Recursive Hierarchical Swapped Networks: Versatile Interconnection Architectures for Highly Parallel Systems

Chi-Hsiang Yeh and Behrooz Parhami
Department of Electrical and Computer Engineering
University of California
Santa Barbara, CA 93106-9560, USA
{yeh@mimd.| parhami@}ece.ucsb.edu

Abstract

In this paper, we propose a new class of interconnection networks called recursive hierarchical swapped networks (RHSN) for general-purpose parallel processing. The node degrees of RHSNs can vary from a small number to as large as required, depending on recursive and hierarchical composition parameters and the nucleus graph chosen. The diameter of an RHSN can be asymptotically optimal within a small constant factor. We present efficient routing, semigroup computation, ascend/descend, matrix-matrix multiplication, and emulation algorithms, thus proving the versatility of RHSNs. In particular, on suitably constructed RHSNs, matrix multiplication can be performed faster than the DNS algorithm on a hypercube. Furthermore, ascend/descend algorithms, semigroup computation, and parallel prefix computation can be done using algorithms with asymptotically fewer communication steps than on a hypercube.

1. Introduction

High performance, low node degree, and simplicity of algorithms are all important, but often conflicting, requirements for interconnection networks. High-performance network classes, such as the hypercube and star graph [1], tend to have high node degree and, as a result, are not suitable for very large scale parallel systems.

To overcome the problem of unbounded node complexity in large hypercube and star networks, some constant-degree variants and alternatives, such as the cube-connected cycles (CCC) [17], de Bruijn graphs, butterfly networks [14], periodically regular chordal (PRC) rings [16], hypernets [10], WK-recursive networks [18], and symmetric hypernets [11], have been proposed and shown to have many desirable properties. Alternative topologies that have intermediate node degrees include reduced hypercubes (RH) [24], recursively connected complete (RCC) networks [8, 9] hierarchical cubic networks (HCN) [7], hierarchical folded-hypercube networks (HFN) [5], tightly connected networks (TCN) [3], and macro-star networks [22].

In this paper, we propose a new class of interconnection networks called *recursive hierarchical swapped networks (RHSNs)* for general-purpose parallel processing. RHSNs, a subclass of recursive hierarchical fully-connected (RHFC) networks [21], have desirable topological and algorithmic properties, which compare favorably with other popular topologies. The node degrees of RHSNs can vary over a wide range; the diameters of suitably constructed RHSNs are asymptotically optimal with respect to their node degrees. Parallel algorithms on RHSNs are usually fast and elegant. We present efficient algorithms for routing, semigroup computation, ascend/descend computation, emulation of homogeneous product networks [6, 23], and matrix multiplication. Existence of many efficient algorithms, along with desirable architectural properties, point to the usefulness and significance of RHSNs as parallel architectures.

In Section 2, we define RHSNs and derive some of their parameters. In Section 3, we present a simple routing algorithm on RHSNs. Section 4 covers efficient ascend/descend and emulation algorithms. In Section 5, we summarize the performance for semigroup computation and matrix-matrix multiplication on RHSNs. We conclude by noting that RHSNs combine some important advantages of both hypercubes (e.g., rich in fast and elegant algorithms) and star graphs (e.g., optimal diameter), with the additional advantage of lower node degree.

The following notations will be used throughout the paper: K_M denotes an M-node complete graph; Q_k denotes a k-cube; and Q_k^m denotes a k-dimensional hypercube of radix m [2, 12].

0-8186-7683-3/96 $05.00

2. Recursive hierarchical swapped networks

A recursive hierarchical swapped network (RHSN) is characterized by its nucleus graph G, its recursive depth r (henceforth simply depth) and numbers $l_r, l_{r-1}, ..., l_1$ of hierarchical levels at various depths. We first define hierarchical swapped networks (HSNs) [20, 21], or RHSNs of depth 1, and then present the construction of general r-deep RHSNs.

2.1. Hierarchical construction of HSNs

An l-level hierarchical swapped network, HSN(l, G), begins with a nucleus G, which forms an HSN$(1, G)$. The nucleus can be any connected graph or hypergraph (of more than one node), such as a mesh, hypercube, complete graph, star graph, or buslet. For simplicity, we always refer to G as the nucleus "graph".

To build a 2-level hierarchical swapped network, HSN(2, G), we use M identical copies of the nucleus G, each having M nodes. Each nucleus is viewed as a level-2 cluster, and is given a k-bit string X_2 as its address, where $k = \lceil \log_2 M \rceil$; we also give each node a k-bit string X_1 as its address within the nucleus to which it belongs. Node X_1 within nucleus X_2 has a $2k$-bit string $X_2' = X_2X_1$ as its address within the HSN$(2, G)$. Each of the M nucleus copies has a link connecting it to each of the other $M-1$ nuclei, via which node X_2X_1 connects to node X_1X_2. The links connecting nodes within the same nucleus are called *nucleus links*, or *level-1 links*.

To build an l-level hierarchical swapped network, HSN (l, G), we use M identical copies of HSN$(l-1, G)$. Each copy of the HSN$(l-1, G)$ is viewed as a level-l cluster, and is given a k-bit string X_l as its address; each node is already given a $k(l-1)$-bit string $X_{l-1}' = X_{l-1:1}$ as its address within the level-l cluster to which it belongs, where $X_{i:j} = X_iX_{i-1}\cdots X_{j+1}X_j$. Node X_{l-1}' within the level-l cluster X_l has a kl-bit string $X_l' = X_lX_{l-1}' = X_{l:1}$ as its address within the HSN(l, G). Each of the M level-l clusters has M^{l-1} nodes and M^{l-2} links connecting it to each of the other $M-1$ level-l clusters, via which node $X_lX_{l-1:2}X_1$ connects to node $X_1X_{l-1:2}X_l$.

This connectivity and the hierarchical construction are the reasons we call such networks "hierarchical swapped networks." The connecting links are called *level-l inter-cluster links*, or simply *level-l links*, $l \geq 2$. The recursive definition allows us to construct arbitrary-level HSNs based on any type of nucleus.

The nodes that do not have a level-l inter-cluster link are called the *leaders* of that level-l cluster. Leaders can be used as I/O ports or be connected to other leaders via their unused ports to provide better fault tolerance or to improve the performance and reduce the diameter of HSNs without increasing the node degree of the network. If leader $X_l'X_{l-1:2}X_l'$ connects to leader $X_l''X_{l-1:2}X_l''$, where $X_l' = M - X_l'' - 1$, the average distance between nodes and, in most cases, the diameter of the network will be reduced. This type of HSN is called *HSN with diameter links*. Varying the connectivity between leaders results in other classes of HSNs.

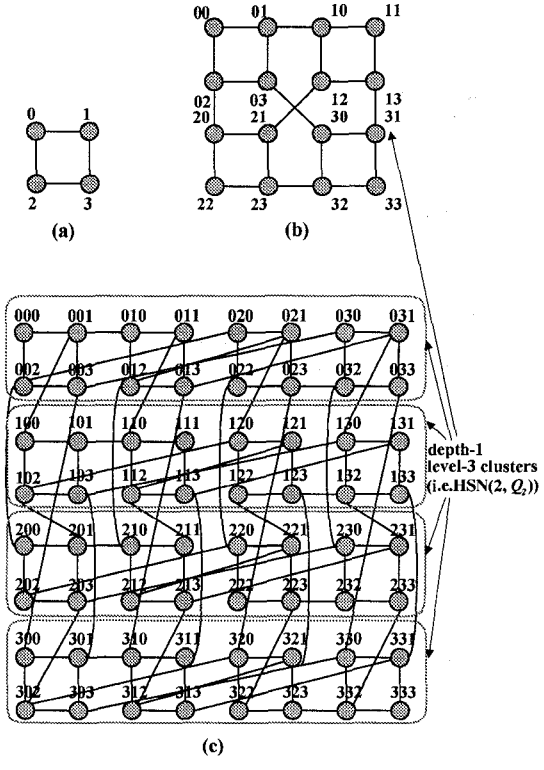

Figure 1. Structure of l-level hierarchical swapped networks, $l \leq 3$. (a) nucleus 2-cube, Q_2. (b) HSN$(2, Q_2)$. (c) The complete structure of HSN$(3, Q_2)$. Node addresses are expressed as radix-4 numbers.

Fig. 1 shows HSN(l, Q_2), $l = 1, 2, 3$. Several previously proposed interconnection networks are special cases of HSNs. For example, HCN(n, n) is a 2-level HSN$(2, Q_n)$ with diameter links, and HFN is a 2-level HSN$(2, FQ_n)$, where FQ_n denotes an n-dimensional folded hypercube.

2.2. Recursive definition of RHSNs

An r-deep RHSN$(l_r, l_{r-1}, ..., l_1, G)$ begins with an HSN (l_1, G), which forms a 1-deep RHSN(l_1, G), where "i-deep" stands for "i recursive levels deep."

To build a 2-deep RHSN, RHSN(l_2, l_1, G), we view the HSN(l_1, G) as a depth-2 nucleus, and construct the network as an HSN$(l_2, \text{HSN}(l_1, G))$. To build an r-deep RHSN, RHSN$(l_r, l_{r-1}, ..., l_1, G)$, we view the $(r-1)$-deep RHSN$(l_{r-1}, l_{r-2}, ..., l_1, G)$ as a depth-r nucleus, and construct it as an l_r-level HSN$(l_r, \text{RHSN}(l_{r-1}, l_{r-2}, ..., l_1, G))$.

The recursive definition allows us to construct networks of arbitrary depth and level based on any type of nucleus.

The level-j inter-cluster links of an HSN(l_i, RHSN(l_{i-1}, l_{i-2}, ..., l_1, G)) within the r-deep RHSN are called its *depth-i level-j inter-cluster links* or simply *depth-i level-j links*. The level-j clusters of an HSN(l_i,RHSN($l_{i-1}, l_{i-2}, ..., l_1$, G)) within the r-deep RHSN are called its *depth-i level-j clusters*.

It is worth noting that RCC networks [8, 9] form a subclass of RHSNs, RHSN($2,2,...,2,G$) without diameter links.

2.3. Basic properties

Let the nucleus G be a graph with M nodes of maximum degree d_G. The number of nodes in an HSN is increased by a factor M when the level is increased by 1. Therefore, the number of nodes N_r of an RHSN($l_r, l_{r-1}, ..., l_1, G$) is given by

$$N_r = N_{r-1}^{l_r} = M^{\prod_{i=1}^{r} l_i}, \tag{1}$$

where N_i is the number of nodes in an RHSN(l_i, l_{i-1}, ..., l_1, G).

From Eq. (1), we have

$$\prod_{i=1}^{r} l_i = \log_M N_r. \tag{2}$$

For an r-deep RHSN($l, l, ..., l, G$) of N_r nodes, the recursive depth r is given by

$$r = (\log_2 \log_2 N_r - \log_2 \log_2 M)/\log_2 l.$$

Clearly, the recursive depth is usually a small integer for an RHSN of practical size.

According to the definition of HSNs, the node degree is increased by 1 with each additional level, where the "node degree" of a network refers to the maximum degree of the nodes in the network throughout the paper. Thus, the node degree of an r-deep RHSN($l_r, l_{r-1}, ..., l_1, G$) of N nodes is given by

$$d_r = d_G + \textstyle\sum_{i=1}^{r} l_i - r.$$

The node degree of an r-deep RHSN($l, l, ..., l, G$) of N nodes is thus given by

$$d_r = d_G + (l-1)(\log_2 \log_2 N - \log_2 \log_2 M)/\log_2 l.$$

The node degree of an RHSN can be as small as $O(\log \log N)$ when d_G and l_i, $i = 1, 2, ..., r$, are not large.

3. Packet routing and network diameter

In this section, we first present a recursive routing algorithm to route a packet from node X to node Y in an HSN(l, G). We then generalize the algorithm to RHSNs.

Suppose that a routing algorithm for the nucleus G is known and that the routing algorithm for an HSN($l-1, G$) network is also known. Then, here is how routing is done at level l, $l \geq 2$.

Let the addresses of nodes X and Y within the HSN(l, G) be $X_{l:1}$ and $Y_{l:1}$, respectively, with the bit-strings X_l and Y_l being the addresses of the level-l clusters to which nodes X and Y belong.

- Case 1: $X_l = Y_l$: Nodes X and Y belong to the same level-l cluster. We use the routing algorithm for HSN($l-1, G$) to route the packet, since any level-l cluster is an HSN($l-1, G$).
- Case 2: $X_l \neq Y_l$: Nodes X and Y belong to different level-l clusters. To route a packet from node X to node Y, we use the routing algorithm for the nucleus G to route the packet from node $X_l X_{l-1:2} X_1$ to node $X_l X_{l-1:2} Y_l$. We next send the packet to node $Y_l X_{l-1:2} X_l$ via its level-l inter-cluster link in one step, and then use the routing algorithm for HSN($l-1, G$) to route the packet to node $Y_l Y_{l-1:1}$. Hence, the path through which the packet travels can be expressed as follows:

$$X_l X_{l-1:2} X_1 \xrightarrow{\text{nucleus}} X_l X_{l-1:2} Y_l \xrightarrow{\text{level-}l \text{ link}} Y_l X_{l-1:2} X_l \xrightarrow{\text{level-}l \text{ cluster}} Y_l Y_{l-1:1}.$$

Since an r-deep RHSN can be decomposed into an HSN based on an $(r-1)$-deep RHSN, the preceding routing algorithm can be applied to RHSNs recursively without modification, leading to the following lemma.

Lemma 3.1 *A packet can be routed in*

$$(T_R(1,G)+1)\log_M N - 1$$

time on an N-node RHSN based on an M-node nucleus G, where $T_R(1,G)$ is the time required for the packet routing algorithm on the nucleus G.

Proof: Let $T_R(l_i, l_{i-1}, ..., l_1, G)$ be the time required for the routing algorithm on an RHSN($l_i, l_{i-1}, ..., l_1, G$). The preceding recursive routing algorithm on HSN(l, G) requires time at most equal to

$$T_R(l,G) = T_R(l-1,G) + T_R(1,G) + 1 = lT_R(1,G) + l - 1.$$

Thus, we have

$$\begin{aligned} T_R(l_r, l_{r-1}, ..., l_1, G) &= l_r(T_R(l_{r-1}, l_{r-2}, ..., l_1, G) + 1) - 1 \\ &= (T_R(1,G)+1)\textstyle\prod_{i=1}^{r} l_i - 1, \end{aligned}$$

and from Eq. (2),

$$T_R(l_r, l_{r-1}, ..., l_1, G) = (T_R(1,G)+1)\log_M N - 1. \quad \square$$

It is interesting to note that for RHSNs of the same size, the routing time is determined exclusively by the nucleus graph (and its size), regardless of the various combinations of the numbers of recursive and hierarchical levels.

A simple fault-tolerant routing algorithm for RHFC networks can be found in [21].

Theorem 3.2 *The diameter of an N-node RHSN (without diameter links) is* $(D_G+1)\log_M N-1$, *where M is the number of nodes in the nucleus graph and* D_G *is the diameter of the nucleus graph.*

Proof: The routing algorithm gives an upper bound on the diameter of the RHSN as $(D_G+1)\log_M N-1$, assuming optimal routing algorithm on the nucleus graph.

Let X' and Y' be the addresses of two nodes that have distance D_G within the same nucleus G. It is straightforward to prove that the upper bound is equal to the distance between node $X=\underbrace{X'X'\cdots X'}_{\log_M N}$ and node $Y=\underbrace{Y'Y'\cdots Y'}_{\log_M N}$ in an RHSN without diameter links. □

Let G be a static undirected interconnection network that has N_G nodes, degree d_G, and diameter D_G. It is well known that a lower bound on the diameter D_G is $D_G=\Omega(\log_{d_G} N_G)$.

In this paper, a network G will be said to have *(asymptotically) optimal diameter* if and only if $D_G=\Theta(\log_{d_G} N_G)$. It has been shown in [21] that the diameter of an N-node RHFC network based on a degree-d_G nucleus G is asymptotically optimal if the number of inter-cluster links per node is at most equal to a polynomial in d_G and the diameter D_G of the nucleus G is asymptotically optimal.

By properly selecting parameters and the nucleus graph, the ratio of diameter over lower bound for an RHSN can be very small.

Corollary 3.3 *The diameter of an RHSN based on a* Q_n^m *is optimal asymptotically with a constant factor 1 if* $\log n=o(\log m)$, $\log_2\sum_{i=1}^m l_i\le\log_2 m+o(\log m)$, *and n is not a constant (in N).*

For example, we can choose $l_i, n=O(\log\log N)$ and $m=O(\sqrt[3]{\log N})$. Similarly, RHSNs based on generalized hypercubes [2] can also have optimal diameter asymptotically with a constant factor 1 by properly choosing the parameters of the nucleus generalized hypercubes. For comparison, the ratio of the network diameter over its lower bound is asymptotically 1.5 for a star graph, 2.5 for a CCC, and $\log_2\log_2 N$, for an N-node hypercube.

4. Emulating homogeneous product networks

RHSNs can perform ascend/descend algorithms efficiently. Suitably constructed RHSNs can also emulate a corresponding homogeneous (Cartesian) product network (HPN) [6, 23] of the same size *optimally (asymptotically within a constant factor) with respect to the node degrees* of the HPN and RHSNs, assuming either single-port or all-port communication. Since each node in the emulated HPN is mapped onto the node with the same address in the RHSN, the expansion and load are 1, and the algorithms (embeddings) are simple.

4.1. Homogeneous product networks

Homogeneous product networks (or *power networks*) form a subclass of product networks with identical component networks [6]. A homogeneous product network (HPN) is the iterated Cartesian product of the same graph, and 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 (binary) pk-dimensional hypercube is the p^{th} power of a k-cube; p-dimensional hypercube of radix $M\ge 2$ [2, 12] is the p^{th} power of K_M; and M-ary p-cube is the p^{th} power of an M-node ring.

Let l_0 be the node degree of the graph G. Each link of a node in G is given a distinct integer $i_a\in[0,l_0-1]$ as its label, and is called the dimension-i_a link. The dimension-i_a link of G_p is called the dimension-$(i_a+l_0(p-1))$ link of the product network $G_p\times G_{p-1}\times\cdots\times G_1$ with $G_j=G$.

4.2. Mixed radix number systems

In this subsection, we introduce a mixed radix number system, which is useful in presenting the time complexity of emulation algorithms and their operation steps.

Let i_a be an integer, $i_a\in[0,\prod_{j=0}^r l_j-1]$. It can be seen that i_a has a unique $(r+1)$-tuple $(a_r,a_{r-1},...,a_1,a_0)$, $a_j\in[0,l_j-1]$, such that $i_a=\sum_{j_1=0}^r(a_{j_1}\prod_{j_2=0}^{j_1-1}l_{j_2})$. The $(r+1)$-tuple $(a_r,a_{r-1},...,a_1,a_0)_{(l_r,l_{r-1},...,l_1,l_0)}$ is called the *mixed radix representation* of i_a.

4.3. Ascend/descend algorithms

Ascend/descend algorithms [14, 17] require successive operations on data items that are separated by a distance equal to a power of 2. Many applications, such as Fast Fourier Transform (FFT), bitonic sort, matrix multiplication, and convolution, can be formulated using algorithms in this general category.

Let $i_a=(a_1,a_0)_{(l,k)}$, $i_b=(b_1,b_0)_{(l,k)}$, and $i_a\le i_b$. We present the ascend algorithm ASC(i_a,i_b,l,G) (for operations on data separated by a distance 2^i, $i=i_a,i_a+1,...,i_b$) on HSN(l,G) as follows, assuming that the nucleus G has $M=2^k$ nodes.

ASC(i_a,i_b,l,G)
  for $i:=a_1$ to b_1 **do**
    begin
      Each node exchanges data
        via its level-$(i+1)$ inter-cluster link.
      $c_0:=$ **if** $i=a_1$ **then** a_0 **else** 0.
      $d_0:=$ **if** $i=b_1$ **then** b_0 **else** $k-1$.
      Perform ASC$(c_0,d_0,1,G)$.
      Each node exchanges data
        via its level-$(i+1)$ inter-cluster link.
    end

In this and other algorithms, if "level-1 inter-cluster links" (which do not exist) are specified, or a leader is asked to send data using the inter-cluster link that it does not have, the corresponding operation is simply ignored.

After performing the exchange step via level-i inter-cluster links, node $X_{l:1}$ will hold the data item from node $X_{l:i+1}X_1X_{i-1:2}X_i$. In essence, this moves data items separated by a distance of 2^j, $j = ki-k, ki-k+1, ..., ki-1$, into the same nucleus, such that they are now separated by a distance of 2^{j-ki+k}. Thus, we can use the ascend algorithm on the nucleus G (i.e., HSN$(1,G)$) to emulate ascend algorithm performed in dimensions j, $j = ki-k, ki-k+1, ..., ki-1$.

To perform descend algorithms, we simply replace ASC$(c_0,d_0,1,G)$ with DES$(c_0,d_0,1,G)$ and set the "step" of the **for** loop to -1; that is, we modify the first line to

for $i := b_1$ to a_1 step -1 **do**.

Since an RHSN can be viewed as an HSN, the ASC/DES algorithm can be applied recursively to RHSN$(l_r, l_{r-1}, ..., l_1, G)$ by letting ASC/DES$(i_a,i_b,l_r,l_{r-1},...,l_1,G) \stackrel{def}{=}$ ASC/DES$(i_a,i_b,l_r,$ RHSN$(l_{r-1},...,l_1,G))$. The algorithm can be viewed as emulating the ascend/descend algorithm on an HPC(p,G), where $p = \prod_{i=1}^{r} l_i$.

Theorem 4.1 *Ascend/descend algorithms involving s dimensions can be performed on an r-deep G-based RHSN in no more than*

$$(T_{asc/des}(1,G)+2)(\lceil (s-1)/k \rceil + 1) + 2r - 2$$

time, where the number of nodes in a nucleus G is 2^k, and $T_{asc/des}(1,G)$ is the time required for the k-step ascend/descend algorithms on the nucleus G.

Proof: The time complexity bound can be verified by expanding the algorithm. The number of steps required for exchanging data on inter-cluster links is no more than $2r + 2\lceil (s-1)/k \rceil$; the number of steps required to perform ascend/descend algorithms within the nuclei is no more than $T_{asc/des}(1,G)(\lceil (s-1)/k \rceil + 1)$. □

Corollary 4.2 *Ascend/descend algorithms (for all the $\log_2 N$ operations) on an N-node RHSN based on a k-cube can be performed in $\log_2 N + 2(\log_2 N/k - 1)$ time.*

Corollary 4.3 *Sorting can be performed in $\frac{1}{2}\log_2^2 N + O(\log^2 N/k)$ time on an N-node RHSN based on a k-cube.*

Proof: The result follows by performing the parallel bitonic sort, which requires $\log_2 N$ ascend iterations. □

When the number of nodes in the nucleus graph is not a power of two, some of the nodes can hold 2 values to emulate the ascend/descend algorithms of problem size $2^{p\lceil \log_2 M \rceil}$. The time is at most doubled, leading to the following corollary to Theorem 4.1.

Corollary 4.4 *Ascend/descend algorithms (for all the $\lceil \log_2 N \rceil$ operations) on an N-node RHSN based on an M-node complete graph can be emulated in $\Theta(\log_M N)$ communication steps.*

Although we can use the nucleus graph to emulate a graph whose size is a power of 2, most algorithms in the category of ascend/descend algorithms can be performed on a network whose size is not a power of 2 by assuming some "padding nodes." We present the result for sorting algorithm in the following corollary and also provide the tradeoffs between communication and computation steps.

Corollary 4.5 *Sorting can be performed in no more than*

$$(1+t/2)p^2 2^{\sum_{i=1}^{t}\lceil \log_2 m_i \rceil} \approx (1+t/2)\log_2^2 N/\log_2 M$$

communication steps and

$$\sum_{i=1}^{t}(m_i-1)p^2 2^{\sum_{i=1}^{t}\lceil \log_2 m_i \rceil} \approx t\left(\sqrt[t]{M}-1\right)\log_2^2 N/\log_2 M$$

computation steps on an N-node RHSN$(l_r, l_{r-1}, ..., l_1, K_M)$, for any integer $t \in [1, \log_2 M]$, where $p = \prod_{i=1}^{r} l_i = \log_M N$, $\prod_{i=1}^{t} m_i \geq M$, and m_i's are integers for all i.

Proof: To perform an ascend/descend computation on a complete graph K_{m_i}, node X within the K_{m_i}, $X = 0, 1, ..., m_i - 1$, first receives data from all the other nodes within the K_{m_i} and then find the X^{th} result of the corresponding ascend/descend computation. As a result, only one communication step and at most $m_i - 1$ computation steps are required. Since the complete graph K_M can be easily partitioned into $\prod_{i=2}^{t} m_i$ subghraphs, each having at most m_1 nodes, where $\prod_{i=1}^{t} m_i \geq M$, the tradeoff results follow. □

Clearly, this result can be applied to an RHSN based on a t-dimensional generalized hypercube with mixed radix $(m_1, m_2, ..., m_t)$ since we essentially emulate such RHSN (nucleus) in the proof of Corollary 4.5.

When there is no restriction on the ordering of outputs, the last step in the ASC/DES algorithm for an HSN can be removed, resulting in even better performance.

4.4. HPN emulation: single-dimension case

In this subsection, we assume single-port communication, with all the nodes only capable of using links of the same dimension at the same time. This assumption is used in some SIMD architectures and their algorithms, in order to reduce the cost of implementation, and is called the *single-dimension communication model* in this paper. Algorithms

assuming such model are also suitable for a parallel system using wormhole routing, especially when one input can only drive one output and is used in some papers (e.g., [15]).

The emulation algorithm is similar to the special case ASC/DES($i_a, i_a, l_r, l_{r-1}, ..., l_1, G$), which we call EPS algorithm.

Theorem 4.6 *Any step of the actions of all dimension-i_a links of an HPN(p, G) can be emulated on an RHSN(l_r, l_{r-1}, ..., l_1, G) in $2|S_a| + 1$ steps, where $p = \prod_{i=1}^{r} l_i$, $|S_a|$ is the number of elements in $S_a = \{i | a_i \neq 0, i = 1, 2, ..., r\}$, $i_a = (a_r, a_{r-1}, ..., a_1, a_0)_{(l_r, l_{r-1}, ..., l_1, l_0)}$, and l_0 is the node degree of the nucleus G.*

Proof: Emulation is done by exchanging data via depth-i level-$(a_i + 1)$ links for all nonzero a_i, using i, $i = r, r-1, ..., 2, 1, 0, 1, 2, ..., r-1, r$. No contention will occur. □

Clearly, any HPN(p, G) algorithm with single-dimension communication can be emulated on the corresponding r-deep RHSN with a slowdown factor at most equal to $2r+1$.

The following theorem provides the necessary and sufficient conditions for the emulation slowdown (or, alternatively, embedding dilation) to be optimal with respect to the node degree. The proof is omitted.

Theorem 4.7 *Any algorithm on an HPN(p, G) with single-dimension communication can be emulated with optimal slowdown (asymptotically within a constant factor) with respect to the node degrees using the EPS algorithm on an RHSN($l_r, l_{r-1}, ..., l_1, G$), if and only if $\log r = O(\log l_{max})$ and $|S| = \Theta(r)$, where $S = \{i | \log l_i = \Theta(\log l_{max}), i = 1, 2, ..., r\}$, [i.e., $\sum_{i=1}^{r} \log l_i = \Theta(r \log l_{max})$], $p = \prod_{i=1}^{r} l_i$, l_0 is the node degree of the nucleus G, and $l_{max} = max_{i=0,1,...,r}(l_i)$.*

4.5. HPN emulation: all-port communication

In this subsection, we assume all-port communication, with all the nodes capable of using links of all dimensions at the same time. The required emulation can be done simply by performing single-dimension emulation for all dimensions at the same time with proper scheduling.

Theorem 4.8 *Any HPN(p, G) algorithm with all-port communication can be emulated on an RHSN(l_r, l_{r-1}, ..., l_1, G) with a slowdown factor no more than $2r + 1 + max_{i=1,2,...,r}(2pl_0/l_i, p)$, where $p = \prod_{i=1}^{r} l_i$ and l_0 is the node degree of the nucleus graph G.*

Proof: There exist many schedules that guarantee the slowdown upper bound. It can be verified from the EPS algorithm that $2pl_0/l_i$ packets will pass a depth-i level-j inter-cluster link and p packets will pass a nucleus link. Since the dilation for embedding the HPN is $2r + 1$ (Theorem 4.6), the slowdown factor is given by $2r + 1 + max_{i=1,2,...,r}(2pl_0/l_i, p)$. □

Note that the actual slowdown is usually smaller than the upper bound given in Theorem 4.8 since the dilation can be virtually hidden. For example, when $r = 1, l_1 = 4$ and $l_0 = 3$, the slowdown factor is 6 rather than 9.

By properly choosing the nucleus size and the number of recursive and hierarchical levels, an RHSN can emulate a corresponding HPN with asymptotically optimal slowdown (within a constant factor).

Theorem 4.9 *Any HPN(p, G) algorithm with all-port communication can be emulated on an RHSN(l_r, l_{r-1}, ..., l_1, G) with asymptotically optimal slowdown if $l_i = \Theta(l_j)$, $i, j = 0, 1, ..., r$, and $\log r = \Theta(\log l_0)$, where $p = \prod_{i=1}^{r} l_i$ and l_0 is the node degree of the nucleus graph G.*

5. Other parallel algorithms

In this section, we summarize the tradeoff results for the performance of semigroup computation and matrix-matrix multiplication on RHSNs.

5.1. Semigroup computation

Assume that each node in the HSN(l, G) holds a value v_j for computation. Semigroup computation on the HSN can be performed in three phases:

- Phase 1: Perform semigroup computation on each level-l cluster, separately.
- Phase 2: Each node exchanges its value via its level-l inter-cluster link.
- Phase 3: Perform semigroup computation on each nucleus, separately.

Theorem 5.1 *Semigroup computation can be performed in $(T_S(1, G) + 1) \log_M N - 1$ time on an N-node RHSN based on an M-node nucleus G, where $T_S(1, G)$ is the time required to perform semigroup computation on G.*

Proof: Similar to the proof for Lemma 3.1. □

Corollary 5.2 *Semigroup computation can be performed in $(1 + 1/k) \log_2 N - 1$ time on an N-node RHSN based on a k-cube.*

A strategy similar to the one in the proof of Corollary 4.5 leads to the following corollary to Theorem 5.1.

Table 1. Complexity of several algorithms on the hypercube and RHSNs with size N.

Networks	Semigroup Computation	Matrix Multiplication	FFT	Bitonic Sort
RHSN$(3l'_r, l_{r-1}, ..., l_0, Q_k)^\dagger$	$\log_2 N + o(\log N)$	$\frac{1}{3}\log_2 N + o(\log N)$	$\log_2 N + o(\log N)$	$\frac{1}{2}\log_2^2 N + o(\log^2 N)$
RHSN$(3l'_r, l_{r-1}, ..., l_0, K_M)^\ddagger$	$\frac{2\log_2 N}{\log_2 M} - 1$	$\frac{2\log_2 N}{3\log_2 M} + o(\frac{\log N}{\log M})$	$O\left(\frac{\log N}{\log M}\right)$	$\frac{3\log_2^2 N}{2\log_2 M} + o(\frac{\log^2 N}{\log M})$
Hypercube	$\log_2 N$	$\frac{5}{3}\log_2 N$	$\log_2 N$	$\frac{1}{2}\log_2^2 N - 1$

† k is not a constant in N.
‡ Performance evaluated by the number of communication steps required.

Corollary 5.3 *Semigroup computation can be performed in* $(1+t)\log_M N$ *communication steps and*

$$(\Sigma_{i=1}^t m_i - t)\log_M N \approx t\left(\sqrt[t]{M} - 1\right)\log_M N$$

computation steps on an N-node RHSN based on an M-node complete graph, for any integer $t \in [1, \log_2 M]$, *where* $\prod_{i=1}^t m_i \geq M$.

Parallel prefix computation can be performed using algorithms similar to semigroup computation.

5.2. Matrix-matrix multiplication

Another major advantage of RHSNs is that they can perform matrix-matrix multiplication at a high speed. In [19], we have shown that 3-level hierarchical cubic networks, which are 3-level HSNs based on the hypercube, can perform matrix-matrix multiplication efficiently. Those result can be easily generalized to an RHSN$(l_r, l_{r-1}, ..., l_1, G)$ where l_r is a multiple of 3.

Theorem 5.4 $\sqrt[3]{N} \times \sqrt[3]{N}$ *matrix-matrix multiplication can be performed on an N-node RHSN*$(l_r, l_{r-1}, ..., l_1, G)$ *in* $(T_S(1,G)/3 + 1/3)\log_M N + 5l_r/3 - 2$ *time steps, where* $T_S(1,G)$ *is the time required for semigroup computation on the nucleus G, M is the number of nodes in the nucleus G, and* l_r *is a multiple of 3.*

Note that the output configuration of the resultant product matrix is exactly the same as the initial configuration required by the algorithm [19], making cascaded operations possible.

Corollary 5.5 $\sqrt[3]{N} \times \sqrt[3]{N}$ *matrix-matrix multiplication can be performed on an N-node RHSN*$(l_r, l_{r-1}, ..., l_1, Q_k)$ *in* $\frac{\log_2 N}{3} + \frac{\log_2 N}{3k} + \frac{5l_r}{3} - 2$ *time, where* l_r *is a multiple of 3.*

From Corollary 5.5, we know that matrix-matrix multiplication on some RHSNs can be performed asymptotically faster than the DNS algorithm on hypercube by a factor of 5, and faster than the DNS algorithm on shuffle-exchange by a factor of 13 [4].

The following corollary follows from Theorem 5.4 and Corollary 5.3.

Corollary 5.6 $\sqrt[3]{N} \times \sqrt[3]{N}$ *matrix-matrix multiplication can be performed in* $(1+t)\log_M N/3 + 5l_r/3 - 2$ *communication steps and*

$$(\Sigma_{i=1}^t m_i - t)\log_M N/3 \approx t\left(\sqrt[t]{M} - 1\right)\log_M N/3$$

computation steps on an N-node RHSN$(l_r, l_{r-1}, ..., l_1, K_M)$, *for any integer* $t \in [1, \log_2 M]$, *where* $\prod_{i=1}^t m_i \geq M$, *and* l_r *is a multiple of 3.*

6. Scaling up RHSNs

To obtain variants of RHSNs with smaller step sizes while retaining most algorithmic and topological properties, we can use $M_{i,j}$ identical copies of the RHSN, where $M_{i,j}$ is the number of nodes in a depth-i level-j cluster of the original RHSN. Each copy is given a $k_{i,j}$-bit string as its address, which forms the most significant $k_{i,j}$ bits of the addresses of the nodes belonging to that copy, where $k_{i,j} = \lceil \log_2 M_{i,j} \rceil$. The links connecting these copies can be obtained in a way similar to the construction of an HSN. In other words, the address of the new neighbor of a node can be obtained by swapping the most significant $k_{i,j}$-bit string of the node with the the least significant $k_{i,j}$-bit string of the node.

It can be seen that all the algorithms presented in this paper can be applied to such networks with minor modifications. For example, to perform the semigroup computation on such networks, we simply perform the semigroup computation on each copy in parallel, exchange data via the new links, and then perform the semigroup computation on each depth-i level-j cluster in parallel. Analysis of the properties and performance of such networks is straightforward.

To obtain even smaller step size, we can also adopt a strategy similar to the *clustered star* or *incomplete star* [13]. That is, we can remove some of the top-level clusters from an RHSN. The diameter of the resultant network will be at most equal to that of the original RHSN.

7. Conclusion

In this paper, we have proposed a new class of interconnection networks for modular construction of massively parallel computers. RHSNs combine important properties of both hypercube (e.g., a wealth of fast, elegant algorithms) and star graphs (e.g., optimal diameter), with the additional advantage of using nodes of low degree, making them less expensive to implement.

We have presented simple and efficient algorithms for routing and ascend/descend computation. RHSNs based on a complete graph or generalized hypercube can perform semigroup computation, matrix-matrix multiplication, and the above mentioned algorithms with asymptotically fewer communication steps than hypercube. Moreover, suitably constructed RHSNs can emulate corresponding homogeneous product networks with asymptotically optimal slowdown. As a consequence, we obtain a variety of efficient algorithms on RHSNs through emulation, thus proving the versatility of these networks. We compare the performance of several important algorithms on the hypercube and RHSNs in Table 1. These results demonstrate that RHSNs are attractive candidates for high-performance networks with reasonable cost.

References

[1] 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.

[2] 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.

[3] Breznay, P.T. and M.A. Lopez, "Tightly connected hierarchical interconnection networks for parallel processors," *Proc Int'l Conf. Parallel Processing,* vol. I, 1993, pp. 307-310.

[4] Dekel, E., D. Nassimi, and S. Sahni, "Parallel matrix and graph algorithms," *SIAM J. Computing,* vol. 10, no. 4, Nov. 1981, pp. 657-675.

[5] 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.

[6] 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.

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

[8] Hamdi, M., "A class of recursive interconnection networks: architectural characteristics and hardware cost," *IEEE Trans. Circuits and Sys.–I: Fundamental Theory and Applications,* vol. 41, no. 12, Dec. 1994, pp. 805-816.

[9] Hamdi, M. and R.W. Hall, "RCC-FULL: an efficient network for parallel computations," *J. Parallel Distrib. Comp.,* to appear.

[10] Hwang, K. and J. Ghosh, "Hypernet: a communication efficient architecture for constructing massively parallel computers," *IEEE Trans. Comput.,* vol. 36, no. 12, Dec. 1987, pp. 1450-1466.

[11] Kaushal, R.P. and J.S. Bedi, "Comparison of hypercube, hypernet, and symmetric hypernet architectures," *Computer Architecture News,* vol. 20, no. 5, Dec. 1992, pp. 13-25.

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

[13] Latifi, S. and N. Bagherzadeh, "Incomplete star: an incrementally scalable network based on the star graph," *IEEE Trans. Parallel Distrib. Sys.,* vol. 5, no. 1, Jan. 1994, pp. 97-102.

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

[15] Mišić J. and Z. Jovanović, "Communication aspects of the star graph interconnection network," *IEEE Trans. Parallel Distrib. Sys.,* vol. 5, no. 7, Jul. 1994, pp. 678-687.

[16] Parhami B., "Periodically regular chordal ring networks for massively parallel architectures," *Proc. Symp. Frontiers of Massively Parallel Computation,* 1995, pp. 315-322.

[17] 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.

[18] Vecchia, G.D. and C. Sanges, "Recursively scalable networks for message passing architectures," *Proc. Conf. Parallel Processing and Applications,* 1987, pp. 33-40.

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

[20] 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,* 1996, pp. 90-96.

[21] Yeh, C.-H. and B. Parhami, "Recursive hierarchical fully-connected networks: a class of low-degree small-diameter interconnection networks," *Proc. IEEE Int'l Conf. Algorithms and Architectures for Parallel Processing,* 1996, pp. 163-170.

[22] Yeh, C.-H. and E.A. Varvarigos, "Macro-star networks: efficient low-degree alternatives to star graphs for large-scale parallel architectures," *Proc. Symp. Frontiers of Massively Parallel Computation,* 1996, to appear.

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

[24] Ziavras, S.G., "RH: a versatile family of reduced hypercube interconnection networks," *IEEE Trans. Parallel Distrib. Sys.,* vol. 5, no. 11, Nov. 1994, pp. 1210-1220.