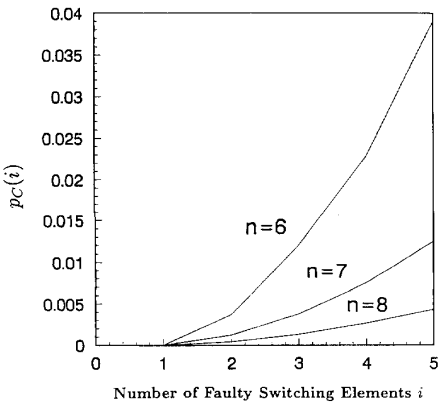

Fig. 4. $p_C(i)$ against number of faulty switching elements.

elements. For example, given that there are five faulty switching elements, this probability is roughly equal to 0.039, 0.012, or 0.004 for $n = 6$, 7, or 8, respectively. Therefore, it is very likely that a complicated reconfiguration process can be avoided, as long as the switching elements in the first and the last stages can be kept fault-free. In real implementations, the switching elements located in the first and the last stages can be well protected by adding redundant circuits.

V. Conclusion

In this short note, we determined a necessary and sufficient condition for a faulty banyan network to possess the DFA property, and designed a testing procedure based on the condition. The testing procedure was used to decompose a faulty banyan network into subnetworks possessing the DFA property. We showed that banyan networks can tolerate quite a few faulty switching elements without losing the DFA property. As a consequence, a complicated reconfiguration process can often be avoided if the switching elements in the first and the last stages can be kept fault-free. There are several interesting topics in this area that can be further studied. For example, sufficient conditions for a fault set to be noncritical, other than those determined previously, can be searched to expedite the testing. Another topic that is currently under investigation is to design and evaluate the performance of rerouting schemes.

References

[1] L. R. Goke and G. J. Lipovski, "Banyan network for partitioning multiprocessor systems," *Proc. 1st Ann. Symp. Comput. Architecture*, 1980, pp. 21–30.

[2] C. L. Wu and T. Y. Feng, "On a class of multistage interconnection networks," *IEEE Trans. Comput.*, vol. C-29, no. 8, pp. 694–702, Aug. 1980.

[3] C. P. Kruskal and M. Snir, "The performance of multistage interconnection network," *IEEE Trans. Comput.*, vol. C-32, no. 12, pp. 1091–1098, Dec. 1983.

[4] T. T. Lee, "Nonblocking copy networks for multicast packet switching," *IEEE J. Selected Areas Commun.*, vol. 6, pp. 1455–1467, Dec. 1988.

[5] J. P. Shen and J. P. Hayes, "Fault-tolerance of dynamic-full-access interconnection networks," *IEEE Trans. Comput.*, vol. C-33, pp. 443–454, Mar. 1984.

[6] T. Y. Feng and I. P. Kao, "On fault-diagnosis of some multistage networks," *Proc. 1982 Int. Conf. Parallel Processing*, 1982, pp. 99–101.

[7] D. P. Agrawal, "Testing and fault-tolerance of multistage interconnection networks," *IEEE Comput.*, vol. 15, pp. 41–53, Apr. 1982.

[8] S. Thanawastien and V. P. Nelson, "Optimal fault detection test sequences for shuffle/exchange networks," in *Proc. 13th Ann. Int. Symp. Fault-Tolerant Comput.* , 1983, pp. 442–445.

[9] C. L. Wu and T. Y. Feng, "Fault diagnosis for a class of multistage interconnection networks," *IEEE Trans. Comput.*, vol. C-30, pp. 743–758, Oct. 1981.

[10] D. P. Agrawal and J. S. Leu, "Dynamic accessibility testing and path length optimization of multistage interconnection networks," *IEEE Trans. Comput.*, vol. C-34, no. 3, pp. 255–266, Mar. 1985.

[11] A. Varma and C. S. Raghavendra, "Fault-tolerant routing in multistage interconnection networks," *IEEE Trans. Comput.*, vol. 38, pp. 385–393, Mar. 1989.

[12] V. P. Kumar and S. J. Wang, "Reliability enhancement by time and space redundancy in multistage interconnection networks," *IEEE Trans. Reliability*, vol. 40, pp. 461–473, Oct. 1991.

RH: A Versatile Family of Reduced Hypercube Interconnection Networks

Sotirios G. Ziavras

***Abstract*—The binary hypercube has been one of the most frequently chosen interconnection networks for parallel computers because it provides low diameter and is so robust that it can very efficiently emulate a wide variety of other frequently used networks. However, the major drawback of the hypercube is the increase in the number of communication channels for each processor with an increase in the total number of processors in the system. This drawback has a direct effect on the very large scale integration complexity of the hypercube network. This short note proposes a new topology that is produced from the hypercube by a uniform reduction in the number of edges for each node. This edge reduction technique produces networks with lower complexity than hypercubes while maintaining, to a high extent, the powerful hypercube properties. An extensive comparison of the proposed reduced hypercube (RH) topology with the conventional hypercube is included. It is also shown that several copies of the popular cube-connected cycles network can be emulated simultaneously by an RH with dilation 1.**

***Index Terms*—Emulation, hypercube, hypercube-like systems, interconnection networks, cube-connected cycles, parallel processing**

I. Introduction

A wide variety of interconnection networks have been proposed for parallel computing systems, including rectangular meshes, trees, shuffle exchange networks, Omega networks, indirect binary n-cubes, and direct binary n-cubes. The family of direct binary n-cubes is a special case of the family of direct k-ary n-cubes with n dimensions and k nodes in each dimension [24]. In the rest of this short note, the term *n-dimensional hypercube or n-cube* is used to denote the direct binary n-cube. The n-dimensional hypercube is composed of 2^n nodes and has n edges per node. If unique n-bit binary addresses

Manuscript received January 4, 1993; revised August 23, 1993, and November 24, 1993. This work was supported in part by the National Science Foundation under Grants CCR-9109084 and CDA-9121475. and in part by the New Jersey Institute of Technology under Grant SBR-421240.

The author is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, University Heights, Newark, NJ 07102 USA; e-mail: ziavras@hertz.njit.edu.

IEEE Log Number 9403105.

are assigned to the nodes of the hypercube, then an edge connects two nodes if and only if their binary addresses differ in a single bit.

A large number of papers have dealt with hypercubes, and some commercial hypercube-based systems have successfully been introduced. Such systems include the Intel *i*PSC, the *n*CUBE, and the Connection Machine CM-2 [15]. The most important reason for the success of the hypercube network has been its inherent capability of supporting the interconnection of large numbers of resources with small resultant diameters. The *diameter* of a network is defined as the maximum of the shortest distances for all pairs of nodes. The small diameter of the hypercube network, which is equal to n for the n-dimensional hypercube, is the direct result of its high degree of interconnectivity and its regular structure. These advantages of the hypercube network can result in the efficient emulation of other topologies frequently used in parallel processing. Several algorithms have been proposed for embedding important topologies into the hypercube, like rectangular meshes [4], [18], trees [10], [16], [18], and pyramids [4], [21], [29].

Nevertheless, systems that implement the pure hypercube network have a major drawback. The number of communication ports and channels per processor is the same as the logarithm of the total number of processors in the system. Therefore, the number of communication ports and channels per processor increases by increasing the total number of processors in the system. This drawback has a direct effect on the very large scale integration (VLSI) complexity of the network; therefore, the construction of massively parallel hypercube systems becomes a Herculean task. Although this drawback seems to be waived in the case of incomplete hypercubes [19], [26], a large number of communication ports may be wasted, and, as a consequence, a significant portion of a system's cost may be spent for unused resources. For example, an incomplete hypercube with 1280 processors can be constructed from two complete hypercubes composed of 1024 and 256 processors, respectively. Assuming bidirectional channels, the interconnection of the two complete hypercubes requires a number of communication ports per processor equal to 11 and 9, respectively, for the two constituent hypercubes (in contrast to 10 and 8, respectively, for the corresponding conventional hypercubes). The total number of unused communication ports in this system is equal to 768 (i.e., 1024–256), assuming that all the extra links of the hypercube with the smaller number of dimensions are used. In addition, the VLSI complexity of the incomplete hypercube is not often drastically lower than that of the respective complete hypercube.

In a similar fashion, the basic building block of hierarchical cubic networks (HCN's) [13] is the hypercube. However, HCN's use a smaller number of edges per node, and their diameter is smaller than that of respective hypercubes. An HCN that contains 2^{m+n} nodes, where $n \geq m$, has a diameter that is less than or equal to $m + n$; $m + n$ is the diameter of the hypercube with the same number of nodes. This HCN will employ $n + 1$ edges per node. The structure of HCN's for additional levels is not regular, because interhypercube edges are distinguished according to their class.

A methodology proposed by this author in [28] performs slight modifications in hypercubes to support their incremental growth without introducing or wasting any resources for individual hypercubes. This methodology allows the construction of systems that satisfy the following four vital goals.

1) The basic building blocks of the resultant (multicube) systems are slightly modified hypercubes (MH's). Although their structure is not perfectly regular, their performance is comparable to that of the hypercube. For practical cases, the diameter and the average internode distance of MH's are smaller than those of conventional hypercubes with the same number of nodes.
2) The total number of processors in a multicube system is not necessarily a power of 2.
3) The numbers of communication ports and links per processor do not necessarily grow with an increase in the total number of processors in multicube systems.
4) Finally, all the resources in multicube systems can be fully used.

The cube-connected cycles [23] CCC(n) is obtained from the n-dimensional hypercube by substituting a ring with n nodes for each node in the hypercube. Each node in a ring then implements a distinct connection in one of the n dimensions. The advantage of the CCC(n) is that the node connectivity is always 3, independently of the value of n.

Several other variations of the hypercube have been proposed. Although they may improve one or more topological properties of the hypercube or its already discussed variations, they do not reduce its VLSI complexity. Quite often the VLSI complexity is increased. A brief discussion of the most important remaining hypercube variations follows. The twisted cube [1] needs the same amount of resources as the conventional hypercube. It is constructed by repositioning some of the links in the hypercube so that they now span two dimensions. The advantage of the twisted cube is that its diameter is only $\lceil (n+1)/2 \rceil$ for a network with 2^n nodes. The asymmetry in the network produced affects its dynamic performance, so the improvement in performance over the hypercube is not nearly as much as the reduction in diameter. Another twisted cube variant repositions two links within a single 4-cycle [12]. For a network composed of 2^n nodes, the diameter is $n-1$. Enhanced incomplete hypercubes [5] are constructed from incomplete hypercubes by directly connecting pairs of processors with extra links attached to otherwise unused ports. The extra links decrease the diameter of the network and the average distance between pairs of nodes, but increase its VLSI complexity.

The crossed cube [11] has the same node and edge complexity as the hypercube, but about half its diameter. It emulates the hypercube with dilation equal to 2. A folded hypercube [2] is constructed from a conventional hypercube by establishing for each node direct connection with the unique node that is farthest from it. The folded hypercube may perform better than the corresponding conventional hypercube, because of its smaller diameter, which is $\lceil n/2 \rceil$ for a network containing 2^n nodes, better average internode distance, and less message traffic density. Because a folded hypercube with 2^n nodes has 2^n additional communication ports and 2^{n-1} additional communication channels when compared to the corresponding n-dimensional hypercube, it results in higher VLSI complexity.

Some other variations for which it could safely be stated that they are farther away from the hypercube than the above variations are the cubical ring connected cycles [3], the block-shuffled hypercube [17], hyper-rectangulars [20], the generalized folding cube [6], the spanning bus hypercube [27], and the generalized supercube [25]. Attempts to produce networks with better topological properties than the hypercube have also been made (see [7] for recent work).

This short note proposes a new family of interconnection networks which are produced by uniform removal of several edges from the conventional hypercube. As a result, the new networks, namely, reduced hypercubes (RH's), have reduced complexity when compared to the conventional hypercube. This allows the construction of RH's with many more processors than in conventional hypercubes, and therefore mitigates the architecture expansion problem. The results show that performance comparable to that of the conventional hypercube may be obtained at a much lower cost.

The rest of this short note is organized as follows. Section II formally introduces the family of reduced hypercube networks. Section III introduces routing algorithms for RH's and investigates

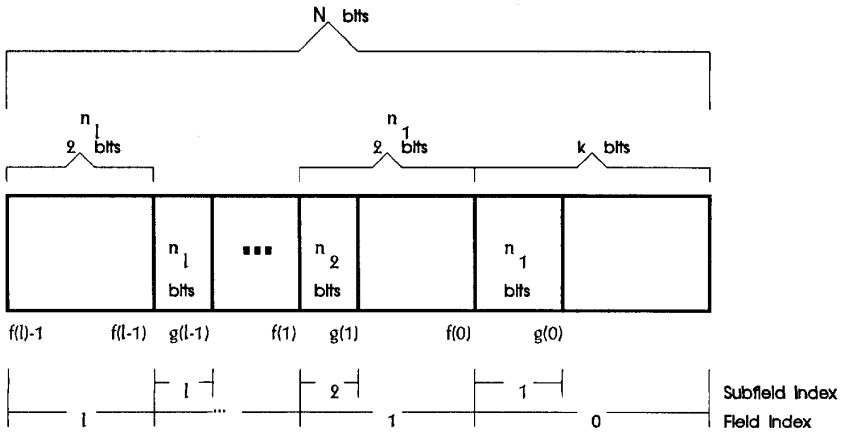

Fig. 1. The fields and subfields of node addresses for the RH($k, n_1, n_2, \cdots, n_l$).

their performance. Section IV discusses the efficiency of hypercube emulation on RH's. Section V presents the most important topological properties of RH's. Finally, Section VI presents conclusions and future research goals.

II. Reduced Hypercube Interconnection Networks

This section formally introduces the family of RH interconnection networks. The main objective is to produce networks that carry on, to a large extent, the capabilities of the conventional hypercube network while requiring lower cost due to lower VLSI complexity. The lower VLSI complexity of RH's permits the construction of systems with more PE's (processing elements) than are found in conventional hypercubes.

In the following discussion, RH($k, n_1, n_2, \cdots, n_l$) denotes a *reduced hypercube* network where every PE is attached to $k + l$ bidirectional communication channels and the total number of PE's is equal to 2^N, with the following condition:

$$N = k + \sum_{i=1}^{l} 2^{n_i}. \tag{1}$$

l is referred to as the number of levels in the RH. It is assumed that $k > 0$, $n_1 \leq k$, $n_{i+1} \leq 2^{n_i}$ for $1 \leq i \leq l-1$, and $n_i > 0$ for $1 \leq i \leq l$. RH(k) denotes the conventional k-dimensional hypercube.

The proposed family of reduced hypercube networks is produced by uniformly removing some of the edges from conventional hypercubes. For the construction of the RH($k, n_1, n_2, \cdots, n_l$) from the N-dimensional hypercube with the same number of nodes, the N-bit address of each node is divided into $l + 1$ fields. In addition, l subfields are identified. More specifically, the least significant k bits of a node's address constitute the 0th field and represent the address of the node within a complete k-cube *building block*. The first subfield comprises the most significant n_1 bits of this 0th field and represents the address of a $(k - n_1)$-dimensional subcube in the k-cube building block that contains this node. In general, assuming that the subscript of the least significant bit in the address is 0, the bits with subscripts $f(j-1) = k + \sum_{i=1}^{j-1} 2^{n_i}$ through $f(j) - 1 = k + \sum_{i=1}^{j} 2^{n_i} - 1$, where $j = 1, 2, \cdots, l$, form the jth *field* of the address. The 0th field of the address contains the least significant k bits. In addition, the bits with subscripts $g(j-1) = f(j-1) - n_j$ through $f(j-1) - 1$ form the jth *subfield* of the address, for $j = 1, 2, \cdots, l$. Fig. 1 illustrates the location of these fields and subfields in the N-bit address, where N is given by (1).

The RH($k, n_1, n_2, \cdots, n_l$) is produced with the removal of edges from the N-dimensional hypercube in the following manner. Based on the above identification of fields and subfields in node addresses, from the N edges that are attached to each node in the original N-dimensional hypercube, we keep the following sets that comprise a total of $k + l$ edges.

- *Set 1:* The k edges of the N-dimensional hypercube that traverse the k lowest dimensions (i.e., dimensions 0 through $k-1$) and connect the referenced node with k distinct nodes are kept. Therefore, a complete subcube with k dimensions that includes the referenced node is formed. This subcube was referred to as a building block in the preceding paragraph.
- *Set 2:* This set is composed of l edges that were also present in the original N-dimensional hypercube. More specifically, one edge is kept for each of the remaining l fields of the node's address. For the jth field, where $1 \leq j \leq l$, the selected edge is the one that connects directly the referenced node with the node whose address differs only in the mth bit of the jth field, where m is the value in the jth subfield, and the subscript of the least significant bit in a field is 0.

The RH(k, n) can be viewed as a 2^n-cube of k-cubes. For the purpose of simplicity, ";" is used here to separate consecutive fields in the address while a subfield lies between a "," and the first ";" to its left. For example, the neighbors attached to set 1 edges of the node with address (01101010; 101, 0; 11, 01) in the RH(4,2,3) have addresses (01101010; 101, 0; 11, 0$\underline{0}$), (01101010; 101, 0; 11, $\underline{1}$1), (01101010; 101, 0; 1$\underline{0}$, 01), and (01101010; 101, 0; $\underline{0}$1, 01). The underlined bits are those that differ from the corresponding bits in the address of the referenced node. The two neighbors that correspond to set 2 edges are (01101010; $\underline{0}$01, 0; 11, 01) and (01$\underline{0}$01010; 101, 0; 11, 01).

Fig. 2 shows the structure of the RH($k, 2$), where the large squares represent the k-cube building blocks, the numbers above these squares represent the decimal values in field 1 of addresses for PE's in the corresponding building blocks, the numbers within the quadrants of large squares are subcube addresses of $(k-2)$-cubes as well as the decimal values in subfield 1 of contained-PE addresses, and each curved line represents 2^{k-2} bidirectional communication channels that interconnect pairs of PE's with the same addresses in two $(k-2)$-cubes in order to form a $(k-1)$-cube.

Assuming a conventional hypercube with the same number of PE's (i.e., an N-cube), the ratio of the total number of channels in the conventional hypercube to the total number of channels

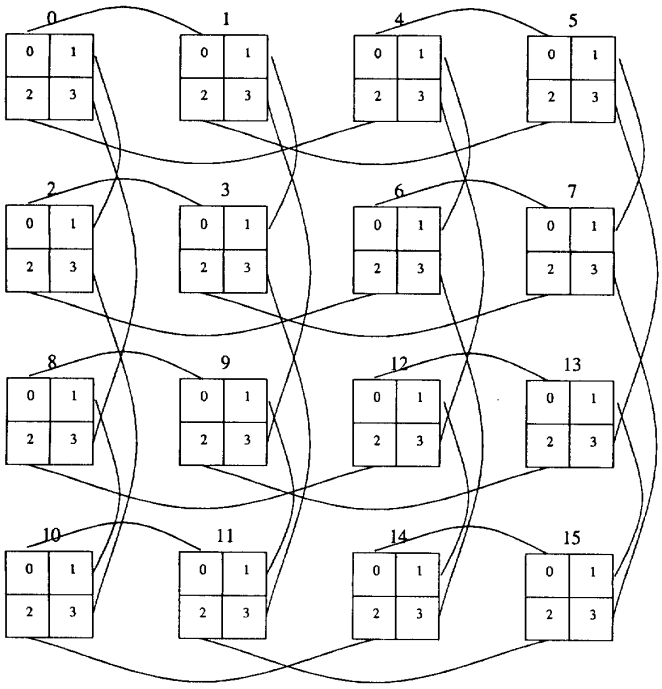

Fig. 2. The structure of the RH$(k, 2)$.

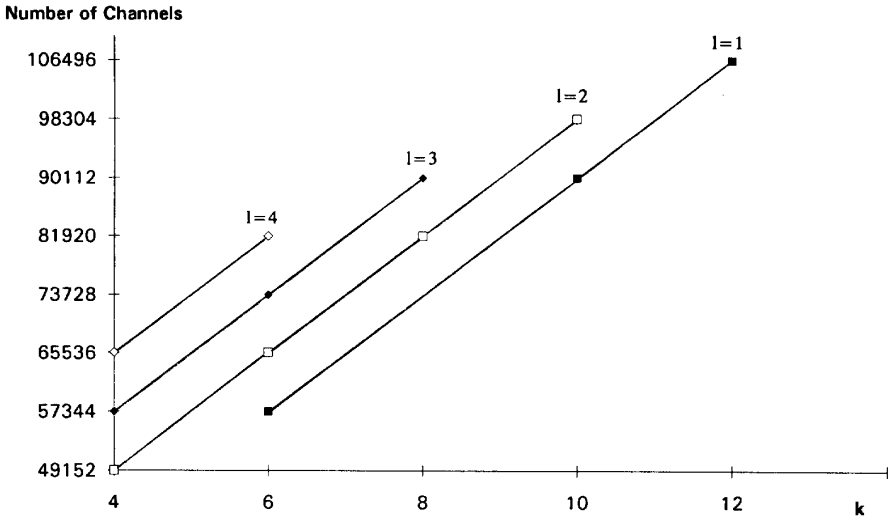

Fig. 3. The number of bidirectional channels in the RH$(k, n_1, n_2, \cdots, n_l)$ as a function of k and l, for $N = 14$.

in the RH network is equal to $\frac{N}{k+l}$. For example, for $k = 5$, $l = 2$, $n_1 = 2$, and $n_2 = 2$, each of the two networks contains 8192 PE's; the conventional hypercube contains $\frac{2^{13} \times 13}{2} = 53\,248$ bidirectional channels, and the RH network contains $\frac{2^{13} \times (5+2)}{2} = 28\,672$ bidirectional channels. The reduction in the total number of bidirectional channels for this example is about 46%, or 24 576 channels.

Fig. 3 shows the number of bidirectional channels in RH's as a function of k and l, for $N = 14$ and important values of k. This value is equal to $2^{N-1}(k + l)$. For the sake of comparison, the number of bidirectional channels in the 14-cube, which has the same number of nodes as the RH's considered for the creation of the graph, is equal to 114 688.

For the sake of simplicity, the rest of this short note focuses on reduced hypercube networks RH(k, n) with $l = 1$, and uses n instead of n_1. The condition $n \leq k$ still holds. The properties of RH's with $l > 1$ will be studied in a future paper. It is important to observe that the RH$(1, 1)$ is the ring with eight nodes, and the RH$(2, 2)$ is identical to the cube-connected cycles CCC(4). The cube-connected cycles network was presented in Section I.

Routing Alg. I (LSDF)	Routing Alg. II (forward)	Routing Alg. II (backward)
11110100;000,00	11110100;000,00	11110100;000,00
11110100;010,00	11110100;010,00	11110100;100,00
11110000;010,00	11110000;010,00	11100100;100,00
11110000;000,00	11110000;110,00	11100100;101,00
11110000;100,00	10110000;110,00	11000100;101,00
11100000;100,00	10110000;111,00	11000100;111,00
11100000;101,00	00110000;111,00	01000100;111,00
11000000;101,00	00110000;101,00	01000100;110,00
11000000;100,00	00010000;101,00	00000100;110,00
11000000;110,00	00010000;100,00	00000100;010,00
10000000;110,00	00000000;100,00	00000000;010,00
10000000;111,00	00000000;000,00	00000000;000,00
00000000;111,00		
00000000;110,00		
00000000;100,00		
00000000;000,00		
path length=16	path length=12	path length=12

Fig. 4. Routing a message from (11110101; 000, 00) to (00000000; 000, 00) in the RH(5, 3).

The CCC(4) is derived from the 4-D hypercube by substituting a ring with four nodes for each node in the hypercube. Each node in a ring then implements a connection in one of the four distinct dimensions. In general, the RH(n, n) contains the CCC(2^n) with the same number of nodes as a proper subgraph, because the ring with 2^n nodes is a proper subgraph of the n-cube building block. However, for $n < k$, the RH(k, n) is viewed as 2^{k-n} RH(n, n)'s where all nodes with the same address in the 2^{k-n} distinct RH(n, n)'s are connected to form a $(k-n)$-dimensional hypercube. The nodes' addresses in the latter hypercube become the least significant $k - n$ bits of the nodes' addresses in the RH(k, n). Therefore, another impressive property of the RH(k, n) is its capability of optimally emulating simultaneously 2^{k-n} CCC(2^n)'s.

III. Routing Algorithms

This section presents two routing algorithms for the family of RH networks that were introduced in Section II. A routing algorithm that always chooses the shortest path would be very time-consuming in the general case. In contrast, the routing algorithms proposed here require low cost and always find very efficient routes. Though the first routing algorithm has lower complexity, the second routing algorithm often chooses shorter paths for distant pairs of PE's.

A. Routing Algorithm I

Routing in the conventional hypercube is performed by a local process that changes a single bit of the source address in order to match the corresponding bit of the destination address. This process produces the address of a neighbor that receives the message. The neighbor then becomes the source, and this process is repeated until the message reaches the destination. The distance between two nodes is equal to the Hamming distance between their binary addresses.

To avoid deadlocks in the simultaneous transmission of messages between several pairs of PE's, the E-cube routing algorithm for conventional hypercubes uses dimension ordering so that messages traverse in a predetermined increasing or decreasing order the dimensions where the source and destination addresses differ. The increasing dimension order is used throughout the short note. Deadlock can occur if packets are permitted to hold resources while requesting others. These resources are packet buffers in store-and-forward and virtual cut-through switching, whereas they are channels in circuit switching and wormhole routing [22]. The description of the routing algorithm below ignores potential deadlock problems. Virtual channels are introduced at the end of this subsection to make the routing algorithm deadlock-free.

The presentation of the first routing algorithm for the RH(k, n) is verbose for the purpose of clarity. The execution steps in this routing algorithm are as follows.

Step 1: Apply the E-cube routing algorithm for conventional hypercubes to the least significant $k - n$ bits of the source address if the source and destination addresses differ in any of these bits. Use dimension ordering, starting with dimension 0 and following the increasing order. This step implements E-cube routing within the source subcube.

Step 2: If the destination was reached, then stop. Otherwise, let λ be the subscript of the most significant bit position where the current and destination addresses differ (it can be found by an XOR operation). If $\lambda < k$, then apply the E-cube routing algorithm for conventional hypercubes to the lower subfield of the current address (it comprises the bits with subscripts $k - n$ through $k - 1$) and stop. Otherwise, let m be the value represented by the lower subfield (i.e., subfield 1) of the current address. If the current and destination addresses differ in the mth bit of their upper field (i.e., field 1), then transmit the message to the PE whose address differs from the current address only in the bit with subscript $k + m$ (a direct connection exists for the implementation of this transfer), and go back to Step 2. Otherwise, if the two addresses do not differ in the mth bit of their upper field, then go to Step 3.

Step 3: Find the bit in the upper field that differs in the current and destination addresses and whose subscript in this field differs in the smallest number of bits from the n-bit value in the lower subfield of the current address. For multiple subscripts corresponding to the smallest number of bits, choose one of them at random. To find the neighbor that then receives the message, carry out E-cube routing within the k-cube building block to the nearest PE that can correct the aforementioned bit in the upper field. Go to Step 2.

An example of applying this distributed routing algorithm follows. If the source and destination addresses in the RH(7,3) are (00010010; 010, 0101) and (01110110; 101, 1101), respectively, then the routing algorithm produces addresses in this order: (00010010; 010, 1101), (00010110; 010, 1101), (00010110; 110, 1101), (01010110; 110, 1101), (01010110; 111, 1101), (01010110; 101, 1101), and (01110110; 101, 1101). Each underlined bit differs from the respective bit in the address of the preceding PE in the sequence. Theorem 1 illustrates the effectiveness of this routing algorithm.

Theorem 1: Routing algorithm I always delivers the messages to the right destinations without following any cycles.

Proof: The first step in the routing algorithm uses E-cube routing within the source subcube. E-cube routing is also applied immediately to the lower subfield if the two addresses do not differ in their upper fields. Otherwise, high priority is given to the transmission of messages over links that implement connections in level 1 (i.e., between building blocks). In addition, only one bit at a time is changed in the upper field of an address based on the result of the XOR operation between the source and destination addresses. The third step in the routing algorithm changes bits in the lower subfield using the E-cube routing algorithm for conventional hypercubes, so that the appropriate subscript is produced each time in order to traverse a dimension in level 1. The incorporation of the XOR result between the source and destination addresses for the traversal of dimensions guarantees that the messages reach the required destinations. The traversal of dimensions in level 1 at most once, as well as the incorporation of the routing algorithm for conventional hypercubes for the traversal of dimensions in level 0 (i.e., within building blocks), guarantees that messages reach their respective destinations without following any cycles. □

The complexity of the third step in this routing algorithm can be reduced by focusing each time on the least significant bit that differs

Routing Alg. I	Routing Alg. I	Routing Alg. II (forward)	Routing Alg. II (backward)
10101001;011,00	10101001;000,00	10101001;011,00	10101001;011,00
10100001;011,00	10101000;000,00	10100001;011,00	10100001;011,00
10100001;111,00	10101000;001,00	10100001;111,00	10100001;001,00
00100001;111,00	10101000;101,00	00100001;111,00	10100001;101,00
00100001;101,00	10001000;101,00	00100001;101,00	10000001;101,00
00000001;101,00	10001000;111,00	00000001;101,00	10000001;111,00
00000001;100,00	00001000;111,00	00000001;100,00	00000001;111,00
00000001;000,00	00001000;011,00	00000001;000,00	00000001;110,00
00000000;000,00	00000000;011,00	00000000;000,00	00000001;100,00
	00000000;010,00		00000001;000,00
	00000000;000,00		00000000;000,00
path length=9	path length=11	path length=9	path length=11

Fig. 5. Routing a message from (10101001; 010, 00) to (00000000; 000, 00) in the RH(5, 3).

in the upper field of the two addresses. Let least significant dimension first (LSDF) represent this modified routing algorithm. The original and modified versions of this routing algorithm are not deadlock-free. Deadlock can occur in RH's because some messages may route first in level 0 and then in level 1, whereas other messages may route first in level 1 and then in level 0. Dimension order routing, such as E-cube routing for hypercubes, is often used in regular networks to avoid deadlocks by eliminating cycles in the channel dependency graph.

A slight enhancement of the proposed routing algorithm for RH's can make it deadlock-free. This enhancement is based on a static adaptive routing algorithm for k-ary n-cubes [8]. The latter algorithm associates a dimension reversal (DR) number with each packet to avoid cycles in channel dependency graphs due to adaptive routing. The DR number of a packet is defined as the count of the number of times the packet has been routed from a channel in a higher dimension to a channel in a lower dimension (i.e., it counts dimension reversals). All packets are initialized with DR = 0. The DR number of a packet is incremented each time it results in a dimension reversal. The algorithm introduces virtual channels and divides the virtual channels of each physical channel into $r + 1$ classes, numbered 0 to r, where r represents the maximum number of dimension reversals permitted. Packets with DR $< r$ can use only virtual channels of class DR. Once a packet reaches DR $= r$, it must strictly use E-routing on the virtual channels of class r. Virtual channels are multiplexed efficiently over a physical channel in a demand-driven fashion.

This technique is incorporated here in the proposed routing algorithm, for deadlock-free routing in the RH(k, n). $r + 1$ virtual channels are associated with each physical channel, where r is defined as the largest possible number of dimension reversals. Therefore, $r = 2^n$ because this is the largest possible number of bits that need be changed in the upper field, and this results in dimension reversals. These reversals do not conform to E-cube routing with the increasing dimension order.

B. Routing Algorithm II

This subsection contains a routing algorithm that often further reduces the distance between PE's in distant building blocks. Before the routing algorithm is presented, a definition is pertinent.

Definition 1: The *binary reflected Gray code* of n bits RGC(n) is defined recursively by RGC(n) = [0•RGC$(n-1)$, 1•RGC$^{-1}(n-1)$], where • denotes concatenation, RGC$^{-1}(n-1)$ denotes the sequence derived by reversing the order of elements in the sequence RGC$(n-1)$, and RGC(1) = $[0, 1]$.

For example, RGC(2) = $[00, 01, 11, 10]$ and RGC(3) = $[000, 001, 011, 010, 110, 111, 101, 100]$. Any n-bit Gray code has the property that any two consecutive elements in its sequence of all possible 2^n n-bit numbers differ in a single bit.

The first two steps of the new algorithm are identical to those of routing algorithm I. Its third step follows.

- *Step 3:* Create a sequence of n-bit numbers that starts with the value in the lower subfield of the current node, contains the offsets of bits that differ in the upper field of the current and destination addresses, and ends with the value in the lower subfield of the destination node. Order the sequence of offsets as follows. First, create two candidate sequences that contain all of the offsets in the order in which they appear in the RGC(n) when forward or backward traversal is carried out, respectively. From the two sequences, choose the one with the smaller total number of 1-bits in the XOR results between consecutive pairs of offsets, including the current and destination subfields. This choice results in a smaller number of changes for the lower subfield, and thus reduces the total number of intermediate nodes in the path. Then apply the E-cube routing algorithm for conventional hypercubes to the lower subfield of the current address in order to go closer to the node whose lower subfield is the same as the next offset in the chosen sequence. Go to Step 2.

This routing algorithm becomes deadlock-free in a way identical to that for routing algorithm I. Let the source and destination addresses be (11110101; 000, 00) and (00000000; 000, 00), respectively, in the RH(5, 3). Fig. 4 shows the sequences of addresses produced by the two routing algorithms; the modified bit for each cycle is shown underlined. The forward and backward traversals of the RGC(3) by routing algorithm II produce paths of equal length for this example. The first routing algorithm selects a longer path if the LSDF priority scheme is used. Nevertheless, the original version of the first routing algorithm most often produces short paths. Fig. 5 illustrates such a case. The first routing algorithm produces the path shown in the first column or the path shown in the second column of Fig. 5. The difference between these two paths stems from the fact that there are two choices for the lower subfield in the address of the first intermediate node. The average performance of the second routing algorithm is better for source and destination addresses that differ in a large number of bits in their upper field. This is true because the majority of the elements in the RGC(n) will be used, whereas the RGC(n) minimizes the Hamming distance between consecutive elements. Simulation results also support this statement.

Although routing algorithm II performs better, on average, than routing algorithm I for PE's in distant k-cubes, its computation complexity is higher than that of routing algorithm I. It is important to know the time taken by each node to forward a message. Steps 1 and 2 of both routing algorithms take constant time in a single node, assuming that it takes constant time to find the *1-bits in*

the XOR result. Step 3 of routing algorithm I takes time $O(\tau)$, where τ is the total number of bits differing in the upper field of the source and destination addresses. However, Step 3 takes constant time if the LSDF priority scheme is used. Step 3 of routing algorithm II also requires $O(\tau)$ steps if the RGC(n) is stored locally. However, routing algorithm II generally takes more time than routing algorithm I, because of larger time constants. The shortest route can be found by modifying Step 3 of routing algorithm II to produce all possible sequences of the aforementioned offsets. The sequence that corresponds to the smallest total number of 1-bits in the XOR results mentioned in Step 3 of routing algorithm II is then implemented. This modification in the routing algorithm requires time $O(\tau!)$. This problem is equivalent to the well-known traveling salesman problem.

C. Finding the Distance Between Processors

The following theorem is applied for the calculation of the distance between any given pair of PE's.

Theorem 2: The distance between any given pair of PE's in the RH(k, n) is given by $= N_1 + 2 \times (N_2 - N_3) - (N_4 + N_5) + N_6$, where the N variables are defined as follows.

- N_1: The total number of 1-bits in the upper field of the result of the XOR operation between the source and destination addresses.
- N_2: The total number of 1-bits in the binary representations of the offsets in the chosen sequence; the lower subfields of the source and destination addresses are included.
- N_3: The total number of 1-bits in the results of the AND operations between all pairs of consecutive offsets in the chosen sequence.
- N_4: The total number of 1-bits in the lower subfield of the source address.
- N_5: The total number of 1-bits in the lower subfield of the destination address.
- N_6: The total number of 1-bits in the least significant $k - n$ bits of the result of the XOR operation between the source and destination addresses.

Proof: N_6 steps are required to change bits differing in the least significant $k - n$ bits of the source address, because the E-cube routing algorithm for conventional hypercubes is applied to this part of the address. N_1 bit-change operations are required in the upper field of the source address in order to match the same field of the destination address. Because of the special structure of RH's, a bit in the upper field can be changed only when the value represented by the lower subfield is equal to the bit's offset in the upper field. $2 \times (N_2 - N_3)$ is the minimum number of steps required to produce, in the chosen order, all the offsets in the sequence so that the corresponding bits in the upper field of the address can be modified. Because this evaluation using N_2 assumes that the source and destination addresses have 0 in their lower subfields, $N_4 + N_5$ is subtracted from the calculated value. □

For example, if the source and destination addresses in the RH(5, 3) are (00010001; 010, 11) and (00011000; 001, 01), respectively, and the chosen sequence is [010, 011, 000, 001], then $N_1 = 2$, $N_2 = 4$, $N_3 = 1$, $N_4 = 1$, $N_5 = 1$, $N_6 = 1$, and the distance is equal to 7. The binary addresses of the intermediate nodes are (00010001; 010, 0$\underline{1}$), (00010001; 01$\underline{1}$, 01), (0001$\underline{1}$001; 011, 01), (00011001; 01$\underline{0}$, 01), (00011001; 0$\underline{0}$0, 01), and (0001100$\underline{0}$; 000, 01), in this order.

Theorem 2 calculates the distance for a particular ordered sequence of bit offsets. The absolute shortest distance is obtained for the sequence that maximizes the value of the parameter N_3; from the parameters N_i in Theorem 2, where $i = 1, 2, \cdots, 6$, the chosen routing algorithm can affect only the value of N_3.

IV. Emulation of Conventional Hypercubes

The efficient emulation of conventional hypercubes by RH's is a very interesting problem, because of the wide variety of application algorithms that have been developed for the hypercube. For the purpose of evaluating the degradation in performance, the dilation of edges associated with such a hypercube mapping must be found. Let the function $h : G \mapsto G'$ represent the mapping of the source graph G (i.e., the hypercube) onto the target graph G' (i.e., the RH). It is a mapping of the vertices (nodes) of G to the vertices of G' in one-to-one fashion. The aforementioned performance measure is then defined as follows.

Definition 2: When two neighboring nodes from G are mapped onto two distinct nodes in G', the *dilation* of the edge connecting the two nodes in G is the length of the shortest path that connects these two nodes in G'.

The dilation measures the increase in the communication overhead when compared to one-hop data transfers in the source graph. Let the source N-dimensional hypercube and the target RH(k, n) contain the same number of nodes; that is, $N = k + 2^n$. Assume that nodes from the conventional hypercube are mapped to nodes of the RH with the same address. The following theorem presents the resultant dilation of edges.

Theorem 3: For the emulation of the $(k + 2^n)$-dimensional hypercube on the reduced hypercube RH(k, n) with the same number of nodes, the dilations of the edges incident to a single node of the hypercube are: 1 for $k + 1$ of them, and $2p + 1$ for $\binom{n}{p}$ of them, where $p = 1, 2, \cdots, n$ and $\binom{n}{p}$ represents the number of distinct p-combinations of n items.

Proof: Without loss of generality, the proof focuses on the node with address 0. Therefore, the distances of this node from the $k + 2^n$ nodes whose addresses contain a single 1-bit is found for the reduced hypercube. If the 1-bit is one of the least significant k bits of the node's address, then the dilation is equal to 1, because the considered node belongs to the same k-cube building block with the node having address 0. If the least significant bit in the upper field is equal to 1, then this dilation is also 1, because the lower subfield of address 0 contains the appropriate offset for direct data transfer. However, the dilation becomes larger if one of the higher-order bits differs. Since any of the 2^n bits in the upper field of the address may be equal to 1, to set the n-bit lower subfield to the offsets of the 1-bits in the upper field requires p steps for $\binom{n}{p}$ of the offsets, where $p = 1, 2, \cdots, n$. One more step is needed to change the corresponding bit in the upper field, and the same number of steps as above (i.e., p) are required to set the lower subfield back to its original value (i.e., all 0's). Therefore, the dilation is equal to $2p + 1$. □

For example, the dilations of the edges incident to a single node of the RH(5, 2) for the emulation of the 9-D hypercube are 1, 3, and 5 for 6, 2, and 1 edge, respectively. Similarly, the dilations of the edges incident to a single node for the emulation of the 16-D hypercube on the RH(8, 3) are 1, 3, 5, and 7 for 9, 3, 3, and 1 edge, respectively. The following two corollaries give the maximum and average dilations of edges for hypercube emulation.

Corollary 1: The maximum dilation of edges for hypercube emulation on the RH(k, n) is equal to $2n + 1$.

Proof: The proof is derived directly from Theorem 3 for $p = n$. □

Corollary 2: The average dilation of edges for hypercube emulation on the RH(k, n) is equal to the following:

$$\frac{k + (n + 1)2^n}{k + 2^n}.$$

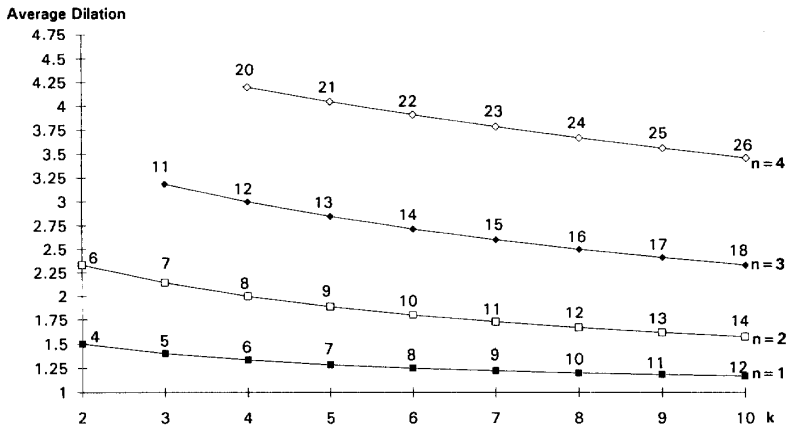

Fig. 6. The average dilation of edges for hypercube emulation on the RH(k, n) as a function of k and n, for $k + 2^n \leq 26$.

Proof: The average dilation of edges for hypercube emulation is the same as the average dilation of the edges incident to a single node of the hypercube. Based on Theorem 3, the value of the latter is equal to the following:

$$\frac{(k+1) + \sum_{p=1}^{n}(2p+1)\binom{n}{p}}{k+2^n},$$

where the denominator represents the total number of neighbors for a single node in the $(k+2^n)$-dimensional hypercube. Since we have the following conditions:

$$\sum_{p=1}^{n} p\binom{n}{p} = n2^{n-1},$$

and

$$\sum_{p=1}^{n}\binom{n}{p} = 2^n - 1,$$

the average dilation becomes equal to the following:

$$\frac{k+(n+1)2^n}{k+2^n}.$$

□

The average dilation of edges for the last two examples is 1.88 and 2.5, respectively. Fig. 6 shows the average dilation of edges for hypercube emulation as a function of k and n, with $1 \leq n \leq 4$, $2 \leq k \leq 10$, and $k+2^n \leq 26$. Therefore, the emulation of hypercubes with up to 26 dimensions is considered in Fig. 6. The numbers next to graph points represent the numbers of dimensions in the emulated hypercubes. It must be mentioned that the curves for $n = 1, 2$, and 3 represent realistic cases with respect to the actual implementation of the respective target systems with the current technology, because of the relatively low complexity of RH's. Fig. 6 proves that the average dilation of edges for hypercube emulation is relatively small. This observation guarantees small performance degradation for the implementation of hypercube algorithms on RH's.

The effect of dilation is reduced from left to right for the set of four well-known packet switching techniques: store-and-forward, virtual cut-through, circuit switching, and wormhole routing [22]. In advanced systems with wormhole routing, the physical distance between communicating nodes may have a relatively minor effect on message latency, unless it is very large. In this case, the ratio of hypercube bandwidth to RH bandwidth may prove to be a more important measure. This ratio, which is simply the ratio of the numbers of edges in the two systems, was found in Section II to be equal to $\frac{k+2^n}{k+1}$ for the RH(k, n). For example, this ratio is equal to 1.5 and 1.77 for the RH$(5, 2)$ and RH$(8, 3)$, respectively. It can be observed that RH's perform comparably to conventional hypercubes of higher cost under this performance measure also, for practical cases, where k has a relatively large value and $n = 1, 2, 3$, or 4.

V. Topological Properties of Reduced Hypercube Networks

Although it was shown that the proposed family of reduced hypercube networks possess very high potential for the efficient implementation of parallel computing techniques because of their hypercube-like topology, a more thorough investigation of their capabilities deserves our attention. Therefore, this section investigates the most important topological properties of these networks.

The *total node degree* of the RH$(k, n_1, \cdots, n_l)$ is equal to $k+l$. It becomes equal to $k+1$ for $l = 1$. The *diameter* of a topology, which is defined as the maximum of the shortest distances between all pairs of nodes, is a measure of its suitability to yield high performance. The low diameter of the hypercube network, which is equal to N for the N-dimensional hypercube, is the direct result of its high degree of interconnectivity and its regular structure. The smaller the value of the diameter, the faster the exchange of values between diametrically opposite PE's in the system. The following theorem assumes optimal routing that minimizes the distance between any pair of nodes. Therefore, the diameter represents the largest physical distance for pairs of nodes in the RH. The same diameter also results if routing algorithm II is applied.

Theorem 4: The diameter of the RH(k, n) is equal to $2^{n+1}+k-2$ for $n > 1$, and is equal to $k+3$ for $n = 1$.

Proof: The addresses of pairs of nodes having the largest distance in the RH(k, n) differ in all $k-n$ least significant bits. This contributes $k-n$ hops to the diameter, because the building block is a k-cube. All 2^n dimensions also must be traversed in level 1 for pairs of nodes that correspond to the largest distance. The bit in the upper field with offset equal to the value stored in the lower subfield of the source address is changed in a single step. The traversal of each additional dimension in level 1 also requires to change a single bit in the lower subfield, if optimal routing is considered. The latter process contributes $2 \times (2^n - 1)$ hops to the longest path. Successive lower subfields produced differ in a single bit; therefore, a Gray code sequence is generated so that the last lower subfield produced differs in a single bit from the source's lower subfield that is the *first lower*

TABLE I
COMPARING THE DIAMETERS OF REDUCED HYPERCUBES AND CONVENTIONAL HYPERCUBES

No. of Nodes	Diam. of Conv. N-D Hypercube	$RH(k,n)$	Diam. of RH's
1,024	10	RH(8,1)	11
		RH(6,2)	12
2,048	11	RH(9,1)	12
		RH(7,2)	13
		RH(3,3)	17
4,096	12	RH(10,1)	13
		RH(8,2)	14
		RH(4,3)	18
8,192	13	RH(11,1)	14
		RH(9,2)	15
		RH(5,3)	19
16,384	14	RH(12,1)	15
		RH(10,2)	16
		RH(6,3)	20
32,768	15	RH(13,1)	16
		RH(11,2)	17
		RH(7,3)	21
65,536	16	RH(14,1)	17
		RH(12,2)	18
		RH(8,3)	22
131,072	17	RH(15,1)	18
		RH(13,2)	19
		RH(9,3)	23

subfield in the sequence. Because this bit can occupy any position, $n-1$ hops are required in the destination building block in the worst case, if $n > 1$. One hop is required in this building block for $n = 1$. Therefore, the diameter of the RH(k,n) is equal to $2^{n+1}+k-2$ for $n > 1$. It is $2^{n+1}+k-1$ or $k+3$ for $n = 1$. □

For example, the diameter of the RH(6, 2) is 12. This is the distance of the node (0000; 00, 0000) from each of the nodes (1111; 00, 1111) and (1111; 11, 1111). Table I shows the diameters of several RH's and conventional hypercubes with the same numbers of nodes, for important values of k and n. This Table shows that although RH's have smaller numbers of communication channels than conventional hypercubes of similar size, their diameter is comparable to the diameter of conventional hypercubes.

The *average distance* in a regular network is defined as the ratio of the sum of the distances of all its nodes from a given node to the total number of nodes. (For the sake of simplicity, a zero distance is also included for the node that serves as the reference node.) The value of this measure for the N-dimensional hypercube is equal to the following:

$$\frac{\sum_{j=1}^{N} j\binom{N}{j}}{2^N} = \frac{N}{2},$$

because there exist $\binom{N}{j}$ nodes at distance j from each hypercube node.

An important problem is to find the average distance in the RH(k,n) for any routing algorithm that uses a strict dimension order for the upper field of addresses, because such algorithms have the lowest possible complexity. This average distance also represents an upper bound, because more sophisticated routing algorithms, such as the ones proposed here, perform better, on average. Although routing algorithm I with the LSDF priority scheme, and routing algorithm II with only forward or backward traversal of the binary reflected Gray code, apply dimension ordering, the dimension order is not strict. The prespecified dimension order is modified when an offset that appears later in the chosen sequence is generated earlier as an intermediate code while attempting to generate the next offset in the sequence. The corresponding bit in the upper field is then changed in Step 2 of both routing algorithms.

The following theorem finds the average distance in the RH(k,n), assuming a strict dimension order for the traversal of dimensions in the upper field of addresses. For the sake of efficiency, it also assumes cycle reordering of these dimensions; the order is rearranged each time in a cyclic fashion so that the first dimension is the one whose offset appears in the lower subfield of the source address.

Theorem 5: The average distance in the RH(k,n) for any routing algorithm that uses a strict order for dimensions in level 1 is given by the following:

$$\frac{\sigma+k}{2} + \frac{1}{2}\left[\sum_{i=0}^{\sigma-2}\sum_{d=1}^{\sigma-1-i}\frac{\bar{H}(c_i,c_{i+d})}{2^d} + \sum_{d=1}^{\sigma-1}\frac{\bar{H}(c_0,c_d)}{2^d}\right],$$

where the following conditions exist:

- $\sigma = 2^n$.
- c_i is the n-bit representation of the ith dimension in the chosen order, where $0 \le i \le \sigma-1$.
- $\bar{H}(c_i,c_{i+d})$ is the average Hamming distance between c_i and c_{i+d}, assuming all possible 2^n cyclic reorderings of the dimensions in level 1.

Proof: Without loss of generality, the chosen reference node has all 0's in the upper field of its address, whereas its lower subfield is equal to the index of the first dimension in the chosen order. Because all possible 2^σ combinations of $\sigma = 2^n$ bits appear in the upper field of addresses, corresponding to distinct building blocks, we can assume that the chosen order of dimensions goes from right to left in the upper field. Two bits at locations i and $i+d$, where $1 \le d, i+d \le \sigma-1$ and $0 \le i \le \sigma-1$, are 1's and all bits between them are 0's $2^{\sigma-1-d}$ times, if all possible combinations are considered. Let c_i and $H(c_i,c_{i+d})$ be the n-bit index of the ith dimension in the chosen order, and the Hamming distance between c_i and c_{i+d}, respectively. Then the contribution of the aforementioned terms to the sum of the distances of all building blocks from the reference node becomes equal to $H(c_i,c_{i+d})2^{\sigma-1-d}$. Thus, the average distance from all building blocks for all possible source nodes becomes the distance calculated in (2) at the bottom of this page. The term $\sigma 2^{\sigma-1}$ represents the total number of dimension changes in the upper field of addresses for a single reference node, whereas the last summation accounts for cases where one or more of the least significant dimensions need not be changed (i.e., for trailing 0's in the upper field of addresses). Finally, $k/2$ steps are needed,

$$\frac{\sum_{i=0}^{\sigma-2}\sum_{d=1}^{\sigma-1-i}\bar{H}(c_i,c_{i+d})2^{\sigma-1-d} + \sigma 2^{\sigma-1} + \sum_{d=1}^{\sigma-1}\bar{H}(c_0,c_d)2^{\sigma-1-d}}{2^\sigma}$$

or

$$\frac{\sigma}{2} + \frac{1}{2}\left[\sum_{i=0}^{\sigma-2}\sum_{d=1}^{\sigma-1-i}\frac{\bar{H}(c_i,c_{i+d})}{2^d} + \sum_{d=1}^{\sigma-1}\frac{\bar{H}(c_0,c_d)}{2^d}\right] \quad (2)$$

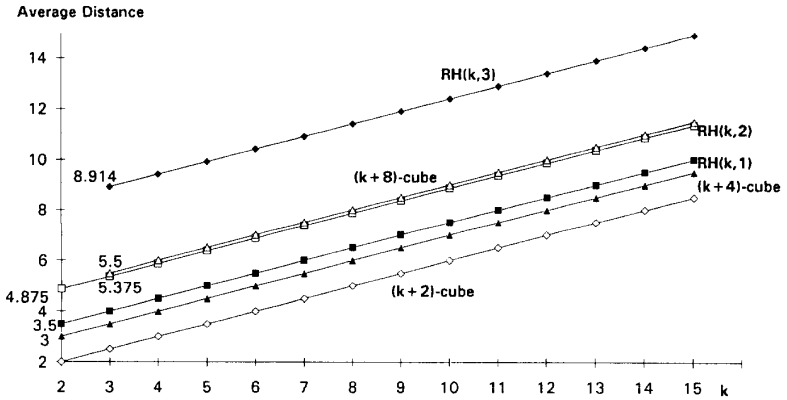

Fig. 7. The average internode distance in conventional hypercubes and RH's for important values of k and n. Strict dimension ordering with the binary reflected Gray code RGC(n) is assumed for routing in RH's.

on average, to reach the destination node in the destination k-cube building block; hence, the result. □

This average distance is proven to be very close to the average physical distance. For example, the average physical distance in the RH$(4,1)$ and RH$(6,2)$ is 3.5 and 6.625, respectively, and Theorem 5 gives 3.5 and 6.875, respectively, if the dimensions are strictly ordered according to the binary reflected Gray code. The average distance in the RH(k,n) is never lower than $\frac{\sigma+k}{2}$, which is the average distance in the respective (2^n+k)-dimensional hypercube with the same number of nodes. Fig. 7 shows the average distance in conventional hypercubes and RH's with the same numbers of nodes for important values of k and n; a strict dimension order according to the binary reflected Gray code RGC(n) is used for routing in RH's.

The development of subcube allocation techniques is required for multiuser and/or multiprogramming environments [14]. Therefore, the identification of subcubes in RH's is a very important issue. The following theorem is relevant.

Theorem 6: The RH(k,n) contains the following sets of maximal-size hypercubes:

- 2^σ k-dimensional hypercubes, and
- $2^{\sigma+n-1}$ $(k-n+1)$-dimensional hypercubes where $\sigma = 2^n$.

Proof: The building blocks are k-dimensional hypercubes. Since σ distinct subcubes are identified in each such hypercube for the implementation of σ distinct dimensions in level 1, there exist 2^σ building blocks (i.e., k-dimensional hypercubes). Subcubes from different building blocks are connected directly in pairs to add an additional dimension in level 1. Each subcube uses $k-n$ dimensions; therefore, its additional interconnections result in a $(k-n+1)$-dimensional hypercube. Because the total number of subcubes is equal to $\sigma \times 2^\sigma$, the interconnection of the aforementioned subcubes in pairs forms $2^{\sigma+n-1}$ $(k-n+1)$-dimensional hypercubes. □

VI. Conclusion

The family of RH networks were introduced in this short note. Any RH has lower complexity than the conventional hypercube with the same number of nodes because of a much smaller number of communication channels. A comparative analysis shows that RH's are capable of yielding performance comparable to that of conventional hypercubes, at much lower cost. It was also observed that any RH can optimally emulate simultaneously multiple systems with the cube-connected cycles topology. Therefore, RH's are viable candidates for the implementation of massively parallel systems. A number of very important relevant issues must be addressed in the future, with the following being the dominant ones:

1) investigation of RH's with $l > 1$,
2) embedding of frequently used topologies into RH's,
3) development of techniques for mapping application algorithms onto RH's, and
4) introduction of processor (i.e., subcube) allocation techniques for RH's.

Acknowledgment

The author is grateful to the referees for their constructive suggestions.

References

[1] S. Abraham and K. Padmanabhan, "An analysis of the twisted cube topology," in *Proc. Int. Conf. Parallel Processing*, vol. I, pp. 116–120, 1989.

[2] A. E. Amawy and S. Latifi, "Properties and performance of folded hypercubes," *IEEE Trans. Parallel Distrib. Syst.*, vol. 2, pp. 31–42, 1991.

[3] P. Banerjee, "The cubical ring connected cycles: A fault-tolerant parallel computation network," *IEEE Trans. Comput.*, vol. C-37, pp. 632–636, May 1988.

[4] T. F. Chan and Y. Saad, "Multigrid algorithms on the hypercube multiprocessor," *IEEE Trans. Comput.*, vol. C-35, pp. 969–977, Aug. 1988.

[5] H.-L. Chen and N.-F. Tzeng, "Enhanced incomplete hypercubes," *Proc. Int. Conf. Parallel Processing*, vol. I, pp. 270–277, 1989.

[6] S. B. Choi and A. K. Somani, "The generalized folding-cube," in *Proc. Int. Conf. Parallel Processing*, vol. I, pp. 372–375, 1990.

[7] P. F. Corbett, "Rotator graphs: An efficient topology for point-to-point multiprocessor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 3, pp. 622–626, 1992.

[8] W. J. Dally and H. Aoki, "Deadlock-free adaptive routing in multicomputer networks using virtual channels," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, pp. 466–475, Apr. 1993.

[9] W. J. Dally and C. L. Seitz, "The Torus Routing Chip," *Distrib. Computing*, vol. 1, pp. 187–196, 1986.

[10] S. R. Deshpande and R. M. Jenevin, "Scalability of a binary tree on a hypercube," in *Proc. Int. Conf. Parallel Processing*, 1986, pp. 661–668.

[11] K. Efe, "The crossed cube architecture for parallel computation," *IEEE Trans. Parallel Distrib. Syst.* vol. 3, pp. 513–525, 1992.

[12] A.-H. Esfahanian, L. M. Ni, and B. E. Sagan, "The twisted N-cube with application to multiprocessing," *IEEE Trans. Comput.*, vol. C-40, pp. 88–93, Jan. 1991.

[13] K. Ghose and K. R. Desai, "The HCN: A versatile interconnection network based on cubes," in *Proc. Supercomputing '89*, 1989, pp. 426–435.

[14] N.G. Haravu and S.G. Ziavras, "Processor allocation for a class of hypercube-like supercomputers," *Proc. Supercomputing '92*, 1992, pp. 740–749.
[15] W.D. Hillis, *The Connection Machine.* Cambridge, MA: MIT, 1985.
[16] C.-T. Ho and S.L. Johnsson, "Spanning balanced trees in Boolean cubes," *SIAM J. Sci. Stat. Comput.* vol. 10, pp. 607–630, July 1989.
[17] W.T.-Y. Hsu, P.-C. Yew, and C.-Q. Zhu, "An enhancement scheme for hypercube interconnection networks," in *Proc. Int. Conf. Parallel Processing*, 1987, pp. 820–823.
[18] S.L. Johnsson, "Communication efficient basic linear algebra computation on hypercube architectures," *J. Parallel Distrib. Computing*, vol. 4, no. 2, pp. 133–172, Apr. 1987.
[19] H.P. Katseff, "Incomplete hypercubes," *IEEE Trans. Comput.*, vol. C-37, pp. 604–608, May 1988.
[20] T. Kerola and A. Hartmann, "Operational analysis on hyper-rectangulars," in *Proc. Int. Conf. Parallel Processing*, vol. I, pp. 78–82, 1988.
[21] T.-H. Lai and W. White, "Mapping pyramid algorithms into hypercubes," *J. Parallel Distrib. Computing*, vol. 9, pp. 42–54, 1990.
[22] L.M. Ni and P.K. McKinley, "A survey of wormhole routing techniques in direct networks," *Comput.*, pp. 62–76, Feb. 1993.
[23] F.P. Preparata and J. Vuillemin, "The cube-connected cycles: A versatile network for parallel computation," *Commun. ACM*, vol. 24, pp. 300–309, May 1981.
[24] C.L. Seitz, "Concurrent VLSI architectures," *IEEE Trans. Comput.*, vol. C-33, no. 12, pp. 1247–1265, Dec. 1984.
[25] A. Sen, "Supercube: An optimally fault-tolerant network architecture," *Acta Informatica*, vol. 26, pp. 741–748, 1989.
[26] N.-F. Tzeng, H.-L. Chen, and P.-J. Chuang, "Embeddings in incomplete hypercubes," in *Proc. Int. Conf. Parallel Processing*, vol. I, pp. 335–339, 1990.
[27] L.D. Wittie, "Communication structures for large networks of microcomputers," *IEEE Trans. Comput.* , vol. C-30, no. 4, 264–272, Apr. 1981.
[28] S.G. Ziavras, "On the problem of expanding hypercube-based systems," *J. Parallel Distrib. Computing*, vol. 16, pp. 41–53, Sept. 1992.
[29] S.G. Ziavras and M.A. Siddiqui, "Pyramid mappings onto hypercubes for computer vision: Connection machine comparative study," *Concurrency: Practice and Experience*, vol. 5, pp. 471–489, Sept. 1993.

Task Allocation in the Star Graph

Shahram Latifi

***Abstract*—The star graph has been known as an attractive candidate for interconnecting a large number of processors. The hierarchy of the star graph allows the assignment of its special subgraphs (substars), which have the same topological features as the original graph, to a sequence of incoming tasks. The paper proposes a new code, called *star code* (SC), to recognize available substars of the required size in the star graph. It is shown that task allocation based on the SC is statically optimal. The recognition ability of a given SC or a class of SC's is derived. The optimal number of SC's required for the complete substar recognition in an n-dimensional star is shown to be 2^{n-2}.**

***Index Terms*—Allocation tree, buddy system, permutation, recognition percentage, star code**

I. Introduction

One of the attractive topologies for constructing the symmetric interconnection networks is the star graph [1], [2]. The star graph, being a member of the class of Cayley graphs, has been shown to possess appealing features including low degree of the node, small diameter, partitionability, symmetry, and high degree of fault tolerance. For this reason, recently, much research has been directed toward studying the topological properties of the star graph [7], [8], [12], [14], its fault-tolerance aspects [3], [9], [11], or implementing various algorithms on it [4], [5], [13], [15].

Parallel algorithms targeted at a given network are usually formulated with the dimension of the network as a parameter of the algorithm. This is specially true for highly regular and hierarchical networks, such as the hypercube and star graph. Depending on the size of the incoming task, one portion of the network (which preserves the topological properties of the original network) is allocated to it. Subsequent tasks then will be assigned to disjoint subnetworks, and, if no subnetwork of the required size is available, the task(s) must be queued until some tasks run to completion and make subnetworks with the required size available.

There are many issues of concern here. What should be the criteria for selecting the subnetworks such that maximum processor usage in the network is achieved? How can the recognition of an available network be guaranteed, and at what cost? In the case of network fragmentation (which is inevitable due to task relinquishment), which tasks should be migrated across the network to make room for incoming requests? This paper provides quantitative answers to some of the above questions for the star graph. The following are the main contributions of the paper:

1) proposing the star code as an efficient tool for recognizing available substars in a star graph of a given dimension,
2) proving the static optimality of the proposed allocation strategy,
3) deriving the optimal number of star codes for the complete k-star recognition for a given k, and
4) proposing a method to construct the codes for complete substar recognition.

Furthermore, some enumeration results are presented in several theorems and lemmas. With the emergence of parallel algorithms for

Manuscript received September 16, 1993; revised October 13, 1993.
The author is with the Department of Electrical and Computer Engineering, University of Nevada at Las Vegas, NV 89154 USA; e-mail: latifi@unlv.edu.
IEEE Log Number 9403106.