Advancing Supercomputer Performance Through Interconnection Topology Synthesis

Yi Zhu, Michael Taylor, Scott B. Baden and Chung-Kuan Cheng

Department of Computer Science and Engineering
University of California, San Diego
9500 Gilman Dr., La Jolla, CA 92093-0404, U.S.A.
Email: {y2zhu,mbtaylor,baden,kuan}@cs.ucsd.edu

***Abstract*—In today's many-core era, the interconnection networks have been the key factor that dominates the performance of a computer system. In this paper, we propose a design flow to discover the best topology in terms of the communication latency and physical constraints. First a set of representative candidate topologies are generated for the interconnection networks among computing chips; then an efficient multi-commodity flow algorithm is devised to evaluate the performance. The experiments show that the best topologies identified by our algorithm can achieve better average latency compared to the existing networks.**

I. Introduction

Interconnection networks play an important role in the multi-processor system. Currently, massively parallel computer systems have become popular, where the interconnection networks consist of a lot of processing cores. For example, IBM Blue Gene/L has 65,536 processors located in 64 racks [4] and the Cray Black Widow networks scales up to 32K processors [12]. Thus, the interconnection networks in these systems have become a more critical factor in the performance of a computer system than the computing or memory modules [3]. Communication latency, which largely depends on the interconnection network, is of great concern in these current multiprocessor systems and has become crucial with the growth of system sizes and shrink of clock cycles [12].

Most interconnection networks in the current multiprocessor systems make use of regular low-radix topologies, among which the k-ary n-cube [2] and torus topologies are the most often used [12]. Although the simpler designs imply simpler design and manufacture, they are probably unable to capture the bottleneck of the communications among processors and utilize the limited interconnect resources (e.g. wires, pins, connectors), therefore affecting the performance of the entire system. In this paper, we propose a design methodology that is able to select the best interconnection network topology among a large number of candidates so that the average communication latency is minimized. The major contributions of our work are as follows:

Firstly, we propose a fully automated design flow that is able to evaluate thousands of network topologies and find the best candidate according to the available technology, physical constraints and applications, which appear as the parameters of the flow, and thus can be specified and modified by users. This feature enables our methodology to be applicable for different supercomputer systems, with little modification.

Secondly, we make use of a fast multi-commodity flow (MCF) solver which is based on an efficient approximation algorithm to evaluate the performance of different topologies. In our experiments, it takes less than one minute to evaluate one topology, which makes it possible to explore a large design space to discover the most suitable topology among the candidate pool, which is entirely determined by users.

Thirdly, we demonstrate our flow by using the packaging framework of Blue Gene/L supercomputer. We conduct the experiments using different input parameters. The optimal designs we find in a midplane under different circumstance can improve 12% – 56% of the average communication latency compared to the original 3D torus design, which shows the effectiveness of our methodology.

The project is partially supported by California MICRO Program.

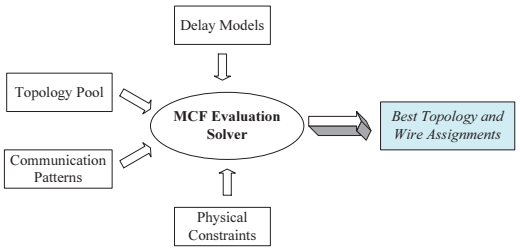


Fig. 1. Design Flow

The rest of the paper is organized as follows. Section II will introduce our design flow and the general formulation to evaluate the performance of interconnection networks. We will describe the MCF algorithms in Section III. In Section IV, we will use the packaging framework of Blue Gene/L as a concrete example to demonstrate our design flow. The experimental results will be presented in Section V. Conclusions will be given in the last section.

II. General Design Flow & Formulation

Fig. 1 shows the general design flow of our methodology. Users need to provide four inputs to perform the topology synthesis: the topology pool contains all candidate topologies users want to use; the delay models of wires and routers are based on their architectures, which are used to calculate the communication latency; the communication patterns among processors will be fed into our evaluation algorithm so that the synthesized network satisfies the communication demands; and users need to discover the physical constraints which must not be violated, including board dimensions, number of layers, number of pins, and number of connectors.

All these inputs are fed into the MCF solver to evaluate the performance and the best topology will be selected according to the communication latency. The MCF solver is able to take all the inputs, perform the evaluation, and provide the synthesis results which can be used by the designers. As the core component of our flow, we introduce the formulation of the MCF evaluation solver first.

We model a given interconnection network topology as a graph $G(V, E)$, where $|V| = n$, $|E| = m$. Each node represents a computing module (processor) and each edge (a, b) represents a link between processors a and b. A set of k communication demands are given, i.e. d_j represents the demand from processors (nodes) s_j to t_j, $\forall 1 \leq j \leq k$. The communication demands represent the demand for bandwidth between each pair of nodes; this quantity becomes the injection rate when averaged over the lifetime of the network that we profiled. Our task is to determine how many wires need to be assigned for each link, so that the physical constraints are satisfied and the overall average latency is minimized.

The above statement can be formulated as a multi-commodity flow problem if we consider the communication demand from s to t as a commodity with source node s to sink node t, and the number of wires for link $e \in E$ as the flow $f(e)$. The average communication latency consists of two portions: the router delay and the wire delay. If we assume there is a router with each node, our objective can be written as follows:

$$\sum_{e \in E} f(e) \cdot D^R_{g(e)} + \sum_{e \in E} f(e) \cdot D^W_e \tag{1}$$

where $D^R_{g(e)}$ represents the delay for the router in node $g(e)$ that is the starting node of edge e, and D^W_e represents the delay for link e. Therefore the first part of the objective denotes the total latency on routers and the second part is the latency on wires.

The physical constraints restrict the number of wires we are able to use. In our formulation, we consider two types of physical constraints:

1) *Intra-board constraints*: a set of wires that link the nodes in different regions in one board must not exceed the cross section of the board. The resource used for each wire can be estimated by the wire pitches and the available cross section can be calculated by the board dimension and the number of signal layers:

$$\sum_{e \in E^{cross}_q} f(e) \cdot pitch \leq W_q \cdot L_q \tag{2}$$

where E^{cross}_q is the set of edges that pass over the cross section q, W_q is the available cross section width and L_q is the number of signal layers in the board.

2) *Inter-board constraints*: the number of wires connecting nodes on different boards is usually constrained by the number of pins in the board connectors. Similarly, we can write the following constraint:

$$\sum_{e \in E^{pin}_r} f(e) \cdot K^{pin}_r \leq C_r \cdot V^{pin}_r \tag{3}$$

where E^{pin}_r is the set of edges that cross over the connector r, K^{pin}_r is the number of pins needed for each wire, C_r is the number of available connectors, and V^{pin}_r is the number of pins in each connector.

The above is a very general formulation to evaluate the communication latency in a supercomputer system with the given topology. It is applicable to any multiprocessor interconnection network by deriving the delay models and setting the parameters accordingly.

III. MCF Algorithms

The MCF solver is implemented using a polynomial time approximation scheme (PTAS), which is able to find $(1+\epsilon)$ optimal solutions in polynomial time, where ϵ is an input parameter to control the accuracy. The idea is based on the primal-dual theory in linear programming by Karakostas [10]. First we transform the problem into a maximum concurrent flow problem, i.e. we would like to route $\lambda \cdot d_j$ units flow for each commodity j, subject to the physical constraints and a latency upper bound LT, where λ is a scalar value representing the throughput. We use path-based flow variables: let p_j denote the set of flow paths from s_j to t_j for commodity j; $f(p)$ denote the flow along path p, $p \in p_j$. Then the LP primal formulation could be written as follows:

$$\begin{array}{lrcl}
\text{Max}: & \lambda & & \\
\forall j: & \sum_{p \in p_j} f(p) & \geq & \lambda \cdot d_j \\
\forall q: & \sum_{e \in E^{cross}_q} \sum_{p: e \in p} f(p) \cdot pitch & \leq & W_q \cdot L_q \\
\forall r: & \sum_{e \in E^{pin}_r} \sum_{p: e \in p} f(p) \cdot K^{pin}_r & \leq & C_r \cdot V^{pin}_r \\
 & \sum_{j=1}^{k} \sum_{p \in p_j} \sum_{e \in p} f(p) \cdot (D^R_{g(e)} + D^W_e) & \leq & LT \\
\forall p: & f(p) & \geq & 0
\end{array}$$

Then we write the dual formulation. Let Z_j be the dual variable for the demand constraint for d_j, X_q and Y_r be the dual variables for the cross section and connector constraints, and ϕ be the dual variable associated with the latency constraint:

$$\begin{array}{lrcl}
\text{Min}: & \sum_q W_q \cdot L_q \cdot X_q + \sum_r C_r \cdot V^{pin}_r \cdot Y_r + LT \cdot \phi & & \\
\forall j, \forall p \in p_j: & \sum_{e \in p} pitch \cdot \sum_{q: e \in E^{cross}_q} X_q + & & \\
 & \sum_{e \in p} K^{pin}_r \cdot \sum_{r: e \in E^{pin}_r} Y_r + \sum_{e \in p} (D^R_{g(e)} + D^W_e) \cdot \phi & \geq & Z_j \\
 & \sum_{j=1}^{k} d_j Z_j & \geq & 1
\end{array}$$

According to the above dual formulation, we can define the length of edge e as

$$l(e) = pitch \cdot \sum_{e \in E^{cross}_q} X_q + K^{pin}_r \cdot \sum_{e \in E^{pin}_r} Y_r + LT \cdot \phi \tag{4}$$

The PTAS will iteratively route the flow along the shortest path from s_j to t_j, according to the length function above, and update the dual variables according to the current flows. Therefore the primal and dual values will be simultaneously updated until the gap is small enough, which implies that it is sufficiently close to the optimal solution according to the LP primal-dual theory. For the detailed algorithms and convergence proof, please refer to [10] and [9].

The concurrent flow algorithm described above is able to maximize the flow amount λ, with the given latency bound LT. Thus, in order to minimize the total latency, we perform binary search on the value LT by checking whether the current λ exceeds 1. The binary search is accelerated by the interval estimation heuristic proposed in [9].

IV. Topology Synthesis in Blue Gene/L: An Example

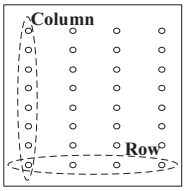
Blue Gene/L computer is a massively parallel supercomputer based on IBM system-on-chip technology. It is designed to scale to 65, 536 dual-processor nodes (computer ASICs) [1][4]. The entire system is organized as 2 nodes per compute card, 16 compute cards per node card, 16 node cards per 512-node midplane, 2 midplanes in a 1024-node rack, and totally 64 racks. The nodes are connected using 3-dimensional torus networks. Each 512-node midplane contains an 8 $\times$ 8 $\times$ 8 torus. In this work, we will consider the networks within one midplane, i.e. the topologies that connect 512 nodes.

In our work, we make the following two assumptions. First, we follow the same hierarchical structure of midplane/node card/compute card in our design. The number of nodes in each board remains the same too. Second, the properties of the boards, including dimensions, number of layers and dielectric keep unchanged. We will seek better topologies than the existing 3D torus to implement the networks in the Blue Gene/L midplane.

We use a 512-node graph to model a midplane, where each node in the graph represents a computer ASIC. Since the ASICs in a midplane are organized hierarchically, we also build a two-level graph model that captures the topologies in one node card (low-level) and the entire midplane (high-level) respectively. The low-level graph therefore consists of 32 nodes, as one node card contains 16 compute cards and one compute card has 2 nodes. The 32 nodes are arranged in 8 $\times$ 4 grids, according to their locations in a node card, as shown in Fig. 2 (a) [7]. Fig. 2 (b) shows the organization of the low-level graph. We highlight one row and one column, where we could plug in different topologies and duplicate them to all rows and columns. We follow the method in [8] to generate all topologies of 8 nodes, with the limitation that the degree of each node is no more than 3. There are totally 192 isomorph-free topologies. Then we place each topology in a line, which results 2092 different linear placements. They are duplicated to each column. We apply the similar method on the high level graph.

We should derive the delay models for both wires and routers. We assume that differential wires are used in the Blue Gene/L boards.

(a) Blue Gene/L Node Card



(b) Graph Model in Node Card

Fig. 2. Blue Gene/L Node Card and Topology

We estimate the unit length delay as the speed of light in the card dielectrics FR4. Therefore the unit length delay is roughly 7.1 ns/m.

The router delay is related to the radix of routers. We make use of the formula derived in [11] which is based on the logical effort, and assume $90nm$ design technology is used.

We should consider two types physical constraints: the available routing area for wires in cards, and the number of pins in the connectors among cards. The first factor is determined by the board dimensions, number of layers and wire pitches, and the second one is determined by the number of connectors and the number of pins per connector, as discussed in Section II. The details of the boards and connectors are obtained in [7]. We do not present the numbers here due to the space limitation.

V. Experimental Results

We implemented the MCF solver using the C language to test the design flow we proposed in a Linux machine with a 2.8GHz CPU and 2GB memory. The experiments were conducted on two groups of test cases: the first group is randomly generated, and the second group contains existing benchmarks.

A. Experiments on Generated Instances

We generate test cases which are the communication pairs among processors. The communication patterns are randomly generated; however, they are controlled by the following parameters:

- *Total number of communication demands* (T): It is reasonable to assume there are only $O(n)$ pairs of communications, as the examples in the benchmark suites in [6].
- *Communication amount/coefficient* (d): Without loss of generality, we assume all the communications have uniform traffic.
- *Communication distribution probability*: During the processor task assignment, we know that the tasks which need to communicate with each other will be assigned to processors that are close together [6].

1) Latency and Throughput Tradeoffs: To demonstrate the tradeoff between latency and communication throughput, we first fix the communication distribution: 40% of the communications happen within a compute card, 50% of the communications happen crossing compute cards but within a node card, and the rest 10% happen across node cards. We use three groups of cases where the total number of communication demands $T = 2048, 3072$ and 4096 respectively (4x, 6x and 8x of the total number of nodes). The latency-throughput tradeoff curves with optimal topology selection are shown in Fig. 3. The X-axis is the traffic demand coefficient (throughput) for each communication demand (all are uniform); the Y-axis is the average latency, which is the total latency on wires and routers divided by the total amount of communications. Different points indicate that different topologies are used. We also compare the latency values of 3D torus networks and our optimal topologies. The results are shown in Table I, where "Capacity" means the maximum number of demand amount it can accommodate; "Min Latency" and "Max Latency" denote the average latency when the demand has minimum and maximum values. For 3D torus topology, these two values are the same since there are no topology optimizations.

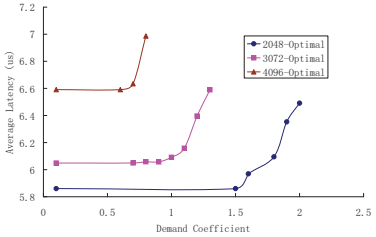


Fig. 3. Latency-Throughput Tradeoff Curves with Fixed Communication Distribution

# Demands	Topology	Capacity	Min Latency (us)	Max Latency (us)
2048	Optimal	2	5.86	6.49
	3D Torus	1.1	6.63	6.63
3072	Optimal	1.3	6.05	6.56
	3D Torus	0.7	10.54	10.54
4096	Optimal	0.8	6.59	6.99
	3D Torus	0.3	15.06	15.06

TABLE I
Comparison of Optimal and 3D Torus Topologies

By analyzing the results, we have the following observations:

(i) Given fixed number of communication demands and patterns, different topologies will be selected with different communication amounts, after the links are congested. We can see the curve is a straight line with low traffic but becomes super linear with the growth of traffic amount. We always prefer a topology with less number of hops and also less detours, since such a topology will incur less router and wire delays. When the traffic increases, however, this topology is no longer feasible since it will cause congestion in some nodes/links. Those topologies with more even link distributions will be chosen.

(ii) The commonly used 3D torus topology has two weaknesses. First, it cannot accommodate large communication traffic. For example, the maximum coefficient is 1.1 with 2048 communications in a 3D torus network while the optimal topology we find can accommodate a coefficient as large as 2. Second, with the same communication coefficient, the latency in a 3D torus is worse than the optimal topology, especially when there are many communications. This is because the torus does not have enough long links, which results in more hops and detours. Thus, it is of importance to look for better alternative topologies.

2) Physical Constraints Impacts: In all the above experiments we define the physical constraints by using the board dimensions, connectors and pin numbers used in Blue Gene/L packaging [7]. According to our analysis, the flows are saturated due to the limited number of pins in the connectors between compute cards and node cards, and between node cards and midplane. In this part of the experiment, we would like to relax this physical constraint and assume we have more pins available for routing.

The case we choose has 2048 communications with demand coefficient 1.6. The previous experiment reports that the average communication latency in midplane is 10.00 with 200 pins (15 connectors) between node cards and midplane. We gradually increase the number of pins and compute the average latency values. The results are shown in Fig. 4. We find that we can reduce the latency by 14% when the pin number is increased to 320; particularly, latency is reduced by 8% if we only add 20 pins, and 13% with only 40 additional pins. This indicates that when the networks are congested, latency can be greatly improved by adding relatively small amount of routing resources. The corresponding topologies for the three points in Fig. 4 are shown in Fig. 5: when pins are limited, long links are essential to reduce the intermediate traffic; with the increase of pin number, the long links are no longer preferred since they will increase the degree of routers therefore add the routing complexity. Hence, using this approach, we are able to know which resources are needed in order to improve the latency, and their marginal impacts

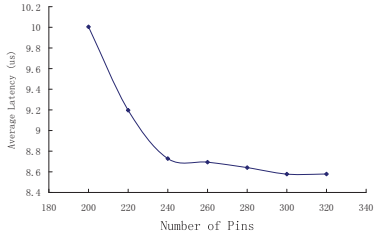


Fig. 4. Average Latency vs. Number of Pins in a Node Card

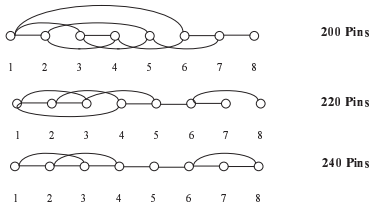


Fig. 5. Optimal Topologies with Different Number of Pins

on the performance.

B. Experiments on Benchmark Instances

We also demonstrate the strengths of our design flow using the NAS parallel benchmarks [5]. We run the class B benchmarks with 121 processes (for BT and SP) or 128 processes (for other six benchmarks). The traffic patterns are then extracted using Intel Trace Analyzer and Collector 7.1 in Linux platform. Among the eight benchmarks, we exclude EP, FT and IS in our experiments, because there are too few communications among the processes.

Before running the experiments, we use a simulated annealing (SA) algorithm to perform the task placement. The SA algorithm optimizes the estimated latency (i.e. estimating the latency by the distance but ignoring the traffic congestion). Then we feed the communication patterns based on the placement result to our design flow. The results are shown in Table II. The latency values are in the unit of us, and the values in the brackets are the latencies normalized to the "Optimal" column. We run our design flow on the five benchmarks in the following three ways.

First, we assume the interconnection network can be customized for each benchmark, therefore obtain the optimal topology for each benchmark. The latency is shown in the "Optimal" column in Table II. Secondly, only one fixed interconnection network is allowed for all the benchmarks. In this case, we aggregate all five traffic patterns by adding them together and feed into the design flow. After the optimal solution for this aggregate traffic pattern is obtained, it will be evaluated for the five instances separately. The latency values are shown in the "Aggregate" column in Table II. Thirdly, we assume the dimension and pin resources are uniformly distributed to all the wires, which is much less flexible design choices than our design flow. The latency values obtained are shown in the "Uniform" column in Table II. We also obtain the latency results of 3D torus structure, which are shown in the last column in Table II.

Table II shows that the optimal topology found by the design flow can achieve much smaller average latency than the 3-D torus. It

Benchmark	Optimal (us)	Aggregate (us)	Uniform (us)	3D Torus (us)
BT	1.30 (1)	1.44(1.11)	1.67(1.29)	2.00(1.54)
CG	0.96 (1)	1.01(1.05)	1.36(1.41)	1.76(1.84)
LU	1.05 (1)	1.38(1.32)	1.60(1.59)	1.69(1.60)
MG	0.89 (1)	0.90(1.02)	1.15(1.29)	1.65(1.83)
SP	2.24 (1)	2.49(1.11)	2.73(1.22)	3.60(1.61)

TABLE II
LATENCY COMPARISON ON NAS PARALLEL BENCHMARKS

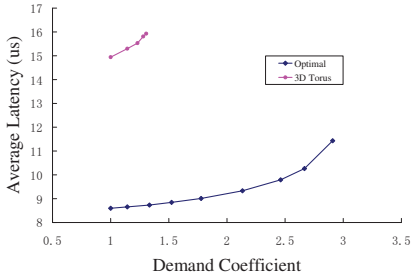


Fig. 6. Latency-Throughput Tradeoff Curves for the Aggregated Benchmark Instance

also indicates that the aggregate traffic pattern is representative, as the optimal topology identified using the aggregate traffic achieves similar latency values to the individual ones. If we compare the latency values in the "Uniform" column with those in the "Optimal" and "Aggregate" columns, we found that it is essential to enable the non-uniform resource allocation for wires, since the flexible choices could accommodate particular traffic patterns and reduce the average latency.

Lastly, to demonstrate the tradeoff between the throughput and latency, we manually increase the communication traffic by multiplying the coefficients to all the communications uniformly. We compute the average latency for the aggregated case. The results are shown in Fig. 6, the two curves represent the latency values for the optimal topologies selected by the design flow, and the 3D torus topology respectively. Similar to the curves in Fig. 3, the curve for optimal topologies is also super linear, which indicates that different optimal topologies are selected with the increase of communication demands. Comparing the two curves, we observe that much lower latencies could be achieved when optimal topologies are used. Also, with the flexibility of the topologies, the system is able to accommodate much larger traffic throughput. This again shows the strength of our design flow over fixed topologies.

VI. CONCLUSION

We propose a design methodology to synthesize the supercomputer interconnection topologies according to the communication patterns, traffic amount, and physical constraints. In the example that we demonstrated, we are able to find different good network topologies on a midplane of the Blue Gene/L supercomputer, each of which has better performance than its original 3D torus. In addition, we are able to correctly identify the bottleneck of the communication, and therefore can provide useful information for designers to further enhance the performance.

REFERENCES

[1] The BlueGene/L Team. An overview of the BlueGene/L supercomputer. In *The 15th Annual SC conference*, 2002.
[2] W. Dally. Performance analysis of k-ary n-cube interconnection networks. *IEEE Trans Comp*, 39(6):775–785, 1990.
[3] W. Dally. Interconnect-centric computing. *Keynote Speech, ISCA*, 2007.
[4] A. Gara et al. Overview of the Blue Gene/L system architecture. *IBM J Res & Dev*, 49(2/3):195–212, 2005.
[5] D. Bailey et al. The NAS parallel benchmarks. *Technical Report NAS-95-020, NASA Ames Research Center*, 1995.
[6] G. Bhanot et al. Optimizing task layout on the Blue Gene/L supercomputer. *IBM J Res & Dev*, 49(2/3):489–500, 2005.
[7] P. Coteus et al. Packaging the Blue Gene/L supercomputer. *IBM J Res & Dev*, 49(2/3):213–248, 2005.
[8] Y. Hu, H. Chen, Y. Zhu, A. Chien, and C.K. Cheng. Physical synthesis of energy-efficient networks-on-chip through topology exploration and wire style optimization. In *ICCD*, pages 111–118, 2005.
[9] Y. Hu, Y. Zhu, H. Chen, R. Graham, and C.K. Cheng. Communication latency aware low power NoC synthesis. In *DAC*, pages 574–579, 2006.
[10] G. Karakostas. Faster approximation schemes for fractional multicommodity flow problems. In *SODA*, pages 166–173, 2002.
[11] L.S. Peh and W. Dally. A delay model for router microarchitectures. *IEEE Micro*, 21(1):26–34, 2001.
[12] S. Scott, D. Abts, J. Kim, and W. Dally. The blackwidow high-radix clos network. In *ISCA*, pages 16–28, 2006.