

The Stretched Network: Properties, Routing, and Performance

P. SHAREGHI¹ AND H. SARBAZI-AZAD^{1,2}

¹*Institute of Studies in Theoretical Physics and Mathematics (IPM)*

School of Computer Science

Tehran, Iran

E-mail: {shareghi; azad}@ipm.ir

²*Department of Computer Engineering*

Sharif University of Technology

Tehran, Iran

E-mails: azad@sharif.edu

In this paper, we study a class of interconnection networks for multiprocessors, called the Stretched- G network, which is based on a base graph G by replacing each edge of the base network with an array of processors. Two interesting features of the proposed topology are its area-efficient VLSI layout and superior scalability over the underlying base network while preserving most of its desirable properties. We conduct a general study on the topological properties of stretched networks. We first obtain their basic topological parameters and derive some embedding results. We then present optimal routing and broadcasting algorithms for such networks. We also present a unified approach to obtain the topological properties and VLSI layout of an arbitrary stretched network based on the properties of the corresponding base network G . We compare an incarnation of the stretched graph concept, namely the Stretched-Hypercube, with its equivalent hypercube and star graph from the topological and performance aspects of view. We also provide the reader with preliminary information and proper references about a wraparound variant of the stretched network called the necklace-network.

Keywords: interconnection networks, stretched networks, necklace networks, hypercubes, star graphs, topological properties, VLSI layout, performance analysis

1. INTRODUCTION

Numerous graphs have been proposed and studied as interconnection topologies for multiprocessor systems [1-10], including the hypercube, dBCube, folded Peterson cube, honeycomb, cube-connected cycles, mesh, star graph, hyper-star graph, k -ary n -cube, and incomplete k -ary n -cube.

Among the various classes of interconnection networks, scalable symmetric (or even partially symmetric) graphs with lower average node degree and lower diameter are of great interest (for their lower cost) to the designers of multiprocessor systems. On the other hand, most of the well-known interconnection networks like the hypercube, star graph, and the pyramid, suffer greatly from not being scalable; for example, in the case of the hypercube, the network size is such quantized that, for adding a single node to the network, the network size must be duplicated.

In this paper, we have tried to overcome the scalability problem by placing some processor nodes on edges of any graph G , thus achieving a far more scalable intercon-

Received June 6, 2006; revised February 12, 2007; accepted May 2, 2007.

Communicated by Sy-Yen Kuo.

nection network, called the Stretched- G . We show that, if the underlying network G is symmetric, then the resultant stretched- G network is partially symmetric. Moreover, stretched- G networks possess lower average degree than the same size G networks. Accordingly, stretched- G networks are of lower average node degree in comparison with G networks when equal network degree is considered.

Like several similar works on defining a new class of graphs based on a specific nucleus graph, *e.g.* OTIS- G [13], swapped networks [14], G -connected cycles (such as cube-connected cycles and star-connected cycles), in this paper, a general class of graphs, the Stretched- G that is based on an arbitrary graph G by replacing each edge of the base network with an array of processors, is proposed and studied. We study basic topological properties, ring embedding results, routing, broadcasting, and VLSI layout issues in the stretched- G .

The rest of paper is organized as follows. In section 2, useful definitions, notation, and some basic properties of stretched graphs are presented. Section 3 considers unicast and broadcast routing in stretched graphs. In section 4, the VLSI layout of the stretched graph is discussed. Section 5 compares the stretched- G with some other famous network graphs. Finally, section 6 concludes this study.

2. DEFINITIONS AND BASIC PROPERTIES

The following notation [15] will be used throughout the paper.

- $|G|$ or $|V(G)|$: size of a graph G , *i.e.* the number of nodes of G .
- $\|G\|$ or $|E(G)|$: number of edges of a graph G .
- $d_G(u)$: degree of a node u in a graph G .
- $d_G(u, v)$: distance between vertices u and v in G , *i.e.* the length of a shortest path between the nodes u and v .
- $\bar{D}(G)$: average distance of G , *i.e.* the average of $d_G(u, v)$ for all u and v pairs in G .
- $\text{diam}(G)$: diameter of a graph G , *i.e.* the maximum $d_G(u, v)$ for all u and v pairs in G .
- $d(G)$: average degree of G .
- $\Delta(G)$: maximum degree of G .
- $BW(G)$: bisection width of G , *i.e.* the minimum number of edges that must be removed to partition a graph G into two equal halves.

Definition 1 Let $G = (V_G, E_G)$ be an undirected graph with multiplicity of one (please note that throughout this paper we always consider graphs with multiplicity of one). The *Regular Stretched- G* network, $RS, G = (V_{RS}, E_{RS})$, is an undirected graph based on G , where each edge of G is replaced by an array of r nodes. That is

$$V_{RS} = \{(b, b', i) \mid (b = b' \ \& \ i = 0) \text{ or } (b < b' \ \& \ \langle \text{label}^{-1}(b), \text{label}^{-1}(b') \rangle \in E_G \ \& \ 0 < i \leq r)\},$$

where *label* is a bijective function as $\text{label}: V_G \rightarrow [|V_G|] = \{1, 2, \dots, |V_G|\}$.

From now on, we shall refer to every label value as a base graph node. For each $u = (b, b', i) \in V_{RS}$, b (base vertex) and b' (last vertex) are two adjacent nodes in the base graph, and i , $0 \leq i \leq r$, represents the index of node u in the array. Conventionally, we

apply zero to the index of the base graph nodes and we set $b' = b$ for such nodes; as a result, the base graph vertices can be addressed uniquely. The edge-set of the graph can be defined as $E_{RS} = \{\langle u, v \rangle \mid u = (b_1, b'_1, i_1), v = (b_2, b'_2, i_2) \in V_G\}$, where nodes u and v must satisfy one of the following conditions:

- Array edges: $b_1 = b_2, b'_1 = b'_2, |i_1 - i_2| = 1$.
- Junction edges: $b_1 = b_2, i_1 = 0, i_2 = 1$; or $b'_1 = b'_2, i_1 = r, i_2 = 0$.

Definition 2 The *irregular stretched-G network*, denoted as $IS_{r_1, r_2, \dots, r_{|E(G)|}} G$, (with $r_k, k = 1, 2, \dots, |E(G)|$ representing the length of the corresponding array), is defined similarly (to the regular stretched-G). The difference is that each array has its own length.

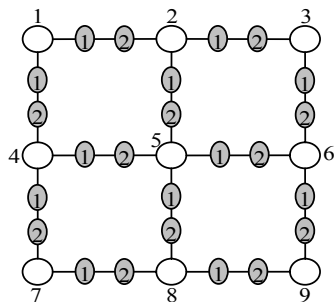


Fig. 1. The $RS_2M_{3 \times 3}$.

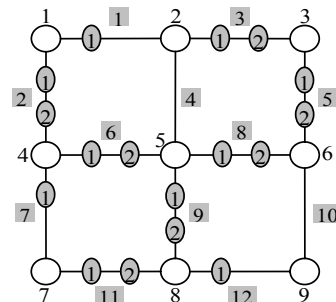


Fig. 2. The $IS_{1,2,2,0,2,2,1,2,2,0,2,1}M_{3 \times 3}$.

An instance of a regular stretched 3×3 mesh is demonstrated in Fig. 1. Fig. 2 illustrates an instance of an irregular stretched 3×3 mesh. Please notice that each number close to a white node is the label value of that base graph node.

The numbering order of the arrays is as follows. Starting with a base graph node that has the least label value, we number the array that connects the mentioned node to another node of the base graph with the next least label value, as 1. Then, the next array that connects the mentioned node to a node of the base graph with the next least label value is numbered as 2, and so on. After numbering all of such arrays, we do the same starting with the next base graph node that has the next least label value; at last, the base graph node with the greatest label value would not be used for numbering any array. Fig. 2 presents such a numbering in the $IS_{1,2,2,0,2,2,1,2,2,0,2,1}M_{3 \times 3}$. The gray numbered rectangles show the numbering order of the arrays.

Instead of replacing each edge of the base graph G with an array of processors, if we attach a processor array to each edge, we obtain a new class of graphs, called Necklace-Networks [23, 24], which can be considered as the wraparound version of the stretched-networks. Figs. 3 and 4 demonstrate typical regular and irregular necklace-networks. In addition to their scalability and area-efficient VLSI layout, necklace- G networks well preserve the structure of their underlying G networks. In other words, they embed the corresponding G network with unit congestion, dilation, and expansion. Such properties make the necklace-networks competitive to most of the traditional interconnection networks. The reader is referred to [23] and [24] for more information on necklace-networks.

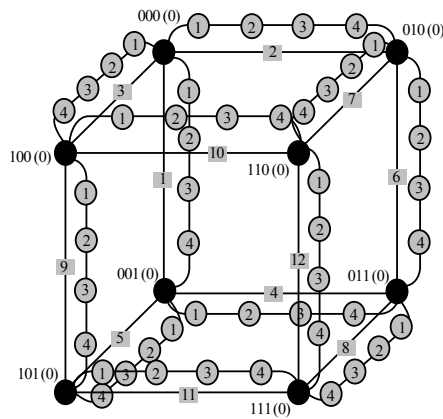


Fig. 3. The RN_4Q_3 .

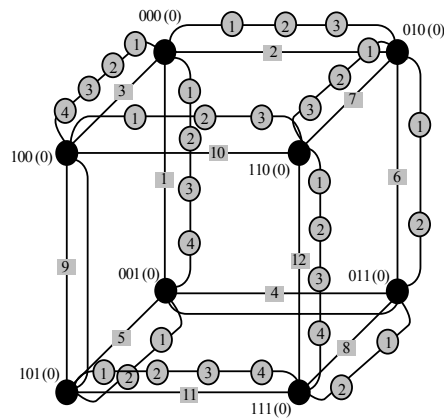


Fig. 4. The $IN_{4,3,4,0,2,2,3,2,0,3,4,4}Q_3$.

Proposition 1 [16] Let X be an undirected graph. If we replace the edges of X with independent paths between their ends (so that none of these paths has an inner vertex on another path or in X), we call the obtained graph Y a *sub-division* of X and write $Y = TX$.¹

According to proposition 1, it is obvious that the following relation holds between any graph G and its stretched variant.

Proposition 2 Let G be an undirected graph. Then, every stretched- G graph, $\forall r: RS_rG$ or $\forall r_{i, 1 \leq i \leq |E(G)|}: IS_{r_1, r_2, \dots, r_{|E(G)}}G$, is a subdivision of G . Accordingly, every stretched- G graph, $\forall r: RS_rG$ or $\forall r_{i, 1 \leq i \leq |E(G)|}: IS_{r_1, r_2, \dots, r_{|E(G)}}G$, is Homeomorphic to G .

The following propositions giving basic topological metrics of stretched- G as functions of corresponding metrics of G can be derived from Definitions 1 and 2.

Proposition 3 Let G be an undirected graph with multiplicity of one. Then, the regular stretched network, RS_rG , has the following topological properties:

- (a) $|RS_rG| = |G| + \|G\| \times r$,
- (b) $\|RS_rG\| = \|G\| \times (r + 1)$,
- (c) $BW(RS_rG) = BW(G)$,
- (d) $d(RS_rG) = \frac{d(G) \times |G| + \|G\| \times r \times 2}{|G| + \|G\| \times r}$,
- (e) $\Delta(RS_rG) = \begin{cases} \Delta(G) & : r = 0 \\ \text{Max}\{2, \Delta(G)\} & : r \neq 0 \end{cases}$
- (f) $\text{diam}(RS_rG) = \text{diam}(G) \times (r + 1)$.

Proposition 4 Let G be an undirected graph with multiplicity of one. Then, the irregular stretched network, $IS_{r_1, r_2, \dots, r_{|E(G)}}G$, has the following topological properties:

¹ The expression TX denotes an entire class of graphs; all those which, viewed as a topological space in the obvious way, are homeomorphic to X . The T in TX stands for ‘topological’.

- (a) $|IS_{r_1, r_2, \dots, r_{|E(G)|}} G| = |G| + \sum_{i=1}^{|G|} r_i$,
- (b) $\|IS_{r_1, r_2, \dots, r_{|E(G)|}} G\| = \sum_{i=1}^{|G|} (r_i + 1)$,
- (c) $BW(IS_{r_1, r_2, \dots, r_{|E(G)|}} G) = BW(G)$,
- (d) $d(IS_{r_1, r_2, \dots, r_{|E(G)|}} G) = \left(d(G) \times |G| + 2 \sum_{i=1}^{|G|} r_i \right) / \left(|G| + \sum_{i=1}^{|G|} r_i \right)$,
- (e) $\Delta(IS_{r_1, r_2, \dots, r_{|E(G)|}} G) = \begin{cases} \Delta(G) & : \forall i = 1, 2, \dots, |E(G)|, r_i = 0 \\ \text{Max}\{2, \Delta(G)\} & : \exists i = 1, 2, \dots, |E(G)|, r_i \neq 0 \end{cases}$,
- (f) Upper bound for $\text{diam}(IS_{r_1, r_2, \dots, r_{|E(G)|}} G) = \sum_{i=1}^{|G|} r'_i$, where $r'_{i, i=1, 2, \dots, |G|}$ are $|G|$ maximums of r_k , $k = 1, 2, \dots, |G|$,
- (g) Lower bound for $\text{diam}(IS_{r_1, r_2, \dots, r_{|E(G)|}} G) = \sum_{i=1}^{|G|} r''_i$, where $r''_{i, i=1, 2, \dots, |G|}$ are $|G|$ minimums of r_k , $k = 1, 2, \dots, |G|$.

It is obvious that Lemma 1 holds for any regular stretched graph based on a symmetric graph G .

Lemma 1 Let G be an undirected symmetric graph. Then, the regular stretched network, $RS_r G$, is partially symmetric (if viewed from every base graph node).

A closed walk in a graph is called an *Euler tour* if it traverses every edge of the graph exactly once. A graph is *Eulerian* if it admits an Euler tour [16]. The following theorem discusses this property in stretched graphs.

Theorem 1 Let G be an undirected graph that is Eulerian. Then, the regular (or irregular) stretched- G , $RS_r G$ (or $IS_{r_1, r_2, \dots, r_{|E(G)|}} G$), is Eulerian too.

Proof: Since the degree of all base graph vertices remain even and some vertices with even degree are added to the network, the resultant stretched-network is Eulerian.

Theorem 2 If G embeds a cycle of length L , then $RS_r G$ embeds cycles of length $L \times (r + 1) - i$ with dilation $i + 1$, where $i = 0, 1, \dots, L \times (r + 1) - 2$.

3. ROUTING AND BROADCASTING

3.1 Unicast Routing

An optimal routing algorithm for a graph $G = (V, E)$ is a function from $V \times V$ to V that associates with each pair of nodes u and w in G , a node v that is on a shortest length path from u to w , and satisfies the condition $d_G(v, w) = d_G(u, w) - 1$ [11].

Assume that we have a routing algorithm for the graph G , given by the function $NEXT_G(c, d)$ that returns the node next to the current node c on the route from c to the destination node d . Then, a similar routing function, $NEXT_{RS,G}$, for the regular stretched- G network can be obtained as follows.

Function $NEXT_{RS,G}(c, d)$

```
{/* determine the next node in the routing from (with a slight abuse of notation)  $c = (b_{1c}, b_{2c}, i_c)$  to  $d = (b_{1d}, b_{2d}, i_d)$  in an  $RS_rG$  */
  1. if  $c = d$  then return null; //  $c$  is the destination node
  2. if  $b_{1c} = b_{1d} \ \& \ i_c = 0$  then return  $(b_{1d}, b_{2d}, 1)$ ;
  3. if  $b_{1c} = b_{2d} \ \& \ i_c = 0$  then return  $(b_{1d}, b_{2d}, r)$ ;
  4. if  $b_{1c} = b_{1d} \ \& \ b_{2c} = b_{2d} \ \& \ i_c < i_d$  then return  $(b_{1c}, b_{2c}, i_c + 1)$ ;
  5. if  $b_{1c} = b_{1d} \ \& \ b_{2c} = b_{2d} \ \& \ i_c > i_d$  then return  $(b_{1c}, b_{2c}, i_c - 1)$ ;
  6. for  $1 \leq i, j \leq 2$  find  $i, j$  that  $d_G(b_{ic}, b_{jd})$  is minimum.
  7.  $k = 2 - i + 1$ ;  $l = 2 - j + 1$ ;
  8. if  $d_G(b_{ic}, b_{jd}) = d_G(b_{kc}, b_{ld})$  then
      8.1 if  $d_{RS,G}(b_{ic}, c) + d_{RS,G}(b_{jd}, d) < d_{RS,G}(b_{kc}, c) + d_{RS,G}(b_{ld}, d)$  then choose  $(b_{ic}, b_{jd})$ 
      8.2 else choose  $b_{kc}, b_{ld}$ ;
  9. else choose  $b_{ic}, b_{jd}$ ;
  10. let  $u, v$  be the chosen base graph nodes;
  11. if  $b_{1c} = u \ \& \ i_c = 0 \ \& \ u < NEXT_G(u, v)$  then return  $(u, NEXT_G(u, v), 1)$ ;
  12. if  $b_{1c} = u \ \& \ i_c = 0 \ \& \ u > NEXT_G(u, v)$  then return  $(u, NEXT_G(u, v), r)$ ;
  13. if  $b_{1c} = u \ \& \ i_c > 1$  then return  $(b_{1c}, b_{2c}, i_c - 1)$ ;
  14. if  $b_{1c} = u \ \& \ i_c = 1$  then return  $(b_{1c}, b_{1c}, 0)$ ;
  15. if  $b_{2c} = u \ \& \ 0 < i_c < r$  then return  $(b_{1c}, b_{2c}, i_c + 1)$ ;
  16. if  $b_{2c} = u \ \& \ i_c = r$  then return  $(b_{2c}, b_{2c}, 0)$ ;
}
```

The $NEXT_{RS,G}$ function routes from node $c = (b_{1c}, b_{2c}, i_c)$ to node $d = (b_{1d}, b_{2d}, i_d)$ as follows.

Case 1: If the source and destination nodes are in the same array, then $NEXT_{RS,G}$ repeatedly executes lines (4) or (5) until i_c becomes equal to i_d . This is the intuitive minimum routing for array networks.

Case 2: If the source and destination nodes are in different arrays or if the source node is in an array and the destination node is a base graph node that is not connected to the array of the source node, then $NEXT_{RS,G}$ repeatedly executes lines (6)-(10) and (13)-(16) until it reaches to the nearest base graph node, b_{1c} or b_{2c} . The remaining steps are as in case 3.

Case 3: If the source node is a base graph node, and the destination is a different base graph node or if the destination node is in an array that is not connected to the source node, then $NEXT_{RS,G}$ executes lines (6)-(12) to reach to the next array node in the path. The remaining steps are as in case 2 or case 4 (the suitable one).

Case 4: If the source node is a base graph node, and the destination is on an array connected to the source node, then line (2) or (3) is executed.

Theorem 3 If $NEXT_G$ achieves *minimum distance* routing in G , then $NEXT_{RS,G}$ achieves minimum distance routing in RS,G .

Proof: In order to prove that the proposed algorithm is optimal, let b_{1c} be at distance $j \times (r + 1)$ from the closest base graph node of the target array, namely v , and b_{2c} be at distance $(j + 1) \times (r + 1)$ from v . Moreover, let i_c be equal to r , which implies that the source node is r hops away from b_{1c} and is just one hop away from b_{2c} . The proposed algorithm chooses node b_{1c} , and it is the best choice. The distance of v from the source node when the shortest path containing b_{1c} is considered, is $j \times (r + 1) + r = jr + j + r$; the distance of v from the source node when the shortest path containing b_{2c} is considered, is $(j + 1) \times (r + 1) + 1 = jr + j + r + 2$.

3.2 Broadcast Routing

Let function $\eta_G(s, c)$ return the set of adjacent nodes to node c that receive message copies from node c during a $BROADCAST_G$ execution initiated at a source node s in G . In addition, let $PassBCReq_{RS,G}(M, c, dr)$ be a procedure that passes the broadcast request (it does not execute the broadcast request, just passes it) to the node next to c in direction dr in the current array, until node c is a base graph node; then, it initiates a $BROADCAST_{RS,G}(M, c, c, 0)$.

A procedure for broadcasting message M originated at a source node like s in regular stretched- G network can be obtained in a straightforward manner. The $BROADCAST_{RS,G}$ procedure, invoked at node $c = (b_c, b'_c, i_c)$ to contribute in broadcasting message M originated at source $s = (b_s, b'_s, i_s)$, works as follows.

Case 1: If the current node is the source node and the source node is within an array (*i.e.* $0 < i_s \leq r$), then the broadcast request is passed toward the nearest base graph node, with a proper direction parameter. This is done in line 1.

Case 2: If the current node is a base graph node, then a broadcast request is sent to every $(b_c, u, 1)$ or (u, b_c, r) where $u \in \eta_G(b_s, b_c)$, with a proper direction value. Line 2 covers case 2 in the above code.

Case 3: If the current node is not the source node, but an end node of an array (*i.e.* $i_c = 1$ or $i_c = r$), and the broadcast direction is toward the neighboring base graph node, then a broadcast request is sent to the neighboring base graph node. Case 3 occurs when line 3 or 4 is executed.

Case 4: If the current node is not the source node, but is within an array, then line 5 is executed to continue broadcasting in the proper direction.

Procedure $BROADCAST_{RS,G}(M, s, c, dr)$

{/* invoked at node $c = (b_c, b'_c, i_c)$ to contribute in broadcasting message M originated at

source node $s = (b_s, b'_s, i_s)$. dr is the direction in which the message is advancing in the current array (-1 denotes backward, $+1$ denotes forward, zero is used for going to a base graph node) */

```

1. if  $c = s$  &  $0 < i_s \leq r$  then
  (a) else if  $i_s \leq \lceil \frac{r}{2} \rceil$  then PassBCReq $_{RS,G}(M, s, -1)$ ; return;
  (b) else if  $i_s > \lceil \frac{r}{2} \rceil$  then PassBCReq $_{RS,G}(M, s, +1)$ ; return;
2. if  $i_c = 0$  then
  for every  $u \in \eta_G(b_s, b_c)$  do
    if  $b_c < u$  then
      BROADCAST $_{RS,G}(M, s, (b_c, u, 1), +1)$ ;
    else if  $b_c > u$  then
      BROADCAST $_{RS,G}(M, s, (u, b_c, r), -1)$ ;
  return;
3. if  $i_c = 1$  &  $dr = -1$  then
  BROADCAST $_{RS,G}(M, s, (b_c, b_c, 0), 0)$ ;
  return;
4. if  $i_c = r$  &  $dr = +1$  then
  BROADCAST $_{RS,G}(M, s, (b'_c, b'_c, 0), 0)$ ;
  return;
5. BROADCAST $_{RS,G}(M, s, (b_c, b'_c, i_c + dr), +dr)$ ;
}

```

Theorem 4 If $BROADCAST_G$ delivers a single copy of the message to each node in G , then so does $BROADCAST_{RS,G}$ in RS,G . In addition, if $BROADCAST_G$ traces a spanning tree of G whose height is minimum, then $BROADCAST_{RS,G}$ traces a spanning tree of RS,G whose height is at most equal to the height of a minimum height spanning tree of RS,G plus $2 \times \lceil r/2 \rceil$.

Proof: Both broadcast algorithms, $BROADCAST_G$ and $BROADCAST_{RS,G}$, are fully characterized by the function $\eta_G(s, c)$. The intuition behind the algorithm is to pass the broadcast request to the nearest base graph node when the initial request is issued at an array node, and then to issue a new broadcast request with the current base graph node as the source node. In this way, in the worst case where the initial source node is located at the middle of an odd-sized array, it takes $\lceil r/2 \rceil$ steps to reach the nearest base graph node and issuing a new broadcast request. In addition, it takes $\lceil r/2 \rceil$ more steps for the broadcast request to reach the initial source node. Therefore, we have taken $2 \times \lceil r/2 \rceil$ more steps for broadcasting the message and creating the spanning tree.

4. THE VLSI LAYOUT

Many graph layout problems are originally motivated as simplified mathematical models of VLSI layout. Given a set of modules, the VLSI layout problem consists of placing the modules on a board in a non-overlapping manner and wiring together the

terminals on the different modules according to a given wiring specification and in such a way that the wires do not interfere among them. There are two stages in VLSI layouts: *placement* and *routing*. The *placement problem* consists of placing the modules on a board; the *routing problem* consists of wiring together the terminals on different modules that should be connected [17].

A simple physical realization of any network topology is a single-row wiring layout. We extend the single-row layout scheme to discuss about stretched- G networks. Some important points about this layout scheme are as follows.

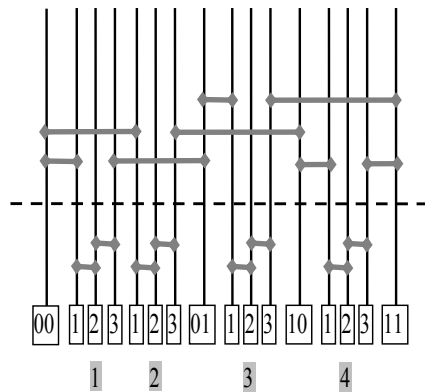
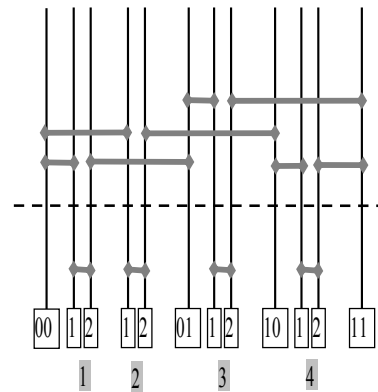
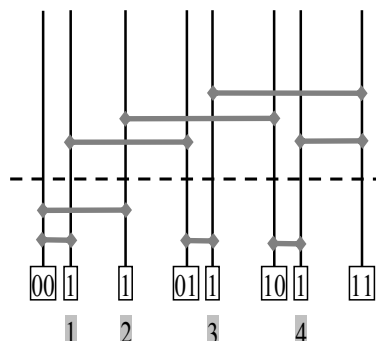
- The nodes of the network are placed in a row, with wiring tracks running parallel to the row of nodes.
- The number of wiring tracks required for the layout is a function of the order in which the graph nodes are placed in a row.
- The order that results in the least tracks required is termed optimal. Finding the optimal order requires solving the matrix permutation problem (MPP). The MPP has been shown to be NP-hard [18].

Theorem 5 Let the number of wiring tracks sufficient for the single-row wiring layout of a given graph G , according to a given node placement order O , be $t_o(G)$. Then, the number of wiring tracks sufficient for the single-row wiring layout of a regular stretched- G , RS_rG , when the node placement order O is used, is given by:

$$t_o(RS_rG) = \begin{cases} t_o(G) & : r = 0 \\ t_o(G) + \Delta(G) & : r = 1 \\ t_o(G) + 1 & : r = 2 \\ t_o(G) + 2 & : r > 2 \end{cases}$$

Proof: We extend the single-row layout, based on node placement order O , of graph G to be suitable for the stretched- G as follows. All of the base graph nodes are placed in a single-row according to the placement order O . However, the point is that, besides each base graph node, we place the arrays in which the above node acts as the base for indexing. The arrays are placed in the following order. First, the array that connects the mentioned node to a node of the base graph, with the least label value, is placed in the row. Then, the next array that connects the mentioned node to a node of the base graph, with the next least label value, is placed in the row, and so on. After placing all of the mentioned arrays, the next base graph node is placed in the row. Fig. 5 demonstrates the single-row VLSI layout of a typical regular stretched network, R_3S_2Q , based on the 2-D hypercube.

As it is shown in Fig. 5, we can separate the tracks needed for wiring, into two groups: first, the tracks needed for wiring inside each array (*i.e.* the tracks below the dashed line); second, the tracks needed for connecting arrays to the base graph nodes (*i.e.* the tracks above the dashed line).

Fig. 5. The VLSI layout of the RS_3Q_2 .Fig. 6. The VLSI layout of the RS_2Q_2 .Fig. 7. The VLSI layout of the RS_1Q_2 .

In this manner, the number of tracks required for wiring regular stretched- G is roughly equal to the number of tracks required for wiring graph G . The remaining difference is a matter of placing the wiring tracks needed for wiring inside the arrays. There are four different cases:

- When r is greater than two (Fig. 5), we need two rows for placing the tracks.
- When r is equal to two (Fig. 6), we can place the tracks in one row.
- When r is equal to one (Fig. 7), we should take another approach. First, for all of the base graph nodes, we place the tracks needed to connect them to their right side neighboring arrays. Then, for all of the arrays, we place the tracks needed to connect them to their other end neighboring base graph node. In this way, we will have a total of approximately $\Delta(G)$ additional tracks, which is a rough upper bound.
- When r is equal to zero, the stretched graph is equivalent to its base graph; hence, we do not need any extra tracks.

Consider the single-row one-dimensional implementation of an RS_rG graph, where each node is a square functional unit, s units on a side. On the other hand, each edge consists of b parallel wires, each wire requiring a width of w units. Then, the *wiring channel length* is $|V_{RS_rG}| \times s$, and the *wiring channel width* is $b \times w \times t_O(RS_rG)$.

The *area of the wiring channel* is the product of its width and length, *i.e.* $|V_{RS,G}| \times s \times b \times w \times t_O(RS_r G)$. The *wiring channel overhead ratio* (the ratio of the total system area to the area of the functional units) can be given as

$$\frac{|V_{RS,G}| \times s \times [b \times w \times t_O(RS_r G) + s]}{|V_{RS,G}| \times s^2} = \frac{b \times w \times t_O(RS_r G) + s}{s}$$

5. COMPARISON TO OTHER NETWORKS

5.1 Comparison of Topological Merits

In this section, we compare stretched- G with the same size G graph considering some topological properties. To do so, as a case study, we compare the stretched-hypercube with its equivalent hypercube, star graph, and cube-connected-cycles, in order to demonstrate that the stretched-hypercube not only has many topological merits over the hypercube, but also is superior to the equivalent star graph, and has comparable properties if not better than the cube-connected-cycles.

Since stretched- G has many nodes with degree 2, we expect a far lower average node degree than the equivalent G network. In addition, since the stretched- G has lower bisection-width than its equivalent G graph, the stretched network must have fewer channels, which leads to lower implementation cost. Moreover, following Theorem 5, we expect fewer wiring tracks for the stretched- G than the equivalent G network.

In Figs. 8-11, the stretched-hypercube network is compared with the hypercube network, the star graph, and the cube-connected-cycles network. One of the interesting features of the stretched-hypercube, not shown in the above diagrams, is its better scalability over the other networks compared here. Since the hypercube network size, 2^n , grows excessively fast, it is not well scalable from a hardware cost point of view. The size of the cube-connected-cycles network, $n2^n$, grows even faster. The star graph is not scalable either, its network size, $n!$, grows dramatically.

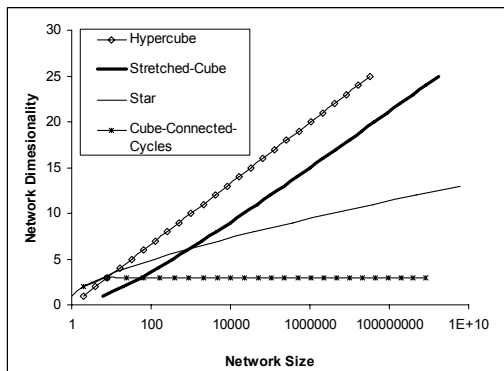


Fig. 8. Network dimensionality as a function of network size (here we have assumed $r = 4$ in the stretched hypercube network).

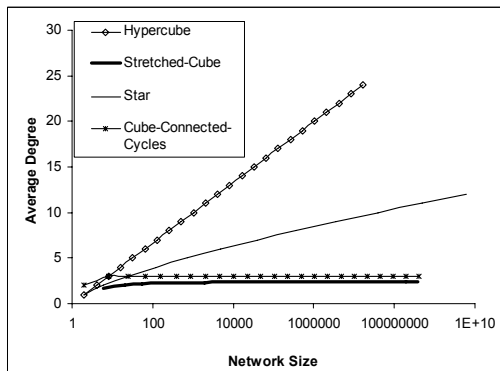


Fig. 9. Average node degree as a function of network size (here we have assumed $r = 4$ in the stretched hypercube network).

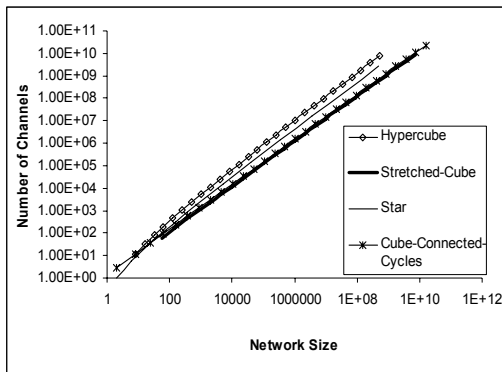


Fig. 10. Number of channels (edges of the network) as a function of network size (here we have assumed $r = 4$ in the stretched hypercube network).

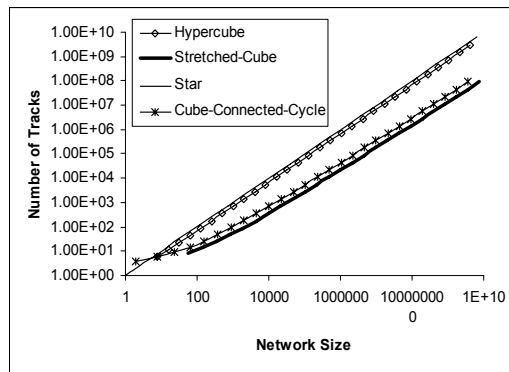


Fig. 11. Number of tracks in VLSI Layout as a function of network size (here we have assumed $r = 4$ in the stretched hypercube network).

Since the maximum number of tracks needed to interconnect nodes of the n -star is roughly $n!$ (see Theorem 9 in [21]) and is $\lfloor 2^{n+1}/3 \rfloor + \lfloor n/2 \rfloor$ for the n -cube [12], both networks suffer greatly from having area-inefficient VLSI layouts. Whereas, the stretched hypercube and the cube-connected-cycles network have reasonably better VLSI layouts.

The number of tracks needed to interconnect nodes in a cube-connected-cycles network, $\lfloor 2^{n+1}/3 \rfloor + \lfloor n/2 \rfloor + 3$ can be achieved in a very similar way that authors have used in [25] for the stretched-hypercube. That is, we can do the wiring of all of the cycles using only 3 tracks, only one more (a track for the wraparound link) than the number of tracks needed to interconnect the nodes of all the arrays in stretched hypercube. Then to interconnect neighboring cycles according to the hypercube topology, we need $\lfloor 2^{n+1}/3 \rfloor + \lfloor n/2 \rfloor$ tracks.

Although the star graph and cube-connected-cycles have better dimensionality with respect to their sizes (when r is fixed and small in the stretched hypercube) for large networks, all other results reveal that the stretched-hypercube is superior to the star graph and hypercube, and has comparable properties if not better than the cube-connected-cycles. Note that the stretched hypercube is better than its equivalent star graph even from dimensionality point of view when the network size is up to 1000s nodes, which is a realistic size for a multicomputer today.

5.2 Discrete Event-Driven Simulation

The XMulator [26] was used to mimic the detailed operation of the stretched networks and to perform simulation experiments. XMulator is an event-based flit-level simulator which can provide various detailed results. For the sake of this study, we have simulated a minimal routing algorithm for wormhole-switched stretched hypercubes.

In *wormhole switching*, a message packet is broken up into (say F) flits. The *flit* (flow control digit) is the unit for which the message flow control is implemented. Input/output buffers at a router are typically large enough to store a few flits. In this switching technique, the message is pipelined through the network at flit level [20]. A flit itself is

usually broken into some smaller digits, called the *phits* (or physical digit), for transmission over network channels. That is a flit size is usually a multiple of phit size. A phit can be transmitted over a network channel in one channel cycle. We have also considered the capability of using virtual channels in order to maximize network throughput and provide flexibility to design new routing algorithm for future work.

Virtual channels associated to a physical channel share the bandwidth of the physical channel and are demand time-multiplexed on the physical channel. Moreover, only virtual channels have messages to transmit use the physical channel in a round-robin manner. In our simulation experiments, the delays for switching and routing are ignored (like many other similar studies in the literature) and only the delay of physical channels is considered and assumed to be 1 cycle (*i.e.* each phit is transmitted over a physical channel in 1 cycle). Also, message consuming at the destination node is not bandwidth limited to avoid performance bottlenecks. The performance metrics calculated and reported here is the *average message latency* which is drawn as a function of the message generation rate at each node (messages per cycle). It is the average of all message latencies transmitted over the network for at least 1,000,000 messages. The first 10% of messages are excluded from the calculation to avoid warm up effects.

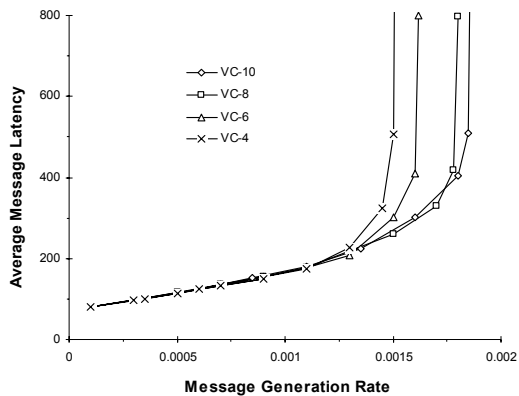


Fig. 12. Average message latency in a wormhole-switched RS_2Q_7 stretched-hypercube for message length of $F = 64$ flits and $VC = 4, 6, 8,$ and 10 virtual channels per physical channel.

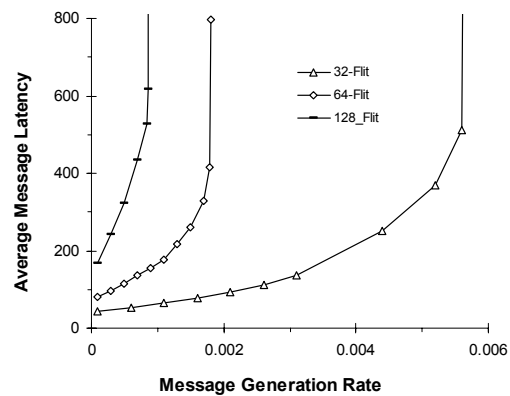


Fig. 13. Average message latency in a wormhole-switched RS_2Q_7 stretched-hypercube for message length of $F = 32, 64,$ and 128 flits and $VC = 8$ virtual channels per physical channel.

Fig. 12 shows the effect of the number of virtual channels on the overall performance when the message length is assumed to be $F = 64$ flits (or 64 phits; here the flit size is assumed to be 1 phit) in a RS_2Q_7 stretched hypercube. As can be seen in the figure, increasing the number of virtual channels, from $VC = 4$ to $VC = 10$, results in a better performance. This is a rational effect because when we increase the number of virtual channels, there are some extra ways for the messages to continue their journey to the destination nodes, and hence the generation rate at which network saturation happens is increased.

Let us now consider the effect of message length on the overall performance. To this end, in our simulation experiments, we fix the number of virtual channels per physi-

cal channel to $VC = 8$ in a RS_2Q_7 stretched hypercube. As can be seen in Fig. 13, increasing the message size increases the average message latency and results in earlier saturation for the network.

5.3 Performance Comparison under Implementation Constraints

Multiprocessor systems implemented using the VLSI technology are wire-limited; the wiring density of the interconnection network determines the overall cost and performance of the system [19]. In this section, as a case study, we compare the performance of the stretched-hypercube with the hypercube network using the following assumptions:

- *Bisection width*, B_w , is used as a parameter of wiring density.
- Each network has its own channel width, C_w (*i.e.* the capacity of every channel of the network).
- *Bisection bandwidth* is defined as $BB_w = B_w \times C_w$.
- Since bisection bandwidth can be incorporated as a measure of network implementation cost, we compare two same-sized networks, with equal bisection bandwidths.
- *Pinout* is defined as $pinout(G) = d(G) \times C_w(G)$, where $d(G)$ is the average degree of the graph G .
- Pinout can be used as another measure of network implementation cost when applying multi-chip implementation techniques.

Assuming equal bisection bandwidth for two same-sized networks, we can calculate the ratio of their phit sizes. In the following, Q stands for hypercube, and SQ stands for stretched-cube:

$$BB_w(SQ) = BB_w(Q)$$

$$B_w(SQ) \times C_w(SQ) = B_w(Q) \times C_w(Q).$$

The ratio of channel cycle times is given by

$$T_c(Q)/T_c(SQ) = C_w(SQ)/C_w(Q) = B_w(Q)/B_w(SQ).$$

Considering $C_w(SQ) = \text{flit size}$, and a unit channel cycle time, we can calculate the time to communicate a flit over the channel of equivalent hypercube. Fig. 14 compares the performance of an 8-dimensional hypercube (with 256 nodes) and a RS_7Q_4 (with 240 nodes) and a RS_8Q_4 (with 272 nodes) under constant bisection bandwidth constraint. The message length is fixed at 64 flits and the number of virtual channels per physical channel is assumed to be $VC = 8$. The figure shows that the stretched hypercube performs better when bisection bandwidth constraint is used.

We can also compare two same-sized networks with equal pinouts. In this way, we can calculate the ratio of their phit sizes as

$$pinout(SQ) = pinout(Q)$$

$$d(SQ) \times C_w(SQ) = d(Q) \times C_w(Q).$$

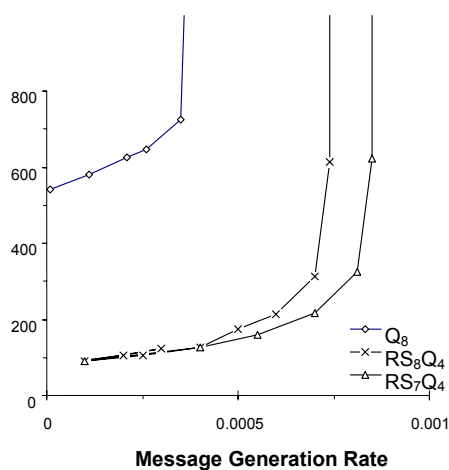


Fig. 14. Comparison of an 8-dimensional hypercube and two (almost) same-sized stretched hypercubes under constant bisection bandwidth constraint with message length $F = 64$ flits and $VC = 8$ virtual channels per physical channel.

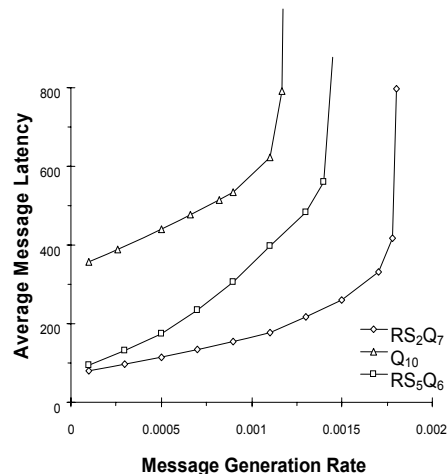


Fig. 15. Comparison of a 10-dimensional hypercube and two same-sized stretched hypercubes under constant pinout constraint with message length $F = 64$ flits and $VC = 8$ virtual channels per physical channel.

The ratio of channel cycle times is given by

$$T_c(Q)/T_c(SQ) = C_w(SQ)/C_w(Q) = d(Q)/d(SQ).$$

Again, considering $C_w(SQ) = \text{flit size}$, and a unit channel cycle time, we can calculate the time to communicate a flit over the channel of equivalent hypercube. Fig. 15 compares the performance of a 10-dimensional hypercube and a RS_5Q_6 and a RS_2Q_7 (all with 1024 nodes) under constant pinout constraint. The message length is fixed at 64 flits and the number of virtual channels per physical channel is assumed to be $VC = 8$. This figure also reveals that the stretched hypercube performs better when pinout constraint is used. Comparing Figs. 14 and 15 reveals that the bisection bandwidth constraint has a more noticeable impact on the comparative performance of stretched networks than the pinout constraint.

6. CONCLUSIONS

In this paper, we introduced a general idea of constructing a new class of networks called stretched- G . The Stretched- G is an interconnection network that is based on the base graph G by replacing each edge of the base network with an array of processors.

We conducted a general study on the topological properties of the stretched graphs as well as some embedding results. Optimal routing and broadcasting algorithms are also of the addressed issues in this paper. We also demonstrated a unified approach to obtain the topological properties, and VLSI layout of an arbitrary stretched network based on the properties of the corresponding base network graph G . With respect to the theorems

and results given in this paper, we can conclude that

- Scalability is one of the most important features of stretched-networks.
- Stretched- G has a very low average degree in comparison with graph G , which results in a less pinout cost and thus less network cost.
- The stretched- G could be laid out on a more area efficient VLSI layout in comparison with the G network.
- The stretched- G outperforms the underlying graph G when implementation constraints (either single-chip or multi-chip) are taken into account.

REFERENCES

1. Y. Saad and M. H. Schultz, "Topological properties of hypercubes," *IEEE Transactions on Computers*, Vol. 37, 1998, pp. 867-872.
2. C. Chen, D. P. Agrawal, and J. R. Burke, "dBCube a new class of hierarchical multiprocessor interconnection networks with area efficient layout," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 4, 1993, pp. 1332-1344.
3. S. Ohring and S. K. Das, "Folded Peterson cube networks: new competitors for the hypercubes," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 7, 1996, pp. 151-168.
4. I. Stojmenovic, "Honeycomb networks: topological properties and communication algorithms," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 8, 1997, pp. 1036-1042.
5. F. P. Preparata and J. Vuillemin, "The cube-connected cycles: a versatile network for parallel computation," *Communications of the ACM*, Vol. 24, 1981, pp. 300-309.
6. H. Sarbazi-Azad, "On some combinatorial properties of meshes," in *Proceedings of the 7th International Symposium on Parallel Architectures, Algorithm and Networks*, 2004, pp. 117-122.
7. S. B. Akers, D. Harel, and B. Krishnamurthy, "The star graph: an attractive alternative to the n -cube," in *Proceedings of International Conference on Parallel Processing*, 1987, pp. 393-400.
8. H. O. Lee, J. S. Kim, E. Oh, and H. S. Lim, "Hyper-star graph: a new interconnection network improving the network cost of the hypercube," in *Proceedings of EurAsia-ICT 2002*, LNCS 2510, 2002, pp. 858-865.
9. H. Sarbazi-Azad, M. Luld-Khaoua, L. M. Mackenzie, and S. G. Akl, "On the combinatorial properties of k -ary n -cubes," *Journal of Interconnection Networks*, Vol. 5, 2004, pp. 79-91.
10. B. Parhami and D. M. Kwai, "Incomplete k -ary n -cube and its derivatives," *Journal of Parallel and Distributed Computing*, Vol. 64, 2004, pp. 183-90.
11. K. Day and A. Al-Ayyoub, "Product-closed networks," *Journal of Systems Architectures*, Vol. 45, 1998, pp. 323-338.
12. A. Patel, A. Kusalik, and C. McCrosky, "Area-efficient VLSI layout for binary hypercubes," *IEEE Transactions on Computers*, Vol. 49, 2000, pp. 160-169.
13. K. Day and A. E. Al-Ayyoub, "Topological properties of OTIS-networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, 2002, pp. 359-366.

14. B. Parhami, "Some properties of swapped interconnection networks," in *Proceedings of the International Conference on Communications in Computing*, 2004, pp. 93-99.
15. R. Diestel, *Graph Theory*, Electronic Edition, Springer-Verlag, 2000, pp. 2-8.
16. R. Diestel, *Graph Theory*, Electronic Edition, Springer-Verlag, 2000, pp. 17-19.
17. J. Díaz, J. Petit, and M. Serna, "A survey of graph layout problems," *ACM Computing Surveys*, Vol. 34, 2002, pp. 313-356.
18. R. H. Mohring, "Graph problems related to gate Matrix layout and PLA folding," *Computing Supplementum*, Vol. 7, 1990, pp. 17-51.
19. W. J. Dally, "Performance analysis of k -ary n -cube interconnection networks," *IEEE Transactions on Computers*, Vol. 39, 1990, pp. 775-785.
20. J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks an Engineering Approach*, Morgan Kaufmann Publishers, 2003.
21. A. Al-Ayyoub and K. Day, "The hyperstar interconnection networks," *Journal of Parallel and Distributed Computing*, Vol. 48, 1998, pp. 175-199.
22. P. Shareghi and H. Sarbazi-Azad, "Topological properties of stretched graphs," in *Proceedings of the 4th ACS/IEEE International Conference on Computer Systems and Applications*, 2006, pp. 123-126.
23. P. Shareghi and H. Sarbazi-Azad, "Topological properties of necklace networks," in *Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms and Networks*, 2005, pp. 40-45.
24. M. Monemizadeh and H. Sarbazi-Azad, "The necklace-hypercube: a well scalable hypercube-based interconnection network for multiprocessors," in *Proceedings of the ACM Symposium on Applied Computing*, 2005, pp. 729-733.
25. P. Shareghi and H. Sarbazi-Azad, "The stretched-hypercube: a VLSI efficient network topology," in *Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms and Networks*, 2005, pp. 462-467.
26. A. Nayebi, S. H. Azad, A. Shamaei, and S. Meraji, "XMulator: an object oriented XML-based simulator," <http://www.XMulator.org>, 2006.



Pooya Shareghi is currently a graduate student in Computer Science at the University of Georgia in Athens, GA, USA. His research interests include interconnection networks, parallel processing, evolutionary computation, data mining, and applications of graph theory.



Hamid Sarbazi-Azad received his B.Sc. degree in Electrical and Computer Engineering from Shahid-Beheshti University, Tehran, Iran, in 1992, his M.Sc. degree in Computer Engineering from Sharif University of Technology, Tehran, Iran, in 1994, and his Ph.D. degree in Computing Science from the University of Glasgow, Glasgow, UK, in 2002. He is currently associate professor of computer engineering at Sharif University of Technology, and heads the School of Computer Science of the Institute for Studies in Theoretical Physics and Mathematics (IPM), Tehran, Iran. His research interests include high-performance computer architectures, NoCs and SoCs, parallel and distributed systems, performance modeling/evaluation, graph theory and combinatorics, wireless/mobile networks, on which he has published more than 150 refereed conference and journal papers. Dr. Sarbazi-Azad has served as the editor-in-chief for the CSI Journal on Computer Science and Engineering since 2005, editorial board member and guest editor for several special issues on high-performance computing architectures and networks (HPCAN) in related journals. Dr. Sarbazi-Azad is a member of ACM and CSI (Computer Society of Iran).