorithm
4    return the local optimal placement solution
5  else
6    call $I(\lceil n/2 \rceil, C-1)$ to place the express links among Router 1 to $\lfloor n/2 \rfloor$
7    call $I(\lfloor n/2 \rfloor, C-1)$ to place the express links* among Router $\lfloor n/2 \rfloor + 1$ to $n$
8    foreach router pair $(i, j), i \leq \lfloor n/2 \rfloor, j > \lfloor n/2 \rfloor$
9      add an express link between them
10     evaluate the current express link placement
11     if the average latency is lower than the current minimum, update the minimum value
12   endfor
13 return the placement result with the minimum latency
```

*The previous placement result can be directly used if $\lfloor n/2 \rfloor = \lceil n/2 \rceil$

solution can be located by enumeration methods such as simple branch and bound.

We analyze the complexity of Procedure $I(n, C)$ using the master theorem [8] as follows. Each problem size of n is divided into 2 sub-problems. Combining the solutions of the two sub-problems has $O(n^2)$ iterations, and in each iteration the algorithm evaluates the current link placement in $O(n^3)$, which is elaborated shortly in Section 4.5.1 where we discuss how packets are routed. Therefore, the combination step takes $O(n^5)$ complexity, and based on the master theorem, the overall Procedure $I(n, C)$ also has $O(n^5) = O(N^{2.5})$ complexity, where N is the total number of nodes in the network.

4.4.2 Candidate Generator Based on Connection Matrix

An efficient candidate generator is a key factor to the efficiency of simulated annealing. A naive generator adds, deletes, stretches, or shortens a randomly selected link in each move. However, a new candidate solution generated this way is highly likely to fall out of the feasible solution space. More precisely, it might have some local links missing or exceed the link limit C. This greatly degrades the efficiency of the candidate generator. To tackle this problem, we identify an equivalent search space that excludes illegal link placements with no loss of possible valid solutions in the proposed algorithm as follows.

For $\hat{P}(n, C)$, we define a binary matrix M of size $(n-2) \times (C-1)$, referred to as the connection matrix hereinafter. Each element represents whether the two links on both sides of a router are connected. We elaborate the construction of such a matrix in Figure 2 for $\hat{P}(8,4)$ as an example. There are $(C-1)$ links available for express link construction as one layer of links is reserved for local links between adjacent routers (three layers as shown in Figure 2(a) as $C = 4$). In this way, we ensure the solution corresponding to this matrix has the required local links. In each express-link layer in Figure 2(a), the binary values at six connection points are used to denote whether the two links on both sides of the router are connected. For example, in the first layer (top layer) in Figure 2(a), the connection point at Router 3 is connected (solid dot), meaning the

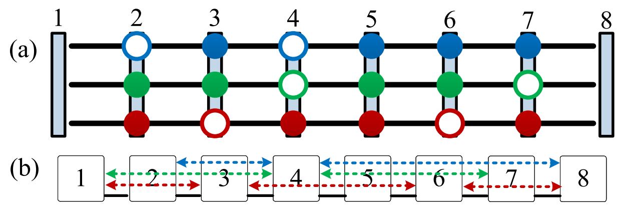


Figure 2: (a) Connection matrix. A solid dot means the two links of both sides are connected as one and a hole means disconnected. (b) The corresponding express link placement. From top to bottom, the blue, green, and red express links are denoted by the three layers of corresponding colors in the connection matrix, respectively.

two links of Router 3 in this layer are connected, making an express link from Router 2 and Router 4 as shown in the top layer of express links in Figure 2(b). Similarly, as the connection points at Router 5, 6, 7 are connected, there is a long express link from Router 4 to 8.

Based on the connection matrix, the candidate generator randomly picks one connection point and flips its value to form the new candidate solution in each move of the simulated annealing. Compared to the naïve annealing procedure in the search space of all links, the new procedure always conducts valid moves that satisfy constraints, and it can be proven that all the possible solutions are probabilistically reachable. As an example, the placement of express links shown in Figure 2 is the best solution to $\hat{P}(8,4)$ given by the proposed simulated annealing-based algorithm.

Table 1 lists the parameters used in the above simulated annealing algorithm, for a row of eight routers and the NoC settings mentioned in Section 5.1. Specifically for each newly generated candidate, the algorithm evaluates the difference in average latency ΔL_{avg} from that of the current candidate, and accepts the move to

Table 1. Parameters in simulated annealing.

Parameter Name	Value
Initial Temperature T_0	10 (cycles)
Total number of moves m	104
Cooldown scale S_c	2
Number of moves before each cooldown m_c	103

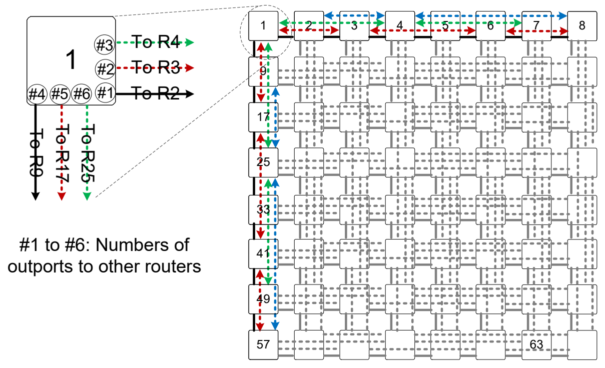
the new candidate if $\Delta L_{avg} \leq 0$ and otherwise accepts it with probability $e^{-\Delta L_{avg}/T}$ if $\Delta L_{avg} > 0$. The current temperature T starts at T_0, and is divided by S_c after each m_c moves.

4.5 System Implementation

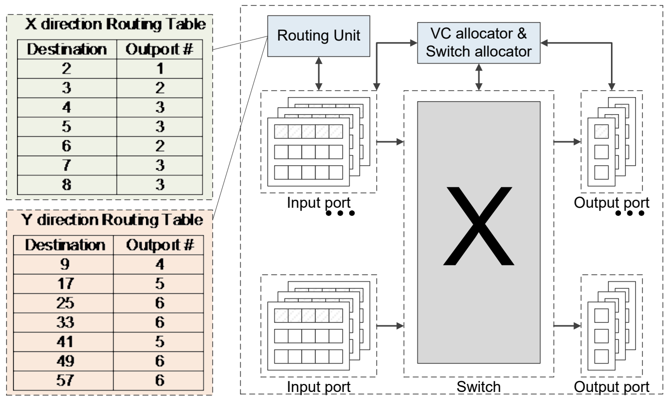
4.5.1 Deadlock-Free Routing

As express links are added on top of the mesh network, the routing paths need to be modified to take advantage of express links, while avoiding new deadlock that might be formed by using these express links. We avoid routing deadlock by enforcing packets to traverse unidirectionally and disallowing "U-turns". For instance,

[2] A formal proof based on this rationale can be easily derived. The proof is omitted here due to space constraints but is available upon request.



(a) Express links and outport numbering of Router 1 on 8x8 network.



X direction Routing Table

Destination	Outport #
2	1
3	2
4	3
5	3
6	2
7	3
8	3

Y direction Routing Table

Destination	Outport #
9	4
17	5
25	6
33	6
41	5
49	6
57	6

(b) Structure and routing table of Router 1.

Figure 3: An example of router implementation.

in Figure 2(b), a packet at Router 1 may use local and/or express links to traverse from left to right to reach Router 6, but the packet cannot use the express link to reach Router 7 first and then come back to Router 6. Specifically, each channel only depends on its downstream channels on the same direction. For the two-dimensional chip, we adopt a dimensional routing fashion, i.e., routing packets on X-dimension first and then Y dimension next. Enforcing this rule eliminates any cyclic dependencies between the channels in a dimension and across dimensions, thus avoiding deadlocks[2].

In addition to ensuring deadlock freedom, the routing algorithm should also minimize the path latency on links and routers. We adopt the following deterministic routing algorithm to achieve this. We first compute the directional shortest paths between all router pairs within each row (or each column, i.e., on one dimension) offline by applying Floyd-Warshall algorithm twice, one for each direction. For example, within a row, the first round of the algorithm calculates the shortest paths from router i to j where $i \in \{1, \ldots, n\}$ and $j > i$, i.e., the paths of packets sent from left to right. To implement this, all edges from j to i (from right to left) are set with infinite weight. The second round then calculates the shortest path from j to i (still $j > i$) by setting all edges from i to j to have infinite weight. The routing computation algorithm returns a look-up table for each router that stores the next-hop router number on the same row/column (more details in the next subsection).

The above step is performed to populate the routing tables in each router. It has a complexity of $O(n^3)$ as Floyd-Warshall algorithm has a cubic complexity [8]. This routing calculation step is executed during the proposed simulated annealing-based algorithm each time when a newly generated placement needs to be evaluated. In addition, at Line 10 in Procedure $I(n, C)$, the current express link placement is evaluated by using this step to determine routing paths and calculate average packet latency, and therefore also has a complexity of $O(n^3)$.

4.5.2 Router Implementation

Figure 3(a) depicts the router implementation. Compared to a typical mesh router, it has more input and output ports but with narrower link width for each port. When a packet arrives at a router, the router first uses XY routing to determine the destination or the X-to-Y turning point, and then uses the aforementioned look-up table (routing table) to find the next-hop router.

Two routing tables, one for X dimension and one for Y dimension, are associated with each router. Figure 3(b) shows a routing table example of the first router in the first row based on the optimal $\hat{P}(8,4)$ solution in Figure 2. For simplicity, we exclude local ports to/from network interfaces in links and the routing table. Router 1 has three connections on one dimension, thus six output ports in total for X and Y dimensions as shown in Figure 3(a). Its routing table records the output port number for each next-hop router on X or Y dimension (as well as its network interface(s) not shown here). For example, if a packet with destination 63 is currently in Router 1, the turning point router is Router 7 according to DOR. Then the packet is routed to Outport #3 based on the sixth entry in the X direction routing table, which directs the packet to the next-hop Router 4. The size of each routing table has at most $2(n-1)$ entries, so the hardware overhead of routing table is minimal. To evaluate this, we use the DSENT [22] NoC area model with 32nm bulk CMOS technology. The result shows that the overhead is less than 0.5% of the router.

4.6 Impact on Power

The power consumption of routers is comprised of dynamic power and static power. The dynamic power consumption is the summation of the dynamic power consumed by each router component, which is proportional to the switching activity factor of that component. A lower average hop count resulted from the use of express links means that the same packet is forwarded through fewer routers and links. This leads to lower router and link activities and potentially lower dynamic power consumption.

The router static power, however, is not affected much by express links. In typical on-chip routers, static power is dominated by input buffers and the crossbar. First, the buffers consume similar static power as long as the total size (in number of bits) is similar. As large buffer size may provide unfair performance advantage to a scheme, we configure the buffer size of each router to be the same for all schemes in comparison. This leads to similar buffer static power consumption as shown in Section 5.5.

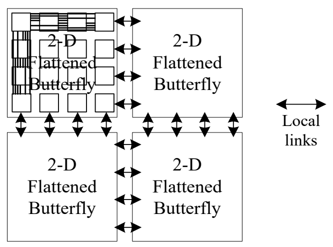


Figure 4: Hybrid flattened butterfly

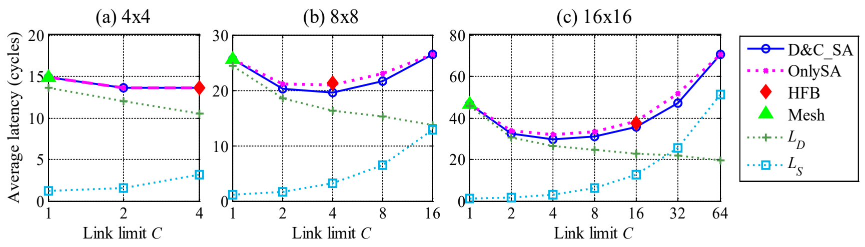


Figure 5: Average packet latencies as a function of link limit C.

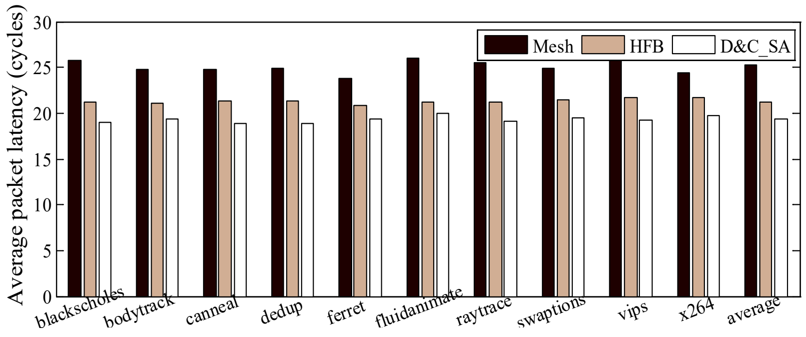


Figure 6: Average packet latency results on 8x8 net-

Second, for the crossbar, its static power consumption is proportional to $b \times k^2$, where b is the link width and k is the number of input ports. Assume b_m and k_m are the link width and the number of input ports of the mesh network, and b_e and k_e are for the cases where express links are used. The crossbar static power overhead with express links is very limited compared to that of the mesh network because of the following two reasons. First, due to the fixed bandwidth, the link width b_e of each input port decreases as more express links are added ($b_e = b_m/C$). Second, although adding express links may theoretically result in k_e to be as large as $C \cdot k_m$, the value of k_e does not always reach this maximum possible value. In fact, k_e is much smaller than $C \cdot k_m$ due to efficient express link placement. Take the optimal solution of $\hat{P}(8{,}4)$ shown in Figure 2 as an example within one row. In the same row, the original mesh network has two input ports within the row (i.e., $k_m = 2$) and $C = 4$, but none of the routers in $\hat{P}(8{,}4)$ has $2 \times 4 = 8$ input ports. The average number of input ports k_e in this case is only 3.5, indicating that the number of crossbar input ports does not increase linearly in good express link placement. These reasons lead to similar crossbar static power with and without express links, as confirmed by evaluation results.

5 EVALUATION

5.1 Evaluation Methodology

We evaluate the proposed express link placement algorithm on three different network sizes, namely 4x4, 8x8, and 16x16. A canonical 3-stage credit-based wormhole router is assumed. The flit size of the baseline mesh network is 256 bits. The bisection bandwidth increases proportionally with network size n. Based on previous findings [19], the ratio of long packets (512 bits) to short packets (128 bits) is set to 1:4 to reflect the characteristics of real applications.

We evaluate the proposed algorithms by running multi-threaded benchmarks on the cycle-accurate full-system simulator Gem5 [4] with GARNET [1] for detailed timing of the on-chip network. The evaluated PARSEC 2.0 benchmarks [3] include emerging applications such as recognition, mining, and synthesis (RMS) to represent a wide range of general-purpose computing applications. The latest DSENT [22] NoC power simulator is integrated into GARNET to estimate NoC power consumption.

We compare the following topologies/schemes in this section.

1) A mesh network (baseline system),
2) The hybrid flattened butterfly (HFB) as proposed in [17],
3) Mesh with express link placement given by the proposed simulated annealing with random initial placement (OnlySA), and
4) Mesh with express link placement given by the proposed simulated annealing with D&C-based initial placement (D&C_SA).

The hybrid flattened butterfly (HFB) given in [17] is proposed as an approach to scale the on-chip flattened butterfly beyond a 4x4 router network. It divides the network into four quadrants, each having a fully-connected flattened butterfly, and then connects them with local links. Figure 4 shows an example of HFB on an 8x8 network.

5.2 Results for PARSEC Benchmarks

Figure 5 plots the average packet latency as a function of link limit C, averaged over the ten PARSEC benchmarks, as well as the head latency L_D and the serialization latency L_S results of the proposed D&C_SA on 4x4, 8x8, and 16x16 networks. The best placement corresponds to the lowest point on the curve of D&C_SA. The Mesh and HFB are represented only as single design points as

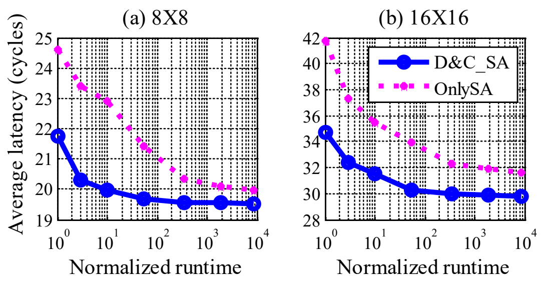


Figure 7: Runtime comparison.

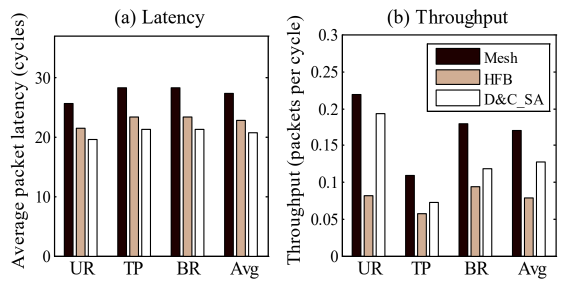


Figure 8: Network latency and throughput comparisons.

they are fixed designs, whereas the various placements of express links offer a wide range of design options. It can be seen that, as the link count increases, the serialization latency gradually cancels out the saving from the head latency.

Figure 5 shows that, the performance improvement of D&C_SA becomes much higher compared with the fixed topologies Mesh and HFB on 8x8 and 16x16 due to higher placement flexibilities. On the small 4x4 network, the proposed D&C_SA reduces the average packet latency by 8.1% compared with the Mesh, and has similar latency as the HFB. The latency savings, compared with the Mesh and HFB, increase to 23.5% and 8.0%, respectively, on the 8x8 network, and to 36.4% and 20.1%, respectively, on the 16x16 network. The packet latency difference between D&C_SA and OnlySA also enlarges as the network size increases, when both schemes are allowed with the same runtime (note that the scale of y-axis in Figure 5 is different from left to right). Specifically, both schemes achieve very similar average packet latency in the 4x4 network. However, on the 8x8 network, OnlySA results in 7.4% higher latency than D&C_SA, and the difference increases to 9.4% on the 16x16 network.

Figure 6 presents the 8x8 network in more detail, showing the average packet latency of the Mesh, HFB, and proposed D&C_SA for each benchmark. As mentioned previously, the placement of the proposed D&C_SA is obtained by first finding the local best placement for each value of link limit C and then comparing the placements from different link limit value cases. It can be seen that, despite the potentially distinct traffic behaviors among the benchmarks, the proposed D&C_SA is able to achieve similar reduction in the average packet latency across the benchmarks. This demonstrates the suitability of the proposed scheme for general-purpose computing.

5.3 Runtime Comparison of SA Schemes

We assess the effectiveness of the proposed initial solution generation by comparing the results of OnlySA and D&C_SA given the same runtime (i.e., how long the algorithm is allowed to run and search for good placements; due to the nature of simulated annealing, the longer the algorithm runs, the closer the placement would approach the optimal case). Figure 7 plots the “goodness” of the placement (i.e., lower average packet latency) of the two schemes with a wide range of allowed runtimes. The runtime is normalized to that of the initial process $I(8,4)$ for the 8x8 network and $I(16,4)$ for the 16x16 network. To reduce the randomness in simulated annealing, the figure shows the average results of the benchmarks. Even after a very long runtime (e.g., 10,000 units of the normalized runtime), OnlySA is not able to reach the same results as D&C_SA; whereas the proposed D&C_SA generates a satisfying result at about 150 units of the normalized runtime. This indicates that the proposed initial solution generation process is able to not only save runtime but also assist in achieving better express link placement.

5.4 Results for Synthetic Traffic Patterns

Besides PARSEC benchmarks, we also evaluate the proposed scheme on several representative synthetic traffic patterns, including uniform random (UR), transpose (TP), and bit-reverse (BR). Figure 8(a) shows the average packet latency for the 8x8 network. The proposed scheme achieves an average of 24.4% and 16.9% latency reduction compared to the Mesh and HFB, respectively.

Due to the fact that real applications typically have very low network load, on-chip networks are more sensitive to latency than throughput. Adding express links in general is a way to leverage this fact by reducing latency at the cost of lowered throughput, as reported in prior work [13]. Adding express links may lead to insufficient utilization of the overall bandwidth between two routers. For example in Figure 2(b), there are only three links between Routers 1 and 2 or Routers 7 and 8 while the maximum link allowed is four, meaning that the bandwidth is not fully utilized. Figure 8(b) compares the throughput results of the three topologies. The Mesh has the highest throughput. The use of express links in the HFB results in less than half of the Mesh throughput, mainly because of the bottleneck links between 2-D flattened butterfly blocks shown in Figure 4. In contrast, the proposed D&C_SA recovers a large part of the unused bandwidth in the HFB. The D&C_SA has higher throughput than the HFB in all the three traffic patterns, with an average of 63.7% higher throughput compared to the HFB. It also restores to more than three quarters of the Mesh throughput.

5.5 Power Consumption

Figure 9 shows the dynamic and static power consumption of the PARSEC benchmarks for the Mesh, HFB, and proposed scheme. The changes in the number of router ports and the datapath width of each port are all accounted for in different schemes. As can be seen from the results, the static power consumption accounts for about two-thirds of the overall power consumption, confirming the relatively low average activity of NoC components. The proposed express link placement algorithm reduces the total router power consumption by 10.4% compared to the Mesh and 0.6% compared

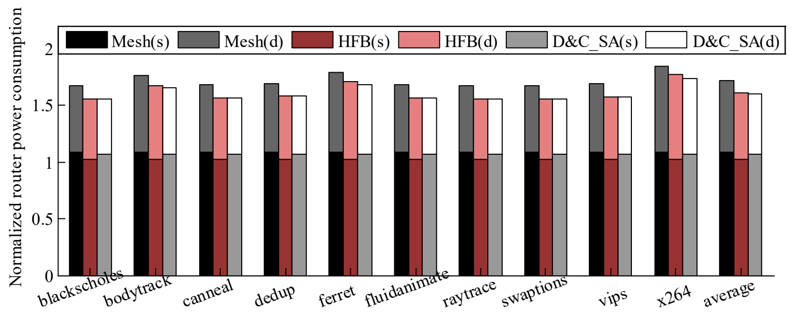


Figure 9: Router power consumption comparison on an 8x8 network.

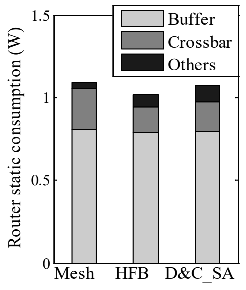


Figure 10: Router static breakdown.

Table 2. Maximum zero-load packet latency (cycles).

Topology	4x4	8x8	16x16
Mesh	28.2	60.2	71.2
HFB	15.2	38.2	63.8
D&C_SA	13.6	33.2	55.2

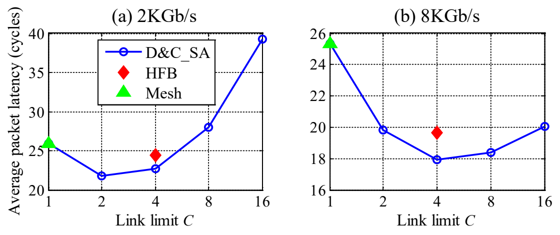


Figure 11: Bandwidth impact.

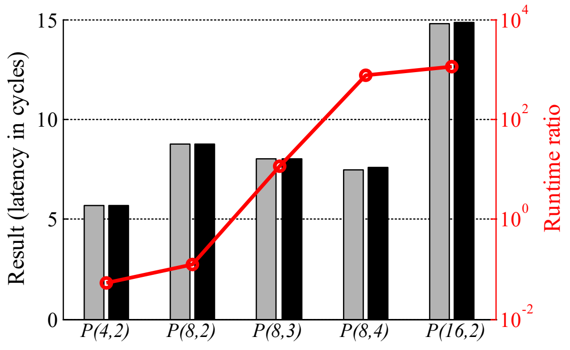


Figure 12: Latency and runtime compared with optimal.

to the HFB on average for the 8x8 network. The power saving mainly comes from the reduction in dynamic power consumption due to reduced packet forwarding activities. The dynamic power in D&C_SA is reduced by 15.1% and 6.6% compared to the Mesh and HFB, respectively. It can also be seen in Figure 9 that the static power consumption of the three topologies is very similar, as explained in Section 4.6. Figure 10 further breaks down the router static power consumption. It confirms that the static power of crossbar does not necessarily increase when express links are added. This is mainly because of the reduced link width associated with the increase in link count, as well as the sub-linear increase in the number of input ports in good express link placement.

5.6 Discussions

5.6.1 Worst-Case Latency

A desirable express link placement scheme for general-purpose processors should provide good support for various traffic patterns, including worst case traffic. Table 2 shows the packet latency for the worst-case, maximum zero-load on-chip latency between any two routers in the network. As can be seen, while the maximum latency increases quickly as the network size increases, the proposed D&C_SA is still considerably superior to the Mesh and HFB across different network sizes.

5.6.2 Impact of Bandwidth Limitations

The bisection bandwidth available for on-chip networks may have a large impact on the network latency. For example, Figure 11 plots the case for the 8x8 network operating at 1.0GHz with the bisection bandwidth increases from 2KGb/s to 8KGb/s (i.e., equivalent to the flit size changing from 128 bits to 512 bits). As the bandwidth increases, the mesh network benefits only from the reduction in serialization latency due to the increased capacity to accommodate larger flit sizes, resulting in latency reduction from 25.9 cycles to 25.3 cycles, or 2.3%. In comparison, good express link placement can take advantage of the increased bandwidth by using multiple narrower express links, thus reducing the average latency from 21.8 cycles to 17.9 cycles, or 17.8% reduction. This highlights that the proposed express link placement algorithm is very effective in utilizing on-chip bandwidth for latency reduction.

5.6.3 Comparison to Optimal

D&C_SA achieves near-optimal results and, for smaller network sizes and link limits, we can verify this by comparing with the optimal solution that can be obtained by exhaustive search algorithm with branch and bound. Figure 12 shows the result comparison of D&C_SA against optimal results for $P(4,2)$, $P(8,2)$, $P(8,3)$, $P(8,4)$, and $P(16,2)$. It can be seen that D&C_SA achieves exactly the same results as the optimal solutions in $P(4,2)$, $P(8,2)$, and $P(8,3)$, and only 1.3% and 0.28% higher in latency compared to the optimal solutions for $P(8,4)$ and $P(16,2)$, respectively. Meanwhile, the exhaustive search has approximately 30X nad 1000X runtime compared to D&C_SA for $P(8,3)$ and $P(16,2)$, respectively.

5.6.4 Application-Specific Design

The proposed problem formulation aims at minimizing the average packet latency between all the source and destination pairs to represent general-purpose designs and provide fairness among various possible applications that may be run on the processors. If the traffic pattern of an executed application is known and relatively fixed, we can further improve the express link placement with the given traffic information. The head latency can be expressed as

$$L_{D,avg} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\gamma_{ij}L_D(i,j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\gamma_{ij}},$$

where γ_{ij} is the communication rate from router i to router j. Similar to the pairwise packet latency, this application-aware latency can be separated into row latency and column latency, and the solution to the application-aware express link placement problem can also be proven as a combination of the one-dimensional placement problem. Note that the one-dimensional problem $\hat{P}(n, C)$ now needs to be solved row by row and column by column instead of simply duplicating results as each row/column has different γ_{ij} values. However, the proposed divide-and-conquer method for generating initial solution and the cleverly-designed connection matrix for representing the search space are still applicable.

To demonstrate the above applicability, we first run each PARSEC benchmark on a baseline 8x8 network once to collect traffic statistics, and then apply the revised scheme. Simulation results show that, with the knowledge of traffic patterns available in advance, the proposed scheme is able to reduce an additional 18.1% of average packet latency reduction on top of the case without advanced traffic information. This indicates that, while the proposed scheme mainly targets general-purpose computing, it is useful in the application-specific scenarios as well.

6 CONCLUSION

This paper investigates the opportunities in express link placement for general-purpose many-core platforms useful in parallel processing systems, and explores a large design space as opposed to the few design points proposed previously. In order to minimize the average packet latency under bisection bandwidth constraints, we need to find good express link placement that balances the number of bisection express links and the serialization latency. To achieve that, we transform the design space of express link placement problem from two-dimensional to one-dimensional and propose an efficient simulated annealing-based algorithm. The algorithm adopts divide-and-conquer to increase the effectiveness of the initial solution and uses a connection-matrix based search space to remove invalid placements so as to speed up the simulated annealing procedure. Evaluation results demonstrate the effectiveness of the proposed scheme. It achieves 23.5% and 8.0% average packet latency reduction on the 8x8 network and 36.4% and 20.1% on the16x16 network compared to the traditional mesh topology and the hybrid flattened butterfly, respectively.

ACKNOWLEDGMENTS

We sincerely thank the reviewers for their helpful comments and suggestions. This research was supported, in part, by the National Science Foundation grants #1619456, #1619472 and #1750047.

REFERENCES

[1] Agarwal, N., Krishna, T., Peh, L. S., & Jha, N. K. (2009, April). "GARNET: A detailed on-chip network model inside a full-system simulator," International Symposium on Performance Analysis of Systems and Software (ISPASS), 33-42.

[2] Bahirat, S., & Pasricha, S. (2009). "Exploring hybrid photonic networks-on-chip for emerging chip multiprocessors," Proceedings of the 7th IEEE/ACM international conference on Hardware/software codesign and system synthesis, 129-136.

[3] Bienia, C., Kumar, S., Singh, J. P., & Li, K. (2008). "The PARSEC benchmark suite: Characterization and architectural implications," In 17th International Conference on Parallel Architectures and Compilation Techniques (PACT), 72-81.

[4] Binkert, N., et al. (2011). "The gem5 simulator," ACM SIGARCH Computer Architecture News, 39(2), 1-7.

[5] Chang, M. F., et al. (2008) "CMP network-on-chip overlaid with multi-band RF-interconnect," High Performance Computer Architecture (HPCA) IEEE 14th International Symposium on.

[6] Chen, C. et al. (2010). "Physical vs. virtual express topologies with low-swing links for future many-core nocs," 4th ACM/IEEE International Symposium on Networks-on-Chip (NOCS), 173-180.

[7] Chen, L. and Pinkston, T. M. (2012). "NoRD: Node-Router Decoupling for Effective Power-gating of On-Chip Routers," In 45th IEEE/ACM International Symposium on Microarchitecture (MICRO), 270-281.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to algorithms," MIT press, 2001.

[9] Dally, W. J. (1991). "Express Cubes: Improving the Performance of k-ary n-cube Interconnection Networks," In IEEE Transactions on Computers, 40(9), 1016-1023.

[10] Dally, W. J., & Towles, B. P. (2004). Principles and practices of interconnection networks. Elsevier.

[11] Daya, B. K, et al. (2014). "SCORPIO: a 36-core research chip demonstrating snoopy coherence on a scalable mesh NoC with in-network ordering," In IEEE International Symposium on Computer Architecture (ISCA).

[12] Dumitriu, V., & Khan, G. N. (2009). "Throughput-oriented NoC topology generation and analysis for high performance SoCs," In IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 17(10), 1433-1446.

[13] Gratz, P., et al. (2007). "On-chip interconnection networks of the TRIPS chip," IEEE Micro, vol. 27, 41-50.

[14] Grot, B., Hestness, J., Keckler, S. W., & Mutlu, O. (2009). "Express cube topologies for on-chip interconnects," In 15th IEEE International Symposium on High Performance Computer Architecture (HPCA), 163-174.

[15] Ho, W., & Pinkston, T. (2006). "A Design Methodology for Efficient Application-Specific On-Chip Interconnects," In IEEE Transactions on Parallel & Distributed Systems (TPDS), vol.17, no. 2, 174-190.

[16] Howard, J., et al. (2010). "A 48-core IA-32 message-passing processor with DVFS in 45nm CMOS," In Proceedings of the International Solid-State Circuits Conference (ISSCC).

[17] Kim, B. and Stojanovi´c, V. (2007). "Equalized interconnects for on-chip networks: modeling and optimization framework," In Int'l Conference Computer-Aided Design (ICCAD), 552–559.

[18] Kim, J., Balfour, J., & Dally, W. (2007). "Flattened butterfly topology for on-chip networks," In Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 172-182.

[19] Kumar, A., Peh, L. S., Kundu, P., & Jha, N. K. (2007). "Express virtual channels: towards the ideal interconnection fabric," In ACM SIGARCH Computer Architecture News, 35(2), 150-161.

[20] Ma, S., Jerger, N. E., & Wang, Z. (2012). "Whole packet forwarding: Efficient design of fully adaptive routing algorithms for networks-on-chip," In International Symposium on High Performance Computer Architecture (HPCA), 1-12.

[21] Ogras, U. Y., & Marculescu, R. (2006). "It's a small world after all": NoC performance optimization via long-range link insertion. In IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 14(7), 693-706.

[22] Park, S., et al. (2012). "Approaching the theoretical limits of a mesh NoC with a 16-node chip prototype in 45nm SOI," In Proceedings of the 49th ACM Annual Design Automation Conference (DAC), 398-405.

[23] Sun, C., et al. (2012). "DSENT-a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," In IEEE/ACM International Symposium on Networks-on-Chip (NOCS), 201-210.

[24] S. Vangal, et al. (2007). "An 80-tile 1.28 TFLOPS network-on-chip in 65nm CMOS," In International Solid-State Circuits Conference (ISSCC).