Parallel Algorithms for Index-Permutation Graphs – An Extension of Cayley Graphs for Multiple Chip-Multiprocessors (MCMP)

Chi-Hsiang Yeh
Dept. of Electrical & Computer Engineering
Queen's University
Kingston, Ontario, K7L 3N6, Canada
yeh@ee.queensu.ca

Behrooz Parhami
Dept. of Electrical & Computer Engineering
University of California
Santa Barbara, CA 93106-9560, USA
parhami@ece.ucsb.edu

Abstract

The index-permutation graph (IPG) model is a natural extension of the Cayley graph model, and super-IPGs form an efficient class of IPGs that contain a wide variety of networks as subclasses. In this paper, we derive a number of efficient algorithms and embeddings for super-IPGs, proving their versatility. We show that a multitude of important networks can also be emulated in super-IPGs with optimal slowdown. Also, the intercluster diameter, average intercluster distance, and bisection bandwidth of suitably constructed super-IPGs are optimal within small constant factors. Finally, we show that when parallel computers, built as multiple chip-multiprocessors (MCMP), are based on super-IPGs, they can significantly outperform those based on hypercubes, k-ary n-cubes, and other networks in carrying out communication-intensive tasks.

1 Introduction

Cayley graphs [3] constitute a useful tool for designing and analyzing symmetric interconnection networks, and have received considerable attention [2, 19, 27, 28]. Recently we proposed *index-permutation graphs (IPGs)*, a natural extension of Cayley graphs, to aid in the systematic development of communication-efficient interconnection networks [27, 30, 34]. We focused on *super-IPGs*, a particularly efficient subclass of IPGs that use identical copies of a small network as their basic modules, and showed that a variety of hierarchical networks belong to the class of super-IPGs. In particular, hierarchical cubic networks (HCN) [15], hierarchical folded-hypercube networks (HFN) [13], hierarchical hypercube networks (HHN) [36], recursively connected complete (RCC) networks [16], hierarchical shuffle-exchange (HSE) networks, shuffle-exchange networks, hierarchical swap networks (HSN) [23, 24, 30], recursive hierarchical swap networks (RHSN) [25], cyclic networks (CN) [26, 31], super-flip networks (SFN) [30, 34], and index-shuffle graphs [6] are all subclasses of super-IPGs. Therefore, the IPG model not only provides new insight into the design of novel communication-efficient networks, but also serves as a framework that ties together a vast variety of previously proposed interconnection topologies.

A main contribution of this paper is to show that super-IPGs significantly outperform other networks such as hypercubes, tori, and k-ary n-cubes in carrying out communication-intensive tasks under the proposed *unit chip capacity model*, which is a reasonable model for parallel and distributed systems built of *multiple chip-multiprocessors (MCMP)*. Examples of such tasks include random routing, fast Fourier transform (FFT), sorting, total exchange, and matrix transposition. We show that for MCMP implementations, super-IPGs lead to lower cost and/or higher performance than competing networks. The lower cost results from the fact that most links are confined to the same chip, thus reducing the number of off-chip links (and thus pins) per node. The higher performance is a direct consequence of the fact that most transmissions are on-chip transmissions (i.e., most of the transmissions are confined within the same chip even for random routing with uniformly distributed destinations) and the relative ease with which a chip can be provided with a wide bisection bandwidth.

We also show that suitably constructed super-IPGs have asymptotically optimal intercluster diameter, average intercluster distance, and bisection bandwidth, which are important parameters for MCMPs, within a small constant factor from their corresponding lower bounds dictated by the intercluster degree. Because the bandwidth limit does not depend on the switching technique used, these advantages occur with packet switching, wormhole routing, virtual cut-through, and so on. In [4, 12, 22], Dally and several other researchers argued that chip-multiprocessors may become the mainstream of computation in a couple of decades. If such a revolution is materialized as envisioned, super-IPGs and their variants will become viable candidates for building parallel systems, given that in an MCMP environment, they achieve the best (and near optimal) performance among all the network topologies proposed in the literature thus far.

Ascend/descend algorithms [20, 21] form an important class of algorithms that includes many computation and communication problems as special cases. Embeddings and emulation of popular topologies are viewed as fast and most cost-effective ways to obtain a variety of algorithms needed on a new network topology. Another main contribution of this paper is to demonstrate the feasibility of developing efficient algorithms and embedding results for the entire class of super-IPGs as a whole. In particular, we show that ascend/descend algorithms, including fast Fourier transform (FFT), sorting, and matrix multiplication, can be performed in several super-IPGs at speeds comparable to or faster than in hypercubes under the unit link capacity model. We also show that a variety of network topologies can be embedded in super-IPGs with constant dilation and asymptotically optimal congestion (from a degree-based lower bound) and/or emulated by super-IPGs with optimal slowdown under both single-dimension and all-port communication models, assuming unit link capacity. The preceding results lead to efficient algorithms for performing multinode broadcast (MNB), total exchange (TE), and many other operations on super-IPGs.

2 The index-permutation graph (IPG) model and super-IPGs

An index-permutation graph (IPG) [27, 30, 34] is defined by a set of *generators* and a *seed element*, where a generator is a permutation. The edges in an IPG correspond to the action of its generators, and the vertices correspond to the elements obtained by applying generators on the seed element or a generated element.

For example, the label of the *seed node* in an IPG can be represented as $Y = y_1y_2y_3y_4y_5y_6 = 123321$, and the generators of the IPG can be permutations

$$\pi_1 = 213456, \quad \pi_2 = 321456, \quad \text{and } \pi_3 = 456123.$$

Then the actions of generators lead to the following 3 neighbors for node Y:

$$Y\pi_1 = \pi_1(Y) = \pi_1(y_1y_2y_3y_4y_5y_6) = y_2y_1y_3y_4y_5y_6 = 213321,$$

$$Y\pi_2 = \pi_2(Y) = \pi_2(y_1y_2y_3y_4y_5y_6) = y_3y_2y_1y_4y_5y_6 = 321321,$$

$$Y\pi_3 = \pi_3(Y) = \pi_3(y_1y_2y_3y_4y_5y_6) = y_4y_5y_6y_1y_2y_3 = 321123.$$

Repeatedly applying the 3 generators to generated nodes will result in 36 distinct nodes for this IPG example.

We have shown in [27, 30, 34] that this simple extension is quite powerful and leads to novel and useful classes of parallel architectures. In what follows, we will focus on super-IPGs, IPGs that use *super-generators*, which are a special class of permutations that interchange two or several sequences of symbols of equal-length. More precisely, a super-IPG is a special class of IPGs where the seed label consists of l identical groups of m symbols and the generators of the IPG can either permute the m symbols within the leftmost group or permute the m-symbol groups without changing the order of symbols within any of the groups. We call each of the m-symbol groups in the label a *super-symbol*. The generators that permute the symbols within the leftmost super-symbol are called *nucleus generators* and the generators that permute super-symbols are called *super-generators*. For example, with the seed label 123 123, the permutation 321 456, which permutes 123 123 to get 321 123, defines a nucleus generator, whereas the permutation 456 123, which permutes 321 123 to get 123 321, corresponds to a super-generator. For each of the super-symbols, there must exist a sequence of super-generators that can bring it to the leftmost position. The smaller IPG whose seed label is a super-symbol of the seed label of a super-IPG and whose generator set consists of all the nucleus generators of the super-IPG is called the *nucleus graph* (or simply *nucleus*) of the super-IPG. Since the nucleus determines the nucleus generators and the seed of the super-IPG, a super-IPG can be completely specified by its super-generators and its nucleus. More details about super-IPG can be found in [27, 30, 34].

In what follows we provide several examples for super-IPGs. An l-level *hierarchical swap network, HSN*(l,G), is a super-IPG that has the seed $\underbrace{S_1S_1\cdots S_1}_{l}$, the generators for the nucleus G, and the *transposition super-generators* $T_{2,m} = (1,2)_m$, $T_{3,m} = (1,3)_m$, $T_{4,m} = (1,4)_m$, $\cdots$, $T_{l,m} = (1,l)_m$, where S_1 is the label of a (seed) node in the nucleus G and $(1,i)_m$ represents the permutation that interchanges the first super-symbol and the i^{th} super-symbol. The subscript m in $(i,j)_m$ denotes the fact that super-symbols of length m at positions i and j are interchanged.

A *ring-cyclic network, ring-CN*(l,G), is defined by nucleus G and *cyclic-shift super-generators* $L_{1,m} = (1 \leftarrow)_m$ and $R_{1,m} = L_{-1,m} = (1 \rightarrow)_m$, where the super-generator $L_{i,m} = (i \leftarrow)_m$ changes the node label $X = X_1X_2\cdots X_l$ into

$$L_{i,m}(X) = X_{i+1}X_{i+2}\cdots X_l\, X_1X_2X_3\cdots X_i$$

and the generator $R_{i,m} = (i \rightarrow)_m$ changes X into

$$R_{i,m}(X) = X_{l-i+1}X_{l-i+2}\cdots X_lX_1X_2X_3\cdots X_{l-i}.$$

A *complete-CN*(l,G) is a super-IPG that is specified by super-generators $L_{1,m}, L_{2,m}, \cdots, L_{l-1,m}$ and its nucleus G.

Another example for the design of super-IPGs is to use *flip super-generators* $F_{i,m}$, where the super-generator $F_{i,m}$ flips the first i super-symbols, $i = 2,3,\ldots,l$ for an IPG with l super-symbols in a node label. For example,

$$F_{2,m}(X_1X_2X_3X_4) = X_2X_1X_3X_4, \; F_{3,m}(X_1X_2X_3X_4) = X_3X_2X_1X_4.$$

A super-IPG with l flip super-generators and nucleus G is called an l-level *super-flip network SFN*(l,G).

3 Parallel algorithms for super-IPGs

Embeddings and emulation between hypercubic networks and for new network topologies have been an intensively studied research area [20]. In this section, we develop algorithms for performing emulation and ascend/descend computation in super-IPGs. We assume the *unit link capacity model* in this section, where all the network links have the same bandwidth and propagation delay. In Section 4, we extend some of the results to the *unit node capacity model* and the *unit chip capacity model*, the latter being particularly suitable for parallel systems based on MCMPs.

3.1 Embeddings and parallel algorithms under the single dimension communication model

In this subsection, we assume the single-dimension communication (SDC) model [28], where the nodes are allowed to use only links of the same dimension at any given time. Many algorithms developed for product networks [14, 35], such as hypercubes and k-ary n-cubes, naturally fall into this category. Ascend/descend [21] and normal [20] algorithms are special cases of the SDC model, which is in turn a special case of the single-port communication model.

Many super-IPGs can emulate a hypercube of the same size with simple and efficient algorithms when they are based on a hypercube as the nucleus. Since each link of an IPG node corresponds to the action of a specific generator, the problem of emulation between IPGs (including hypercubes and k-ary n-cubes) can be transformed to the problem of emulation between their generator sets. For example, the actions of dimension-11 links in a 16-cube, corresponding to the generator $(21,22)$, can be emulated by the sequences of generators

$$T_{2,16} = (1,2)_{16},\ (5,6),\ T_{2,16} = (1,2)_{16} \text{ in an HCN}(8,8),$$

$$T_{3,8} = (1,3)_8,\ (5,6),\ T_{3,8} = (1,3)_8 \text{ in an HSN}(4,Q_4),$$

$$T_{2,16} = (1,2)_{16},\ (5,6),\ T_{2,16} = (1,2)_{16} \text{ in an RCC}(2,Q_4),$$

$$R_{1,8}R_{1,8},\ (5,6),\ L_{1,8}L_{1,8} \text{ in a ring-CN}(4,Q_4),$$

$$R_{2,8} = (\rightarrow 2)_8,\ (5,6),\ L_{2,8} = (\leftarrow 2)_8 \text{ in a complete-CN}(4,Q_4),$$

assuming the 32-symbol seed 01 01 01 $\cdots$ 01.

Homogeneous product networks (HPNs) form a subclass of product networks with identical component networks [14]. More precisely, an HPN is the iterated Cartesian product of the same graph, and an HPN(p,G) denotes the p^{th} *power of* G; that is, HPN$(p,G) = \prod_{j=1}^{p} G = \underbrace{G \times G \times \cdots \times G}_{p}$.

It can be seen that the pk-dimensional (binary) hypercube is the p^{th} power of a k-cube; the p-dimensional (generalized) hypercube of radix $M \geq 2$ [8, 18] is the p^{th} power of an M-node complete graph, and the M-ary p-cube is the p^{th} power of an M-node ring.

Let d_G be the node degree of a *dimensionizable* graph G. Each link of a node in G is given a distinct integer $i \in [1, d_G]$ as its label, and is called the dimension-i link. Then the dimension-$(i + d_G(p-1))$ links of the homogeneous product network $G_p \times G_{p-1} \times \cdots \times G_1$ with $G_j = G$ are the collection of dimension-i links of its factor graphs G_j.

The emulation algorithm for all super-IPGs are similar as shown in the following theorem.

Theorem 3.1 *Let t be the number of applications of the super-generators of a super-IPG required to bring any super-symbol of a node label to the leftmost position and then to restore the super-symbols to the original order. Then the super-IPG can emulate any algorithm in an HPN(l,G) under the SDC model with a slowdown factor of $t+1$, where l is the number of super-symbols in a node label and G is the nucleus of the super-IPG.*

Proof: Let $j_0 = 1 + (j - 1 \bmod n)$ and $j_1 = 1 + \lfloor (j-1)/n \rfloor$, where n is the number of nucleus generators for the nucleus G. Then to emulate the j^{th} generator (or transmissions along the dimension-j links) in an HPN(l,G), we first use the super-generators of the super-IPG to bring the j_1^{th} super-symbol to the leftmost position, then apply the j_0^{th} nucleus generator (or transmissions along the dimension-j_0 links of each nucleus G), and finally use the super-generators to rearrange the super-symbols to their original order, where $j_0 = 1 + (j - 1 \bmod n)$, $j_1 = 1 + \lfloor (j-1)/n \rfloor$, and n is the number of nucleus generators for the nucleus G. □

In other words, if an algorithm require T communication steps in an HPN(l,G) using store-and-forward packet switching under the SDC model, then it can be emulated in the corresponding super-IPG using $T(t+1)$ communication steps. The dilation to embed an HPN(l,G) in the above super-IPG is also equal to $t+1$ since it must be equal to or smaller than the slowdown factor for single-dimension emulation. Note that the slowdown factor for single-dimension emulation may be larger than the dilation for embeddings in general cases for emulation between other networks. For several super-IPGs we have $t = 2$, leading to the following corollaries.

Corollary 3.2 *Any algorithm in an HPN(l,G) under the SDC model can be emulated on an HSN(l,G), complete-CN(l,G), or SFN(l,G) with a slowdown factor of 3, under the SDC model.*

Proof: It follows from Theorem 3.1 and the fact that the super-generators of any of these super-IPGs can bring any super-symbol to the leftmost position in one step and then to restore the original order of the super-symbols in another step. □

Corollary 3.3 *An HPN(l,G) can be embedded on an HSN(l,G), complete-CN(l,G), or SFN(l,G) with dilation 3.*

When wormhole routing or virtual cut-through is used, the slowdown factor is actually reduced to about 2, since the congestion for embedding all the links of an HPN(l,G) that belong to a certain dimension in an HSN(l,G), complete-CN(l,G), or SFN(l,G) is only 2. Similarly, when store-and-forward packet switching is used but two neighboring nodes transmit a nonconstant number of packets on the average during a step of the HPN algorithm, then the slowdown factor for emulating the HPN algorithm in an HSN(l,G) or complete-CN(l,G) is $2 + o(1)$, rather than 3. Note also that if a link (especially an off-chip link) in the HSN(l,G) or complete-CN(l,G) has bandwidth higher than that of an HPN link by a factor of f, then the HSN or complete-CN can in fact perform the algorithm faster than the HPN by a factor of about $f/2$ when wormhole routing or virtual cut-through is used or when packet switching is used but two neighboring nodes transmit many packets during a step on the average. More details will be given in Section 4.

Corollary 3.4 *If a graph can be embedded in an ln-dimensional hypercube with constant dilation, then the* *graph can be embedded with constant dilation in an HCN($ln/2, ln/2$), HFN($ln/2, ln/2$), HSN(l, Q_n), complete-CN(l, Q_n), SFN(l, Q_n), RCC(r, Q_n) with constant r, or an r-deep RHSN based on Q_n.*

Proof: These super-IPGs can emulate an ln-dimensional hypercube under the SDC model with constant slowdown from Corollary 3.3 and the result follows. □

There exist a large variety of networks that can be embedded in a hypercube with constant dilation. Therefore, we can obtain a variety of constant-dilation embeddings in the above super-IPGs using Corollary 3.4. These results can be easily generalized to other super-IPGs.

3.2 Ascend/descend algorithms

When the dimensions of the links used by an SDC algorithm in an HPN are consecutive, the algorithm can be emulated even more efficiently on a corresponding super-IPG, often with a slowdown factor close to 1. In such a case, we do not need to use super-generators to rearrange the super-symbols to their original order for intermediate steps of the emulated algorithm. For example, consider the emulation of the j^{th} and then the $(j+1)^{th}$ generators in an HPN(l, G). Let $k_0 = 1 + (j \bmod n)$ and $k_1 = 1 + \lfloor j/n \rfloor$. We first use the super-generators to bring the j_1^{th} super-symbol to the leftmost position, apply the j_0^{th} nucleus generator, and then use the super-generators of the super-IPG to bring the k_1^{th} super-symbol to the leftmost position if $k_1 \neq j_1$, apply the k_0^{th} nucleus generator, and finally use the super-generators to rearrange the super-symbols to their original order.

Ascend/descend algorithms [20, 21] require successive operations on data items that are separated by a distance equal to a power of 2, and are a special case of the preceding class of algorithms. Many applications, such as Fast Fourier Transform (FFT), bitonic sort, matrix multiplication, and convolution, can be formulated using algorithms in this general category. In what follows we present general algorithms for performing ascend/descend computations on a super-IPG.

Theorem 3.5 *Let t_r be the number of applications of the super-generators of a super-IPG required in order for the 2nd, 3rd, 4th, ... , l^{th} (or the l^{th}, $(l-1)^{th}$, ... , 2nd, 1st) super-symbols of a node label to appear at the leftmost position and then for the super-symbols to be restored to the original order. Then the super-IPG can perform an ascend algorithm (or descend algorithm, respectively) for all the $\log_2 N$ possible operations in $lt_G + t_r$ time, where t_G is the time required to perform an ascend algorithm (or descend algorithm, respectively) in nucleus G, assuming that the size of the nucleus G of the IPG is a power of 2.*

Proof: Let M be the number of nodes in nucleus G. We first perform an ascend algorithm in G, which corresponds to the first $\log_2 M$ ascend operations in the super-IPG. For $i = 2, 3, 4, \ldots, l$, we bring the i^{th} super-symbol to the leftmost position using the super-generators of the super-IPG, which will bring nodes whose addresses are separated by M^{i-1}, $2M^{i-1}$, $4M^{i-1}$, ..., $M^i/2$ into the same nucleus, and then perform an ascend algorithm in G, which corresponds to the $((i-1)\log_2 M + 1)^{th}$ through the $(i \log_2 M)^{th}$ ascend steps in the super-IPG. Finally, we use its super-generators to restore the order of super-symbols, which delivers the results to the appropriate destinations. Since we need t_r steps to rearrange the order of super-symbols during the whole process, and we perform an ascend algorithm in all the nuclei l times, an ascend algorithm requires $lt_G + t_r$ time in the super-IPG.

The algorithm and analysis for performing a descend algorithm in the super-IPG are similar to those for the ascend algorithm. □

CNs are particularly efficient for ascend/descend and normal hypercube algorithms. Since only one final step is required to restore the super-symbols to the original order for performing an ascend algorithm in any CN (for several other super-IPGs, $l-1$ such rearrangements of the super-symbols are needed), we obtain the following corollary to Theorem 3.5.

Corollary 3.6 *Ascend/descend algorithms (for all the $\log_2 N$ possible operations) can be performed in $l(k+1) = (1+1/k)\log_2 N$ time on a CN based on a k-cube, and in $l(k+2) - 2 = (1+2/k)\log_2 N - 2$ time on an HSN, SFN, RCC, or RHSN based on a k-cube, where l is the number of super-symbols in the label of a node.*

Corollary 3.7 *Ascend/descend algorithms (for all the $\log_2 N$ possible operations) can be performed in $l(n+1)$ communication steps and $l\sum_{i=1}^{n}(m_i - 1)$ computation steps on a CN based on an n-dimensional generalized hypercube with mixed-radix $(m_1, m_2, \ldots, m_n)$ and $l(n+2) - 2$ communication steps and $l\sum_{i=1}^{n}(m_i - 1)$ computation steps on an HSN, RCC, or RHSN based on an n-dimensional generalized hypercube with mixed-radix $(m_1, m_2, \ldots, m_n)$, where l is the number of super-symbols in the label of a node and m_i is a power of 2 for all i.*

For example, when $m_i = 4$ and $n = 3$, ascend/descend algorithms can be performed in $\frac{2}{3}\log_2 N$ communication steps on an N-node CN and in $\frac{5}{6}\log_2 N - 2$ communication steps on on an HSN, RCC, or RHSN. Note that these networks can perform ascend/descend algorithms faster than hypercubes (in terms of communication steps) and require smaller node degrees at the same time. Note that if reordering of the results is not required, then the number of communication steps can be further reduced.

3.3 Parallel algorithms under the all-port communication model

We now consider the all-port communication model, where a node is allowed to use all its incident links for packet transmission and reception at the same time. The packets transmitted on different outgoing links of a node can be different.

The following theorem shows that suitably constructed super-IPGs can emulate a corresponding HPN of the same size with asymptotically optimal slowdown, given their node degrees.

Theorem 3.8 *Any algorithm in an HPN(l,G) with all-port communication can be emulated on an HSN(l,G), complete-CN(l,G), or SFN(l,G) with a slowdown factor of max$(2n, l+1)$, where n is the number of generators for the nucleus G.*

Proof: In Theorem 3.1, we have presented the algorithm to emulate an HPN under the SDC model. The emulation algorithm with all-port communication simply performs single-dimension emulation for all dimensions at the same time with proper scheduling to minimize the congestion. Let S_{j_1} be the super-generator that brings the j_1^{th} super-symbol to the leftmost position, $S_{j_1}^{-1}$ be the inverse generator of S_{j_1} (i.e., $S_{j_1}S_{j_1}^{-1} = I$), and N_{j_0} be the j_0^{th} nucleus generator. A packet for a dimension-j neighbor, $j > n$, in the emulated HPN will be sent through links $S_{j_1}, N_{j_0}, S_{j_1}^{-1}$, where $j_0 = 1 + (j-1 \bmod n)$ and $j_1 = 1 + \lfloor (j-1)/n \rfloor$. There exist several schedules for these links that guarantee the desired slowdown factor. A possible schedule can be obtained as follows.

We first consider the special case where $l = rn + 1$ for some positive integer r.

- At time 1, each node sends the packets for its dimension-j neighbors (in the emulated HPN), $j = 1, 2, 3, \ldots, n$, through links N_j.
- At time t, $t = 1, 2, 3, \ldots, n$, each node sends the packets for its dimension-$u_i(t)$ neighbors, $i = 2, 3, 4, \ldots, l$, through links S_i, where $u_i(t) = (i-1)n + 1 + (i + t - 3 \bmod n)$.
- At time t, $t = sn+2, sn+3, sn+4, \ldots, (s+1)n+1$ for $s = 0, 1, 2, \ldots, r-1$, each node forwards the packets for dimension-$v_i(t)$ neighbors, $i = sn+2, sn+3, sn+4, \ldots, (s+1)n+1$, through links $N_{v_i(t)}$, where $v_i(t) = (i-1)n + 1 + (i + t - 2sn - 4 \bmod n)$.
- At time t, $t = n+1, n+2, \ldots, 2n$, each node forwards the packets for its dimension-$u_i(t)$ neighbors, $i = 2, 3, 4, \ldots, n+1$, through links S_i^{-1}, where $u_i(t) = (i-1)n + 1 + (i + t - n - 3 \bmod n)$. At time t, $t = sn+3, sn+4, sn+5, \ldots, (s+1)n+2$ for $s = 1, 2, 3, \ldots, r-1$, each node forwards the packets for dimension-$u_i(t)$ neighbors, $i = sn+2, sn+3, sn+4, \ldots, (s+1)n+1$, through links S_i^{-1}, where $u_i(t) = (i-1)n + 1 + (i + t - 2sn - 5 \bmod n)$.

Figure 1a shows such a schedule for emulating a 12-dimensional HPN on a super-IPG with $l = 4$ and $n = 3$.

In what follows we extend the previous schedule to the general case where l is not of the form $l = rn + 1$. The schedule for $l \leq n$ can be easily obtained by removing the unused part of the schedule for a super-IPG with $l = n + 1$. Other possible cases can be formulated by assuming that $l = rn - w$ for some integers $r \geq 2$ and $0 \leq w \leq n-2$, in which case we can modify the schedule as follows. We initially start with the schedule for a super-IPG that has $rn + 1$ super-symbols in the label of a node. Clearly, the transmissions in the schedule that correspond to the emulation of dimensions $j > ln$ are not used by the super-IPG with $l = rn - w$. Therefore, we can now perform each of the transmissions over links N_{j_0} originally scheduled for time $l+1$ through $rn+1$ at time earlier than $l+1$ by rescheduling these transmissions to the unused part of the schedule. Note that the modified part of the schedule are for the emulation of some dimensions larger than $(r-1)n^2 + n$ (that is, some of the dimensions that correspond to the last $l - (r-1)n - 1 = n - w - 1$ super-symbols). We then swap generators N_{j_0} in the modified part of the schedule with part of the schedule for the emulation of dimensions smaller than $(r-1)n^2 + n + 1$ (that is, for some of the dimensions that correspond to the first $(r-1)n + 1$ super-symbols). Due to the previous modifications, we also have to move the schedule for some generators S_{j_1} and $S_{j_1}^{-1}$. In particular, we will move generator $S_{j_1}^{-1}$ in each of the 3-step single-dimension emulations one time step after the use of N_{j_0} generators when possible. When $l + 1 < 2n$, the schedule for some generators $S_{j_1+1}^{-1}$ can not be moved before time $2n$. As a result, the time required for emulation under the all-port communication model is equal to $l+1$ if $l+1 \geq 2n$, and is equal to $2n$ otherwise. Figure 1b shows such a schedule for emulating a 15 dimensional HPN

Generators of a super-IP graph	Dimension j of the HPN being emulated											
	1	2	3	4	5	6	7	8	9	10	11	12
Step 1	N_1	N_2	N_3	S_2	—	—	—	S_3	—	—	—	S_4
Step 2				N_1	S_2	—	—	N_2	S_3	S_4	—	N_3
Step 3				—	N_2	S_2	S_3	—	N_3	N_1	S_4	—
Step 4				S_2^{-1}	—	N_3	N_1	S_3^{-1}	—	—	N_2	S_4^{-1}
Step 5					S_2^{-1}	—	—		S_3^{-1}	S_4^{-1}	—	
Step 6						S_2^{-1}	S_3^{-1}				S_4^{-1}	

(a)

Generators of a super-IP graph	Dimension j of the HPN being emulated														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Step 1	N_1	N_2	N_3	S_2	—	—	—	S_3	—	—	—	S_4	—	S_5	—
Step 2				N_1	S_2	—	—	—	S_3	S_4	—	N_3	—	N_2	S_5
Step 3				—	N_2	S_2	S_3	—	N_3	N_1	S_4	—	S_5	—	—
Step 4				S_2^{-1}	—	—	N_1	—	S_3^{-1}	—	N_2	S_4^{-1}	—	S_5^{-1}	N_3
Step 5					S_2^{-1}	N_3	S_3^{-1}	N_2		S_4^{-1}	—		N_1		S_5^{-1}
Step 6						S_2^{-1}		S_3^{-1}			S_4^{-1}		S_5^{-1}		

(b)

Figure 1. Schedules for emulating HPNs on super-IPGs under the all-port communication model. Note that a generator appears at most once in a row, and each column $j > 4$ consists of generators $S_{j_1}, N_{j_0}, S_{j_1}^{-1}$, where $j_0 = 1 + (j - 1 \bmod 3)$ and $j_1 = 1 + \lfloor (j-1)/3 \rfloor$. (a) Emulating a 12-dimensional HPN$(4, G)$ on an super-IPG with $l = 4$ and $n = 3$. (b) Emulating a 15-dimensional HPN$(5, G)$ on an super-IPG with $l = 5$ and $n = 3$. The links in the super-IPG are fully used during steps 1 to 5, and are 93% used on the average.

on a super-IPG with $l = 5$ and $n = 3$. □

By properly choosing parameters l and n, we can emulate an HPN with all-port communication on these super-IPGs with asymptotically optimal slowdown with respect to the node degrees.

Corollary 3.9 *Any algorithm in an HPN(l,G) with all-port communication can be emulated on an HSN(l,G), complete-CN(l,G), or SFN(l,G) with asymptotically optimal slowdown if $l = \Theta(n)$, where n is the number of generators for the nucleus G.*

Proof: It follows from Theorem 3.8 and the fact that a graph of degree $O(n)$ cannot emulate a graph of degree $\Omega(n^2)$ with a slowdown factor smaller than $O(n)$ under the all-port communication model. □

Two basic communication tasks that arise often in applications are the multinode broadcast (MNB) and the total exchange (TE) [7, 17]. In the MNB each node has to broadcast a packet to all the other nodes of the network, while in the TE each node has to send a different (personalized) packet to every other node of the network. Using Theorem 3.8 and optimal algorithms for hypercubes, we can obtain asymptotically optimal algorithms for several super-IPGs.

Corollary 3.10 *The multinode broadcast task can be performed in asymptotically optimal time $\Theta\left(N/\sqrt{\log N}\right)$ in an HSN, complete-CN, and SFN based on a hypercube if the degree of the network is $\Theta\left(\sqrt{\log N}\right)$.*

Proof: When the degrees of these networks are $\Theta\left(\sqrt{\log N}\right)$, we have $l = \Theta(n)$, where n is the dimension of the nucleus hypercube. The completion time on the above super-IPGs can be obtained from Theorem 3.8 and the fact that MNB can be performed in $\Theta\left(N/\log N\right)$ time in a $(\log_2 N)$-cube. □

Corollary 3.11 *The total exchange task can be performed in asymptotically optimal time $\Theta\left(N\sqrt{\log N}\right)$ in an HSN, complete-CN, and SFN based on a hypercube if the degree of the network is $\Theta\left(\sqrt{\log N}\right)$.*

Proof: The completion time on the above networks can be obtained from Theorem 3.8 and the fact that TE can be performed in $\Theta(N)$ time in a $\lfloor \log_2 N \rfloor$-cube. □

Note that by using denser graphs for the nucleus, super-IPGs can execute total exchange in even shorter time. Also, an algorithm obtained from emulating a TE algorithm developed for hypercubes will contain some redundant steps that send packets back and forth along intercluster links. By removing these steps, the number of intercluster transmissions for a TE tasks can be reduced to $\Theta(N^2)$ in super-IPGs which is smaller than the $\Theta(N^2 \log N)$ intercluster transmissions required in hypercubes. Therefore, when each of the nuclei or clusters is contained on the same chip and off-chip transmissions form the performance bottleneck, super-IPGs can outperform hypercubes by a factor of $\Theta(\log N)$ for executing TE tasks. More details will be given in Section 4.

4 Properties of super-IPGs for MCMPs

A parallel computer is typically built from several chips on a board, multiple boards in a card-cage and several such card-cages interconnected together. Modules at each level of the packaging hierarchy have their respective characteristics in terms of the number of pins, maximum area/volume, minimum wire width, and the number of wires per link [5]. Since on-chip links are significantly shorter than off-chip links and do not require the extra delay associated with driving off-chip pins, they may be driven at a considerably higher clock rate. Moreover, since the cost for an on-chip connection is much smaller than that of an off-chip connection, which is limited by the number of pin-outs, the channel width of an on-chip link can be increased, if required, without significantly increasing the hardware cost.

In this section, we consider the case where several nodes (processors, routers, and associated memory banks) of a network are implemented on a single chip, or more generally, a single module (e.g., chip, board, wafer, or multi-chip module (MCM)), and several chips are used to build the parallel architecture or a higher-level module. We refer to such an architecture as *multiple chip-multiprocessors (MCMP)*. In order to compare the performance of different networks, we propose the *unit chip capacity model*, where the sum of the bandwidth of all the off-chip links of a chip is fixed. For example, a 16-node cluster of a 12-dimensional hypercube has 128 intercluster links, while a 16-node cluster of an HSN$(3,Q_4)$ has 30 intercluster links, so an off-chip link of the HSN$(3,Q_4)$ has bandwidth about 4 times higher than that of an off-chip link of the 12-cube when a chip can accommodate 16 nodes. We assume that on-chip links can be made fast enough so that they do not form performance bottleneck.

4.1 Embeddings and parallel algorithms

As shown in Section 3, an HSN, complete-CN, or SFN can embed corresponding homogeneous product networks [14, 35], such as hypercubes or k-ary n-cubes, with dilation 3 and asymptotically optimal congestion (from a degree-based lower bound). More precisely, when $l = \Theta(\sqrt{\log N}) = \Theta(n)$, the degree of an HSN$(l,Q_n)$, complete-CN$(l,Q_n)$, or SFN$(l,Q_n)$ is $\Theta(\sqrt{\log N})$ and the congestion for embedding an nl-dimensional hypercube is $\Theta(\sqrt{\log N})$, which is the smallest possible for a degree-$\Theta(\sqrt{\log N})$ network to embed a degree-$\log_2 N$ network.

The *intercluster degree* of a network, defined as the maximum of the average-per-node intercluster links over all clusters, has a bearing on its implementation cost and communication performance. The intercluster degree of a hypercube is $\log_2 N - o(\log N)$ and the intercluster degree of an HSN(l,Q_n), complete-CN(l,Q_n), or SFN(l,Q_n) is $l-1 = \Theta(\sqrt{\log N})$, so a $\Theta(\sqrt{\log N})$ lower bound on the congestion can also be derived based on their intercluster degrees.

Note that a nonconstant congestion for the embeddings does not imply worse performance for these super-IPGs at the hardware level since a network link in an HSN(l, Q_n), complete-CN(l, Q_n), or SFN(l, Q_n) can be made considerably wider (i.e., containing more wires in parallel) than a link in an nl-dimensional hypercube. One of the reasons is that when a node in an HSN(l, Q_n), complete-CN(l, Q_n), or SFN(l, Q_n) has the same capacity as a node in an nl-dimensional hypercube (i.e., assuming unit node capacity), then a link in the super-IPG will have bandwidth higher than that of a link in an nl-cube by a factor of $\Theta(\sqrt{\log N})$. Another reason is that if off-chip bandwidth is the limiting factor so that a node in an HSN(l, Q_n), complete-CN(l, Q_n), or SFN(l, Q_n) has the same off-chip capacity as a node in an nl-dimensional hypercube (i.e., assuming the unit chip capacity model), then an off-chip link in the super-IPG will also have bandwidth higher than that of an off-chip link in an nl-cube by a factor of $\Theta(\sqrt{\log N})$. Therefore, even if simple emulation of hypercube algorithms is used, the performance of HSNs will be comparable to that of hypercubes under the all-port communication model (within a constant factor) if they are both built as MCMPs (with unit chip capacity). This is in contrast to the slowdown factor of $\Theta(\sqrt{\log N})$ under the all-port communication model with unit link capacity. On the other hand, when emulating normal hypercube algorithms or algorithms developed for hypercubes under the single-dimension communication model, surprisingly, the performance of HSNs, complete-CNs, and SFNs will be better than that of hypercubes by a factor of $\Theta(\sqrt{\log N})$ under the unit chip capacity model.

When algorithms (such as random routing, FFT, sorting, and other ascend/descend algorithms) derived directly for HSNs, complete-CNs, and SFNs are executed, the performance of these super-IPGs will also be better than that of hypercubes by a factor of $\Theta(\sqrt{\log N})$ in terms of the maximum achievable throughput when off-chip bandwidth is the limiting factor. The reason is that random routing or FFT in a hypercube requires $\log_2 N - o(\log N)$ off-chip transmissions, while random routing or FFT in an HSN(l, Q_n), complete-CN(l, Q_n), or SFN(l, Q_n) only requires $l - 1 = \Theta(\sqrt{\log N})$ off-chip transmissions. Another way to look at the problem is that an N-node HSN has bisection width $N/4$ when the nucleus graph is dense enough, and an HSN with $\Theta(\sqrt{\log N})$-times wider off-chip links will have bisection bandwidth higher than that of a hypercube by a factor of $\Theta(\sqrt{\log N})$, so HSNs can execute communication-intensive tasks, such as FFT, sorting, and total exchange with considerably higher throughput when they are both built as MCMPs. Note that when the unit link capacity model is assumed, HSNs, complete-CNs, SFNs, and hypercubes have comparable throughput for these communication-intensive tasks (usually within a factor of $1 + o(1)$ or $2 + o(1)$).

When a chip can accommodate more processors and network nodes so that the number l of super-symbols is small, the superiority of HSNs, complete-CNs, and SFNs is even more pronounced for MCMP implementations. As an example, when $l = O(1)$, the throughput of an HSN, complete-CN, or SFN for executing communication-intensive tasks such as FFT, sorting, and total exchange will be higher than that of a hypercube by a factor of $\Theta(\log N)$. More details concerning the bisection bandwidths of super-IPGs, hypercubes, and k-ary n-cubes will be given in the next subsection. Similar arguments are applicable to other super-IPGs whose super-generators can move any super-symbol to the first position and back to its original position using a constant number of steps.

When off-chip bandwidth is the limiting factor on performance, which is expected to be a typical situation for MCMPs, the maximum throughput for executing a task in a network is inversely proportional to the number of off-chip transmissions, assuming that traffic is balanced among all off-chip links. The latter is an important criterion for designing efficient algorithms for MCMPs. In the following subsection, we calculate such parameters for random routing in super-IPGs.

4.2 Intercluster diameter, average intercluster distance, and bisection bandwidth

We define the *intercluster distance* between a pair of nodes as the minimum number of intercluster transmissions required for routing between them, the *intercluster diameter* of a network as the maximum of the intercluster distance between any pair of nodes, and the *average intercluster distance* of a network as the average of the intercluster distances between all pairs of nodes. The average intercluster distance of a network is upper bounded by its intercluster diameter. The proofs for the following theorems and corollaries are similar to those in [27, 30, 34] and are omitted in this paper.

Theorem 4.1 *Let t be the minimum number of applications of the super-generators of a super-IPG in order for each super-symbol to appear at the leftmost position at least once. Then the intercluster diameter of the super-IPG is t, assuming that each cluster is composed of a nucleus graph.*

Corollary 4.2 *Assuming that each cluster contains a nucleus graph, the intercluster diameter of an N-node HSN, RHSN, RCC, CN, directed CN [26, 27], or SFN is*

$$l - 1 = \log_M N - 1,$$

where l is the number of super-symbols and M is the number of nodes in the nucleus graph.

Theorem 4.3 *Let t_S be the minimum number of applications of the super-generators of a symmetric super-IPG [27, 30, 34] in order for each super-symbol to appear at the leftmost position at least once and then for the super-symbols to be arranged to any possible order. Then the intercluster diameter of the symmetric super-IPG is t_S, assuming that each cluster is composed of a nucleus graph.*

Note that t_S of a symmetric super-IPG is usually larger than t of Theorem 4.1 for the corresponding super-IPG.

Corollary 4.4 *The intercluster diameter of a symmetric complete-CN(l, G) is l, the intercluster diameter of a symmetric HSN(l, G), a symmetric SFN(l, G), a symmetric*

RCC, or a symmetric RHSN is $2l-2$, *and the intercluster diameters of a symmetric ring-CN*$(2,G)$*, a symmetric ring-CN*$(3,G)$*, and a symmetric ring-CN*(l,G)*,* $l \geq 4$*, are* 2, 3, *and* $\lfloor 1.5l \rfloor - 2$*, respectively, assuming that each cluster is composed of a nucleus graph.*

The proofs for the following theorems are similar to those in [27, 28, 34] for proving that macro-star networks and super-IPGs have asymptotically optimal diameter and average distance and are omitted in this paper.

Theorem 4.5 *Assuming that a cluster is composed of exactly one nucleus, the intercluster diameter (or average intercluster distance) of an HSN, CN, directed CN, SFN, RCC, or RHSN, is asymptotically optimal within a factor of* $1+o(1)$ *from the corresponding lower bound given its intercluster degree if* $\log l = o(\log M)$*, where* l *is the number of super-generators,* M *is the number of nodes in a nucleus. The intercluster diameter (or average intercluster distance) of an HSN, CN, directed CN, SFN, RCC, or RHSN, is asymptotically optimal within a constant factor from the corresponding lower bound given its intercluster degree if* $\log l = \Theta(\log M)$.

Theorem 4.6 *The intercluster diameter (or average intercluster distance) of a symmetric HSN or symmetric SFN is asymptotically optimal within a constant factor from the corresponding lower bound given its intercluster degree for any combination of* l *and* M*. The intercluster diameter (or average intercluster distance) of a symmetric CN, symmetric directed CN, symmetric RCC, or symmetric RHSN, is asymptotically optimal within a constant factor from the corresponding lower bound given its intercluster degree if* $\log l = \Theta(\log M)$.

A set B_i of links is a bisection of a network if the removal of B_i partitions the network into two parts that differ in size by at most one node. The *bisection width* W_B of a network is the minimum number of links among all bisections B_i of the network. Let $B_{i,1}, B_{i,2}, B_{i,3}, \ldots, B_{i,L_i}$ denote the bandwidths of links in partition B_i. Then the *bisection bandwidth* B_B of a network is given by

$$B_B = \min_{\text{all } i}\left(\sum_{j=1}^{L_i} B_{i,j}\right)$$

for all possible bisections i of the network. If the bandwidth of each network link is B_0, then the bisection bandwidth of the network is given by $B_B = W_B \times B_0$.

Bisection bandwidth of a network is usually the limiting factor on the performance of communication-intensive tasks, such as random routing, sorting, FFT, and total exchange. When the unit link capacity model is used, the bisection bandwidth of a network is equal to its bisection width; when the unit bisection capacity model is used (similar to the one proposed by Dally in [11], which is a reasonable model for *single chip-multiprocessors (SCMP)*), the bisection bandwidths of networks are the same regardless their bisection widths; however, when the unit node capacity model or the unit chip capacity model is used, the bisection bandwidth of a network needs to be computed according to the capacity of a link and is critical to the performance of the network. In what follows we calculate the bisection bandwidth of super-IPGs, assuming that each chip holds a single nucleus and on-chip links are made wide enough so that the bisection bandwidth is derived without removing any on-chip links (since the removal of any such wide on-chip link would increase the sum considerably). We will show that super-IPGs can have bisection bandwidth considerably higher than hypercubes and k-ary n-cubes.

Theorem 4.7 *If the traffic for random unicast routing can be uniformly distributed among all off-chip links of a network, then a lower bound on the bisection bandwidth* B_B *of the network can be found by*

$$B_B \geq \frac{wN}{4a},$$

where w *is the average aggregate off-chip bandwidth of a node and* a *is the average intercluster distance (for random unicast routing with balanced traffic), assuming that each chip holds a single cluster.*

Note that the average intercluster distance is the average of the distances between a node X and all the network nodes (including node X itself), averaged over all network nodes X. For example, the average intercluster distance of a 12-cube is exactly 4 when a cluster has 16 nodes. The same distance would be slightly larger than 4 when the distance of a node to itself is excluded from the calculation.

An upper bound on bisection bandwidth can usually be easily derived by partitioning the chips into two equal groups. The following corollary indicates that the lower bound is tight, at least for some super-IPGs.

Corollary 4.8 *The bisection bandwidth of an* N*-node HSN or SFN is*

$$B_B = \frac{wNM}{4(l-1)(M-1)} = \frac{wNM}{4(\log_M N - 1)(M-1)},$$

where w *is the average aggregate off-chip bandwidth of a node, assuming that each chip holds an* M*-node nucleus graph.*

Corollary 4.9 *The bisection bandwidth of an* N*-node hypercube, CCC, or butterfly networks is* $\frac{wN}{2(\log_2 N - \log_2 M)}$, $\Theta\left(\frac{wN}{\log N}\right)$*, or* $\Theta\left(\frac{wN}{\log_M N}\right)$*, respectively, where* w *is the average aggregate off-chip bandwidth of a node and* M *is the number of nodes on a chip.*

Proof: The bisection width of an N-node hypercube is $N/2$, and a node has $\log_2 N - \log_2 M$ off-chip links, each of which has bandwidth $\Theta(w/\log N)$, so the bisection bandwidth is $\frac{wN}{2(\log_2 N - \log_2 M)}$. The bisection width of an N-node CCC is $\Theta\left(\frac{N}{\log N}\right)$, and a node only has a constant number of off-chip links, each of which has bandwidth $\Theta(w)$, so the bisection bandwidth is $\Theta\left(\frac{wN}{\log N}\right)$, which is of the same order as that of a similar-size hypercube. The bisection width of an N-node butterfly is $\Theta\left(\frac{N}{\log N}\right)$. By using the partitioning that we proposed in [32], the intercluster degree of a butterfly node is only $\Theta\left(\frac{\log_M N}{\log N}\right)$. Therefore, the bisection bandwidth of the butterfly network is $\Theta\left(\frac{wN}{\log_M N}\right)$, which is higher than that of a similar-size hypercube. □

When $M = N^{\varepsilon}$, where ε is a constant smaller than 1, we have $l = \log_M N = O(1)$. In this case, the bisection bandwidths of l-level HSNs, complete-CNs, SFNs, butterfly networks, and several other super-IPGs become $\Theta(wN)$. To the best of our knowledge, these networks and their variants, along with the expander graphs (e.g., the AKS sorting networks [1]), are the only networks that are known to have bisection bandwidth $\Theta(wN)$. Note that $l-1$ is usually a small integer like 1, 2, or at most 3 so the number $\frac{(l-1)(M-1)}{M}$ of off-chip links per node is small. Therefore, the leading constants for the bisection bandwidths of HSNs and SFNs are the largest among these networks, achieving the best performance for communication-intensive tasks.

The bisection bandwidths of different-size hypercubes are the same when the same number of chips are used in these MCMPs. The reason is that if a chip contains fewer hypercube nodes, the off-chip bandwidth per link is increased so that the bisection bandwidth remain the same. For example, a 12-cube with 16-node chips (for a total of 256 chips) has off-chip bandwidth $w/8$ per link and has bisection width 2048 and bisection bandwidth $256w$; while a 10-cube with 4-node chips (for a total of 256 chips too) has off-chip bandwidth $w/2$ per link and has bisection width 512 and bisection bandwidth $256w$. As a comparison, an HSN$(3, Q_4)$ with 16-node chips (for a total of 256 chips) has off-chip bandwidth $8w/15$ per link, has bisection width 1024 (without cutting any nucleus), and has bisection bandwidth $8192w/15 > 512w$, which is slightly more than double that of a hypercube with the same number of chips. Therefore, for communication-intensive tasks, the HSN has the potential of offering a throughput that is double that of a hypercube, when they are both implemented as MCMPs.

If the number of processors per chip is larger than 16, the superiority of super-IPGs over other networks, such as hypercubes and tori, is even more pronounced. For example, if a chip can accommodate 256 nodes, the bisection bandwidths and thus throughput for communication-intensive tasks of HSNs and SFNs are more than 4 times higher than those of hypercubes, when there are 256 or 64K chips in total. The advantages are not limited to large-scale parallel systems, but occur in small-scale systems as well. For example, as long as a chip has at least 4 nodes, and there are 4, 16, 64, or more chips in the parallel systems, the bisection bandwidths of these super-IPGs will be higher than that of a hypercube by at least 33%.

Corollary 4.10 *The bisection bandwidth of an N-node $\sqrt{N}$-ary 2-cube is $\frac{w\sqrt{NM}}{2}$, where w is the average aggregate off-chip bandwidth of a node and M is the number of nodes on a chip.*

Proof: The bisection width of an N-node $\sqrt{N}$-ary 2-cube is $2\sqrt{N}$, and a chip with M nodes has $4\sqrt{M}$ off-chip links, each of which has bandwidth $\frac{w\sqrt{M}}{4}$, so the bisection bandwidth is $\frac{w\sqrt{NM}}{2}$. □

It can be seen that the bisection bandwidth of a 2-dimensional torus is considerably lower than that of a hypercube. The bisection bandwidths of k-ary n-cubes are between those of $\sqrt{N}$-ary 2-cubes and hypercubes, and are thus also lower than those of super-IPGs presented in this paper.

The bisection bandwidths of HSNs, complete-CNs, and SFNs are in fact asymptotically optimal from a trivial lower bound and the gap is usually a small constant ratio.

Corollary 4.11 *The bisection bandwidth of HSNs, complete-CNs, and SFNs are asymptotically optimal when $l = O(1)$, within a factor smaller than $2l-2$ from the trivial lower bound $wN/2$, where l is the number of super-symbols, assuming that all nodes belonging to each nucleus are located on the same chip.*

For example, when $l = 2$ (or $l = 3$), the bisection bandwidth of an HSN, complete-CN, or SFN is somewhat larger than $wN/4$ (or $wN/8$, respectively) and the upper-to-lower-bounds ratio is somewhat smaller than 2 (or 4, respectively). Therefore, no network can outperform these super-IPGs by more than a small constant factor for communication-intensive tasks. In fact, no currently known network, excluding other super-IPGs, can outperform or match the performance of these super-IPGs under the unit chip capacity model. An implication of the results in this section is that even if we use the same number of chips, each with the same number of pins of equal bandwidth, the performance of a resultant parallel system can be considerably increased by simply changing the way in which these chips are interconnected. Note that even though we assumed only two levels of hierarchy for our network performance comparisons in this section, our results and methodology can be easily extended to hierarchical parallel architectures involving more than two levels. More details will be reported in the near future.

In [30], we have proposed to use the product of intercluster degree and diameter (referred to as ID-cost) or intercluster degree and intercluster diameter (referred to as II-cost) as a metric to compare different network topologies. In addition to bisection bandwidth, intercluster diameter, and average intercluster distance, we can also use the product of intercluster degree and average distance or the product of intercluster degree and average intercluster distance as a metric to compare network topologies and to demonstrate the superiority of super-IPGs. The details are omitted in this paper.

5 Conclusions

The success of a network topology depends on its topological parameters, conformance to available or anticipated technologies, algorithmic properties, reliability, and scalability. In [27, 30, 34], we have shown that (symmetric) super-IPGs possess desirable topological properties and are flexible and adaptable to various hardware constraints. In [27, 29, 33], we have shown that several super-IPGs can be laid out in areas smaller than that of a similar-size hypercube. In this paper, we showed the feasibility of developing algorithms for the entire class of super-IPGs as a whole, and provided a variety of important algorithms and embeddings for them, which proved their versatility. We also proposed the unit chip capacity model, and showed that super-IPGs can significantly outperform other networks for communication-intensive tasks when parallel computers are implemented as MCMPs.

References

[1] Ajtai, M., J. Komlós, and E. Szemerédi, "An $O(n \log n)$ sorting network," *Proc. ACM Symp. Theory of Computing,* 1983, pp. 1-9.

[2] Akers, S.B., D. Harel, and B. Krishnamurthy, "The star graph: an attractive alternative to the n-cube," *Proc. Int'l Conf. Parallel Processing,* 1987, pp. 393-400.

[3] Akers, S.B. and B. Krishnamurthy, "A group-theoretic model for symmetric interconnection networks," *IEEE Trans. Comput.,* Vol. 38, Apr. 1989, pp. 555-565.

[4] Amarasinghe, S.P., J.M. Anderson, C.S. Wilson, S.-W. Liao, R.S. French, M.W. Hall, B.R. Murphy, and M.S. Lam, "The multiprocessor as a general-purpose processor: a software perspective," *IEEE Micro*, 16(3), Jun. 1996.

[5] Basak D. and D.K. Panda, "Designing clustered multiprocessor systems under packaging and technological advancements," *IEEE Trans. Parallel Distrib. Sys.,* vol. 7, no. 9, Sep. 1996, pp. 962-978.

[6] Baumslag, M. and B. Obrenic, "Index-shuffle graphs," *Int'l J. Foundations of Computer Science,* vol. 8, no. 3, Sep. 1997, pp. 289-304.

[7] Bertsekas, D.P. and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods,* Athena Scientific, 1997.

[8] Bhuyan, L.N. and D.P. Agrawal, "Generalized hypercube and hyperbus structures for a computer network," *IEEE Trans. Comput.,* vol. 33, no. 4, Apr. 1984, pp. 323-333.

[9] Biggs, N., *Algebraic Graph Theory,* 2nd edition, Cambridge, Cambridge University Press, 1993.

[10] Cypher, R. and J.L.C. Sanz, "Hierarchical shuffle-exchange and de Bruijn networks," *Proc. IEEE Symp. Parallel and Distributed Processing,* 1992, pp. 491-496.

[11] Dally, W.J. "Performance analysis of k-ary n-cube interconnection networks," *IEEE Trans. Comput.,* Vol. 39, no. 6, Jun. 1990, pp. 775-785.

[12] Dally, W.J. and S. Lacy, "VLSI architecture: past, present, and future," *Proc. Advanced Research in VLSI Conf.,* 1999, pp. 232-241.

[13] Duh D., G. Chen, and J. Fang, "Algorithms and properties of a new two-level network with folded hypercubes as basic modules," *IEEE Trans. Parallel Distrib. Sys.,* vol. 6, no. 7, Jul. 1995, pp. 714-723.

[14] Efe, K. and A. Fernandez, "Products of networks with logarithmic diameter and fixed degree," *IEEE Trans. Parallel Distrib. Sys.,* vol. 6, no. 9, Sep. 1995, pp. 963-975.

[15] Ghose, K. and R. Desai, "Hierarchical cubic networks," *IEEE Trans. Parallel Distrib. Sys.,* vol. 6, no. 4, Apr. 1995, pp. 427-435.

[16] Hamdi, M. and R.W. Hall, "RCC-FULL: an efficient network for parallel computations," *J. Parallel Distrib. Comp.,* vol. 41, no. 2, Mar. 1997, pp. 139-155.

[17] Johnsson, S.L. and C.-T. Ho, "Optimum broadcasting and personalized communication in hypercubes," *IEEE Trans. Comput.,* vol. 38, no. 9, Sep. 1989, pp. 1249-1268.

[18] Lakshmivarahan, S. and S.K. Dhall, "A new hierarchy of hypercube interconnection schemes for parallel computers," *J. Supercomputing,* vol. 2, 1988, pp. 81-108.

[19] Lakshmivarahan, S., J.-S. Jwo, and S.K. Dhall, "Symmetry in interconnection networks based on Cayley graphs of permutation groups: a survey," *Parallel Computing,* Vol. 19, no. 4, Apr. 1993, pp. 361-407.

[20] Leighton, F.T., *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes,* Morgan-Kaufman, San Mateo, CA, 1992.

[21] Preparata, F.P. and J.E. Vuillemin, "The cube-connected cycles: a versatile network for parallel computation," *Communications of the ACM,* vol. 24, no. 5, May 1981, pp. 300-309.

[22] Sterling, T., P. Messina, and P. Smith, *Enabling Technologies for Petaflops Computing, MIT Press.*, 1995.

[23] Yeh, C.-H. and B. Parhami, "Parallel algorithms on three-level hierarchical cubic networks," *Proc. High Performance Computing Symp.,* Mar. 1996, pp. 226-231.

[24] Yeh, C.-H. and B. Parhami, "Hierarchical swapped networks: efficient low-degree alternatives to hypercubes and generalized hypercubes," *Proc. Int'l Symp. Parallel Architectures, Algorithms, and Networks,* Jun. 1996, pp. 90-96.

[25] Yeh, C.-H. and B. Parhami, "Recursive hierarchical swapped networks: versatile interconnection architectures for highly parallel systems," *Proc. IEEE Symp. Parallel and Distributed Processing*, Oct. 1996, pp. 453-460.

[26] Yeh, C.-H. and B. Parhami, "Cyclic networks – a family of versatile fixed-degree interconnection architectures," *Proc. Int'l Parallel Processing Symp.*, Apr. 1997, 739-743.

[27] Yeh, C.-H., "Efficient low-degree interconnection networks for parallel processing: topologies, algorithms, VLSI layouts, and fault tolerance," Ph.D. dissertation, Dept. Electrical & Computer Engineering, Univ. of California, Santa Barbara, Mar. 1998.

[28] Yeh, C.-H. and E.A. Varvarigos, "Macro-star networks: efficient low-degree alternatives to star graphs," *IEEE Trans. Parallel Distrib. Sys.,* vol. 9, no. 10, Oct. 1998, pp. 987-1003.

[29] Yeh, C.-H., B. Parhami, and E.A. Varvarigos, "The recursive grid layout scheme for VLSI layout of hierarchical networks," *Proc. Merged Int'l Parallel Processing Symp. & Symp. Parallel and Distributed Processing*, Apr. 1999, pp. 441-445.

[30] Yeh, C.-H. and B. Parhami, "The index-permutation graph model for hierarchical interconnection networks," *Proc. Int'l Conf. Parallel Processing*, Sep. 1999, pp. 48-55.

[31] Yeh, C.-H. and B. Parhami, "Routing and embeddings in cyclic Petersen networks: an efficient extension of the Petersen graph," *Proc. Int'l Conf. Parallel Processing*, Sep. 1999, pp. 258-265.

[32] Yeh, C.-H., B. Parhami, E.A. Varvarigos, and H. Lee, "VLSI layout and packaging of butterfly networks," *Proc. ACM Symp. Parallel Algorithms and Architectures,* 2000, pp. 196-205.

[33] Yeh, C.-H., E.A. Varvarigos, and B. Parhami, "Multilayer VLSI layout for interconnection networks," *Proc. Int'l Conf. Parallel Processing,* 2000, pp. 33-40.

[34] Yeh, C.-H. and B. Parhami, "A unified model for hierarchical networks based on an extension of Cayley graphs," *IEEE Trans. Parallel Distrib. Sys.*, to appear.

[35] Youssef, A., "Design and analysis of product networks," *Proc. Symp. Frontiers of Massively Parallel Computation,* 1995, pp. 521-528.

[36] Yun S.-K. and K.H. Park, "Hierarchical hypercube networks (HHN) for massively parallel computers," *J. Parallel Distrib. Comput.,* vol. 37, no. 2, Sep. 1996, pp. 194-199.