Theoretical Research on Networks: Models and Methodology[*]

Arnold L. Rosenberg
Department of Computer Science
University of Massachusetts
Amherst, Mass. 01003, USA
rsnbrg@cs.umass.edu

Abstract

This note is based on my invited talk at the *4th Intl. Colloq. on Structural Information and Communication Complexity* (*SIROCCO'97*). In it, I discuss a few "meta-issues" concerning theoretical research on networks, and I illustrate these issues with some recent research of mine (so that all critical remarks can be focused in my direction). My central foci are the selection of problems and models, and the tools for studying them.

1 Crucial Concerns for Theoretical Researchers

Technology-focused theoretical computer scientists work in a strange laboratory. On the one hand, the tools they work with are supplied by the field of mathematics. On the other hand, the objects that they work on come from a very different constituency, the practitioners in the areas of computer and communication technology. Researchers in this area face a tantalizing temptation to cleave to the elegance-seeking paradigm of mathematics in their work—at the risk of producing results that apply only in a *gedanken* universe. My intention is to describe here some of the dangers that I see in our research world and to suggest, via examples, ways to avoid some of these dangers. I choose my exemplary studies just from my own work (with valued collaborators) so that any criticism that may be engendered by my thesis will be directed solely at me.

1.1 Fundamental Research Questions

I begin my remarks with two research-oriented questions that I view as among the most important that a theoretician can ask—for only a theoretician can ask them.

What really matters? This question really encapsulates the power of the theoretical researcher's ability to abstract from the details of current technology. It allows the researcher to question what

[*]The research reported herein supported in part by NSF Grants CCR-92-21785 and CCR-97-10367.

features of a

$$\left.\begin{array}{l}\text{computer architecture}\\ \text{programming language}\\ \text{communication network}\\ \\ \qquad\vdots\end{array}\right\}\text{ affect its }\left\{\begin{array}{l}\text{computational power or efficiency}\\ \text{power or efficiency of expression}\\ \text{ability to handle types and amounts}\\ \quad\text{of traffic}\\ \qquad\vdots\end{array}\right.$$

What if ...? This question embodies the theoretical researcher's ability (and responsibility?) to attempt to anticipate technology, with an eye toward informing the technologist of the consequences of probing in various directions. When possible, this is a very efficient use of (human and technological) resources, as the theoretician typically works in a small team, using pencils, paper, and workstations, whereas the technologist may require extensive costly equipment and a large team to study the same directions.

The final question I wish to raise concerns a relatively new research tool of the technology-focused theoretical researcher—experimentation. Whereas the more abstractly focused theoretician has traditionally been satisfied to study optimization problems using purely mathematical techniques—either establishing the probable computational intractability of a problem or, when possible, establishing its efficient approximability—the technology-focused theoretician must derive answers even when the tools of mathematics do not (seem to) provide them. This leads the technology-focused theoretician to the realm of experimental studies, a realm that has been dominated by practicioner. One major danger for the theoretical researcher in this new domain is to forget Theory's role in the grand scheme of things and apply experimentation with the rather limited goals of the practicioner—to establish specific facts. The big need that I see is for the theoretical researcher to continually address the issue of:

how to devise experiments that show *why* phenomena occur, not just *that* they do.

I shall expand on the three questions I have raised in Sections 2-4.

1.2 Responsible Research Guidelines

The basic issue here is that a researcher must pay close attention to the appropriateness of his/her tools for the goals of his/her investigation. The first two sample questions that I believe a theoretical researcher should continually ask focus on what one might call the "honesty" of a theoretical study.

- Will the "eventually" implicit in my asymptotic analyses eventuate?
- Do the speculative models that I study plausibly anticipate future technology, or do they merely create fantasy technology that can exist only as a mathematical construct?

The final question, as in the previous subsection, concerns the truly difficult area of empirical studies—an area that is still searching for a methodology.

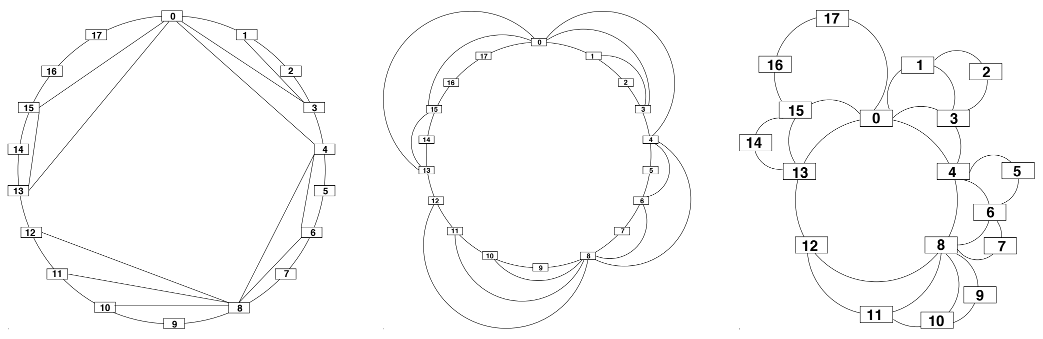

Figure 1: *A chordal (left), express (center), and multi-ring (right) augmentation of a ring.*

- Why should anyone believe my experimental "results?"
  - How should I select relevant sample inputs?
  - How should I analyze my raw data?

2 A Sample "What Matters?" Study [1]

2.1 What Matters in Networks: An Historical Perspective

Several notions of "equivalence" of specific networks have been studied.

Equivalence for all purposes. This strong form of equivalence demands that two network topologies be interchangeable for all computational/communicational applications. Such equivalence has been established most often using the notion of *quasi-isometry*—the inter-embeddability of two families of networks with small dilation. One finds in sources such as [2, 8, 14, 17] proofs of the strong equivalence of network topologies such as: paths and rings; meshes and tori; de Bruijn and shuffle-exchange networks; butterfly, FFT, and Cube-connected-cycle networks.

Equivalence on important classes of problems. This form of equivalence has received less attention than the stronger form, hence has engendered fewer generally applicable tools. Among the most interesting studies of computation-specific equivalence appear in: [21], where de Bruijn and butterfly and hypercube networks are shown to be equally efficient on "normal" algorithms; and [12], where a computationally more robust replacement for quasi-isometry is proposed and studied.

2.2 What Matters in Networks: A Recent Result

Consider the following three significant *augmentations* of ring networks, which are illustrated in Fig. 1.

A chordal ring is obtained by adding "shortcut" edges to a ring network, in order to decrease its diameter. In [1] we study chordal rings whose "shortcut" edges do not cross, a restriction that we show increases diameter by at most a factor of 2. An express ring is obtained from a chordal ring by routing its "shortcut" edges around periphery of ring in a way that preserves their noncrossing. An important parameter of an express ring is its *cutwidth*—the maximum number of edges that travel "above" adjacent nodes. A multi-ring is obtained from a ring by appending subsidiary rings to edges of the ring and, recursively, to edges of subsidiary rings. An important parameter of a multi-ring is its depth—the number of levels of augmentation.

The significance of these augmentations is attested to by the volume and variety of the literature they have engendered. Chordal rings have been studied as potential *interconnection networks* for parallel computers [3, 15]. Networks with express links (express rings being the simplest instance) have been studied in connection with *interconnection networks* [7]. Express rings, in particular, have been studied in connection with *communication networks* [10, 13]; their relation with chordal rings has been established in [9, 18]. Multi-rings have been studied both as *interconnection networks* [5, 16, 23] and as *communication networks* [6].

It is shown in [1] that these augmented rings are equivalent in a very strong sense. When edge-crossings are allowed, the augmented topologies are quasi-isometric. When crossings are not allowed, they are actually (graph-theoretically) isomorphic.

Theorem 1 **(a)** *For every chordal ring, there is an isomorphic express ring, and vice versa.* **(b)** *For every <u>cutwidth-c</u> express ring, there is an isomorphic <u>c-level</u> multi-ring, and vice versa.*

A Surprising Fact: In linear time, one can transform a chordal ring (even with crossing chords) into a *minimum cutwidth* isomorphic express ring (possibly with crossing arcs) [9].

Of course, the significance of Theorem 1 is that researchers and/or technologists who want to work efficiently with any of these network topologies now have access to a substantial body of literature that was not obviously relevant to their concerns.

3 A Sample "What if ...?" Study [19, 20]

The underlying goal of "What if ...?" questions is to investigate the computational and/or communicational consequences of currently speculative technology. Of course, one wants to keep one's models and analyses *"honest"* and *relevant* while anticipating technological advances. The danger one always faces is of forgetting the limitations of (currently) *conceivable* technology, or even of physics; cf., [4, 22]. Common offenses in this regard include:

- analyzing the asymptotics of $f(n)$ even when n must be small
- ignoring the time for traversing long wires or, even worse, for passing through many switches.

We now describe a "What if ...?" study that tries to be both "honest" and relevant. The study concerns the $\mathcal{R}$*econfigurable* $\mathcal{R}$*ing of* $\mathcal{P}$*rocessors* ($\mathcal{RRP}$), a parameterized family of parallel architectures specified as follows.

The (N,ℓ)-$\mathcal{RRP}$ comprises a ring of N identical single-port processors (PEs) hanging off a reconfigurable bus having ℓ lines, each one "word" wide. Every PE has ℓ *communication PEs* (CPEs) which control the bus-lines for that PE. Under SIMD control, the CPEs (independently) can:

- establish point-to-point bidirectional paths, which persist throughout a message transmission;
- relay words *actively* along their line;
- insert a word from their PE onto their line and/or remove a word for their PE from their line

What makes our study of $\mathcal{RRP}$'s notable is the posited *delay model.* We assume that the time to transmit a *one-word message* in an N-$\mathcal{RRP}$, along the path

$$\mathcal{P}_i \longrightarrow \mathcal{P}_{i+1} \longrightarrow \cdots \longrightarrow \mathcal{P}_{j-1} \longrightarrow \mathcal{P}_j$$

is $\lceil \log(j-i \bmod N) \rceil$ time units, where a time unit is the number of machine cycles it takes:

a PE to ...	a CPE to ...
receive a message via *one* CPE	put a word into its PE *and/or* get a word from its PE
and	
do a computation	*or*
and	
send a message via *one* CPE	actively relay a word

Now, a logarithmic-latency delay model is not unique to $\mathcal{RRP}$'s, even within the realm of paper architectures with reconfigurable buses. What is unique to our study—to the best of our knowledge—is our attempt to argue the technological *plausibility* of our model (in contrast to its technological *feasibility*, which we have not yet established, but which we hope to). This plausibility derives from the following strategy for achieving logarithmic message latency.

The Cooperative Message Transmission Protocol operates as follows. Say that PE $\mathcal{P}_i$ has a one-word message which it wants to transmit in the clockwise direction to PE $\mathcal{P}_j$ along line a.

At Step 0, CPE $\mathcal{P}_i^{(a)}$ sends the message to $\mathcal{P}_{i+1}^{(a)}$ along line a; schematically:

$$\bullet \Longrightarrow \circ \longrightarrow \circ \longrightarrow \circ \longrightarrow \circ \longrightarrow \circ \longrightarrow \circ \longrightarrow \circ \longrightarrow \circ \longrightarrow \cdots$$

At Step 1, $\mathcal{P}_{i+1}^{(a)}$ starts helping $\mathcal{P}_i^{(a)}$ by relaying the message actively; schematically:

$$\bullet \Longrightarrow \bullet \Longrightarrow \circ \longrightarrow \circ \longrightarrow \circ \longrightarrow \circ \longrightarrow \circ \longrightarrow \circ \longrightarrow \circ \longrightarrow \cdots$$

At Step 2, $\mathcal{P}_{i+2}^{(a)}$ and $\mathcal{P}_{i+3}^{(a)}$ start helping $\mathcal{P}_i^{(a)}$ and $\mathcal{P}_{i+1}^{(a)}$ by relaying the message actively; schematically:

$$\bullet \Longrightarrow \bullet \Longrightarrow \bullet \Longrightarrow \bullet \Longrightarrow \circ \longrightarrow \circ \longrightarrow \circ \longrightarrow \circ \longrightarrow \circ \longrightarrow \cdots$$

$$\vdots \quad \vdots \quad \vdots$$

At Step k, $\mathcal{P}^{(a)}_{i+2^{k-1}}, \mathcal{P}^{(a)}_{i+2^{k-1}+1}, \ldots, \mathcal{P}^{(a)}_{i+2^{k}-1}$ start helping $\mathcal{P}^{(a)}_{i}, \mathcal{P}^{(a)}_{i+1}, \ldots, \mathcal{P}^{(a)}_{i+2^{k-1}-1}$ by relaying the message actively; schematically:

$$\bullet \Longrightarrow \bullet \Longrightarrow \bullet \Longrightarrow \bullet \Longrightarrow \bullet \Longrightarrow \bullet \Longrightarrow \bullet \Longrightarrow \bullet \Longrightarrow \circ \longrightarrow \cdots$$

This continues until the message reaches $\mathcal{P}^{(a)}_{j}$, which puts the message into PE $\mathcal{P}_j$.

Notes. This scheme (if it works) decreases just *capacitive delays*, so we never approach speed-of-light limitations. Moreover, the scheme works just for one-word messages, not for worms. Most importantly for our message here: If $\mathcal{RRP}$'s can be made to work with logarithmic message latency, then it will be within something like an MOS wafer-scale technology. This implies that

> *N will always be modest in size. Therefore, one is not justified in using asymptotic analyses to determine the capabilities of logarithmic-latency $\mathcal{RRP}$'s!*

Even when timing is expressed in terms of explicit, rather than asymptotic bounds, $\mathcal{RRP}$'s are unexpectedly efficient, as the following theorem indicates.

Theorem 2 *An (N, ℓ)-$\mathcal{RRP}$ can:*

(a) *execute any N-leaf* *<u>tree-sweep computation</u>* *within time* $\log N \log\log \ell + \dfrac{\log^2 N}{\log \ell}$.

When $\ell \geq N^{1/\log\log N}$, *this is time* $2 \log N \log\log N + l.o.t.$

Such computations can perform: broadcast; accumulation; parallel-prefix.

(b) *execute any N-node* *<u>hypercube normal algorithm</u>,* *with slowdown* $\leq 4 \log\log N$.

Such algorithms can perform: matrix operations; fast fourier transform; sorting.

The times reported in Theorem 2 are within constant factors of optimal.

4 A Sample "Educational" Empirical Study [11]

As we noted in Section 1, the basic questions concerning empirical studies revolve around the *credibility* and *extendibility* of their results. Challenges exist at both the coarse and fine levels of planning an experimental study. At the coarse level, one must try to devise experiments that have some hope of explaining *why* phenomena occur, not just *that* they do. At a more detailed level, one must plan how to select sample inputs that are relevant to the real-life situation one is attempting to study empirically; and, one must devise a method of analyzing one's experimental results in a manner that legitimately supports the claims that one makes.

We look now at a sample empirical study that attempts to deal with these issues. The study attempts to analyze and compare two competing policies for scheduling branching computations on ring-structured parallel computers. The study has two quite distinct goals, one relating to parallel computation and one to experimental methodology. Within the former domain, we wish to devise efficient scheduling strategies for large classes of important computations on processor-rings, despite their large diameters and small (bisection) bandwidths. Within the latter domain, we wish to craft a methodology for evaluating parallel scheduling policies, that will compare one policy's schedules against another's (a standard goal) and explain each policy's sources of efficiency and inefficiency (a decidedly nonstandard goal).

In broad overview, the computations we wish to schedule have the structure of dynamically evolving binary trees with unit-sized tasks at each node. We start with just one node, which is simultaneously the root of the tree and its only leaf. Inductively, when a leaf-node is executed, it either "dies"—thereby becoming a permanent leaf of the tree—or spawns two children—thereby becoming an internal node of the tree. The scheduling policies that form the focus of our study are specified as follows.

- Policy KOSO (KEEP-ONE-SEND-ONE) has each PE retain one child of a spawning task and send the other child to its clockwise neighbor.
- Policy KOSO* (refined KOSO) has each PE behave like KOSO, *but* send child to its clockwise neighbor *only if the neighbor has a lighter load.*

Both policies schedule in a *local breadth-first* fashion; i.e., each PE executes the lexicographically smallest node in its *eligible-queue.*

Our hypothesis is that both KOSO and KOSO* produce "good" schedules (i.e., ones that achieve close to optimal p-fold speedup on the p-PE ring $\mathcal{R}_p$) when scheduling trees that turn out to be "bushy;" but, we expect KOSO* to produce better speedup, because of its more intelligent load balancing.

4.1 The Design of the Experiments

We need to design experiments that are computationally both feasible and meaningful. Toward the former end, we must devise an efficient method of generating inputs for our study; toward the latter end, we must ensure that the inputs we generate faithfully model some real-life application. As our "reality check," we selected the real-life application of numerical integration using, say, Simpson's Rule or the Trapezoid Rule. These algorithms can be viewed as the simplest instances of the important family of multi-grid computations. The correspondence that exposes these algorithms as instances of the kind of branching computations we wish to schedule appears in the following table.

Formal Entity	**"Real" Entity**
tree-node	real interval
spawning node	interval being refined
dying node	completed interval (because of smoothness or resolution)

With, say, Trapezoid-Rule integration algorithms as inspiration, we decided to generate random instances of branching computations via the following one-parameter procedure. We choose a real number δ that is close to, but less than 1. Then we generate δ-random trees by having each level-ℓ node of a tree (where $\ell \geq 0$) spawn with probability δ^ℓ and "die" with probability $1 - \delta^\ell$.

We argue heuristically that the trees generated by the preceding procedure should have structure similar to those generated by "typical" runs of a Trapezoid-Rule integration algorithm, as follows. On the one hand, Trapezoid-Rule trees should be "bushy" near their roots, because any nonlinear function will require some interval splitting. On the other hand, Trapezoid-Rule trees should get "scrawny" rather quickly, because functions occurring in actual applications are likely to have large smooth parts. Finally, the limited resolution of any computer should keep Trapezoid-Rule trees rather shallow.

4.2 The Results of the Experiments

Comparing KOSO **and** KOSO*. We performed 480 trials, 20 for each setting of: the ring size (4 values of p), the growth parameter (3 values of δ), and the scheduling policy (KOSO vs. KOSO*). The direct results of the trials are summarized in the first line of Table 1.

	KOSO	KOSO*	Prob. that KOSO*'s mean = KOSO's mean
Mean speed (large std. dev.)	2485 steps	2237 steps	≈ 0.17
Mean deviation from ideal speedup	306 steps	92 steps	< 0.001

Table 1. *The results of our experiments*

While it appears clear from looking at the raw and mean run times of the computations scheduled by KOSO and KOSO* that the latter policy produces better schedules than the former, we discovered to our chagrin that the anticipated "clear" superiority of KOSO* is not borne out in a statistically significant manner by our results! Once we developed the intuition that this lack of significance resulted from the large standard deviation in KOSO's performance, we decided to look at each run's deviation from ideal (p-fold) speedup, rather than its raw speed. We thereby

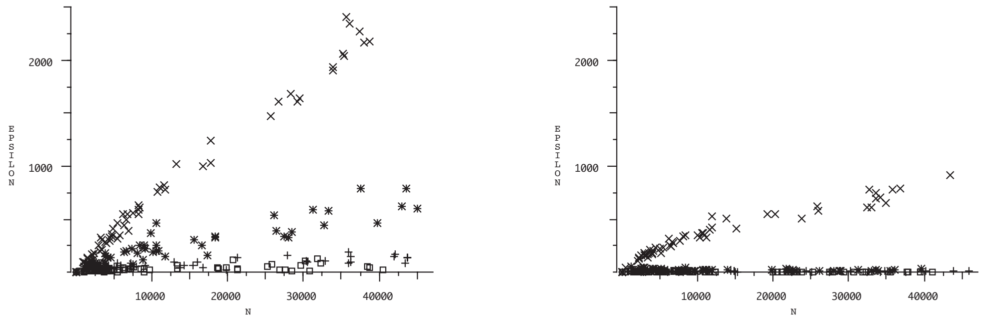

Figure 2: *The plotted deviation from ideal speedup for* KOSO *(on the left) and* KOSO* *(on the right). In the plots:* $\times$ *= 20 PEs,* $*$ *= 10 PEs,* $+$ *= 6 PES,* □ *= 3 PEs.*

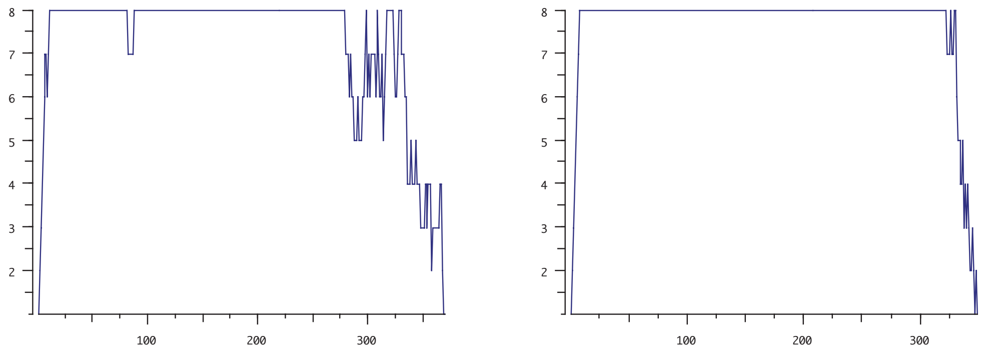

Figure 3: *The time-series profile of the busy PEs on* $\mathcal{R}_8$ *for* KOSO *(on the left) and* KOSO* *(on the right).*

generated the second line of Table 1, which does indicate in a statistically significant manner that policy KOSO* outperforms policy KOSO.

Not surprisingly, the superiority of KOSO* over KOSO is clearly visible in Fig. 2, which plots the observed deviation from ideal speedup of both policies. What we must stress here is that one cannot reliably draw conclusions from eyeballing plots: we were initially "fooled" by the plot of raw speeds into thinking that we had a statistically significant demonstration.

Why KOSO* Outperforms KOSO. Figs. 3 and 4 illustrate eloquently that KOSO* earns its superiority over KOSO in two fundamental ways. Fig. 3's time-series profile of the busy PEs of $\mathcal{R}_8$ during our experimental runs illustrates that KOSO* keeps almost all PEs of $\mathcal{R}_8$ busy almost all of the time. In particular, KOSO* enjoyed "steeper" warm-up and cool-down phases (wherein, respectively, the PEs of $\mathcal{R}_8$ initially load up with work and finally empty out) than KOSO. Additionally, Fig. 4's time-series profile of the queue-sizes of the PEs of $\mathcal{R}_8$ during our experimental runs illustrates that KOSO* balances the workloads of the PEs of $\mathcal{R}_8$ almost perfectly, certainly more evenly than KOSO does.

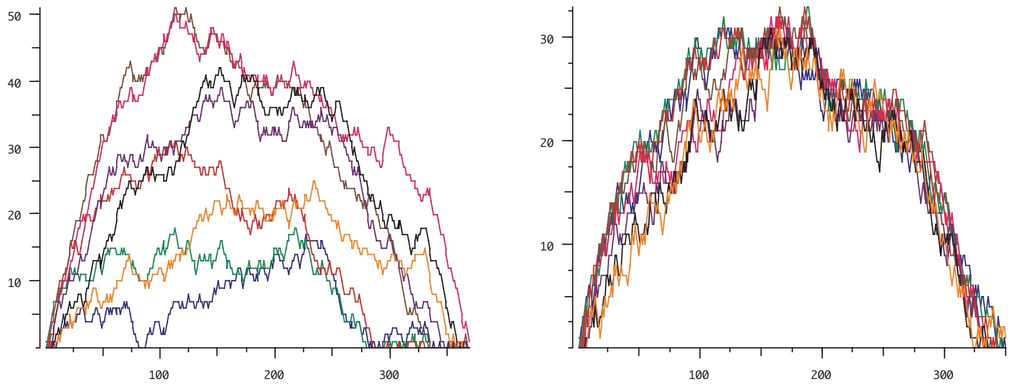

Figure 4: *The time-series profile of queue-sizes on $\mathcal{R}_8$ for* KOSO *(on the left) and* KOSO* *(on the right).*

References

[1] W. Aiello, S.N. Bhatt, F.R.K. Chung, A.L. Rosenberg, R.K. Sitaraman (1997): Augmented ring networks. Tech. Rpt. 97-36, Univ. Massachusetts. See also an extended abstract in *J. Math. Modelling and Scientific Computing 8*.

[2] F.S. Annexstein, M. Baumslag, A.L. Rosenberg (1990): Group action graphs and parallel architectures. *SIAM J. Comput. 19*, 544-569.

[3] B.W. Arden and H. Lee (1981): Analysis of chordal ring networks. *IEEE Trans. Comp., C-30*, 291-295.

[4] G. Bilardi and F.P. Preparata (1995): Horizons of parallel computation. *J. Parallel Distr. Comput. 27*, 172-182.

[5] H. Burkhardt et al. (1992): Overview of the KSR1 computer system. Tech. Rpt. KSR-TR 9202001, Kendall Square Research.

[6] S. Cosares, I. Saniee, O. Wasem (1992): Network planning with the SONET toolkit. *Bellcore Exchange* (Sept./Oct., 1992) 8-15.

[7] W.J. Dally (1991): Express cubes: improving the performance of k-ary n-cube interconnection networks. *IEEE Trans. Comp. 40*, 1016-1023.

[8] R. Feldmann and W. Unger (1992): The cube-connected cycles network is a subgraph of the butterfly network. *Parallel Proc. Let. 2*, 13-19.

[9] A. Frank, T. Nishizeki, N. Saito, H. Suzuki, É. Tardos (1992): Algorithms for routing around a rectangle. *Discr. Appl. Math. 40*, 363-378.

[10] O. Gerstel and S. Zaks (1995): The virtual path layout problem in ATM ring and mesh networks. *1st Conf. on Structure, Information and Communication Complexity*, Carleton Univ. Press, Ottawa.

[11] D.E. Gregory, L.-X. Gao, A.L. Rosenberg, P.R. Cohen (1996): An empirical study of dynamic scheduling on rings of processors. *8th IEEE Symp. on Parallel and Distr. Processing*, 470-473.

[12] R. Koch, F.T. Leighton, B.M. Maggs, S.B. Rao, A.L. Rosenberg, E.J. Schwabe (1997): Work-preserving emulations of fixed-connection networks. *J. ACM 44*, 104-147.

[13] E. Kranakis, D. Krizanc, A. Pelc (1997): Hop-congestion tradeoffs for high-speed networks. *Intl. J. Foundations of Computer Science 8*, 117-126.

[14] C.P. Kruskal and M. Snir (1986): A unified theory of interconnection network structure. *Theor. Comp. Sci. 48*, 75-94.

[15] D.-M. Kwai and B. Parhami (1996): Periodically regular chordal rings: generality, scalability, and VLSI layout. *8th IEEE Symp. on Parallel and Distr. Processing*, 148-151.

[16] C. Lam, H. Jiang, V.C. Hamacher (1995): Design and analysis of hierarchical ring networks for shared-memory multiprocessors. *Intl. Conf. on Parallel Processing*, I:46-50.

[17] F.T. Leighton (1992): *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann, San Mateo, Calif.

[18] A. Litman and A.L. Rosenberg (1993): Balancing communication in ring-structured networks. Tech. Rpt. 93-80, Univ. Massachusetts.

[19] A.L. Rosenberg, V. Scarano, R.K. Sitaraman (1995): The Reconfigurable Ring of Processors: efficient algorithms via hypercube simulation. *Parallel Proc. Let. 5* (Special Issue on Dynamically Reconfigurable Architectures) 37-48.

[20] A.L. Rosenberg, V. Scarano, R.K. Sitaraman (1997): The Reconfigurable Ring of Processors: efficient leveled tree-structured algorithms. *IEEE Trans. Comp. 46*, to appear. See also, The Reconfigurable Ring of Processors: fine-grain tree-structured computations. *6th IEEE Symp. on Parallel and Distr. Processing*, 470-477.

[21] J.D. Ullman (1984): *Computational Aspects of VLSI.* Computer Science Press, Rockville, Md.

[22] P.M.B. Vitanyi (1988): Locality, communication and interconnect length in multicomputers. *SIAM J. Comput. 17*, 659-672.

[23] Z.G. Vranesic, M. Stumm, D.M. Lewis, R. White (1991): Hector: a hierarchically structured shared-memory multiprocessor. *IEEE Computer 24* (1) 72-79.