GENERALIZED CONNECTION NETWORKS FOR PARALLEL PROCESSOR INTERCOMMUNICATION

C. D. Thompson

May, 1977

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pa. 15213

This research was supported in part by the National Science Foundation under Grant MCS 75-222-55 and the Office of Naval Research under Contract N00014-76-C-0370, NR 044-422. The author is an NSF fellow.

Abstract

A generalized connection network (GCN) is a switching network with N inputs and N outputs that can be set to pass any of the N^N mappings of inputs onto outputs. This paper demonstrates an intimate connection between the problems of GCN construction, message routing on SIMD computers, and "resource partitioning." A GCN due to Ofman [1965] is here improved to use less than 8N log N contact pairs, making it the minimal known construction.

Any GCN construction leads to a new algorithm for the broadcast of messages among processing elements of an SIMD computer, when each processing element is to receive one message. Previous approaches to message broadcasting have not handled the problem in its full generality. The algorithm arising from this paper's GCN takes 8 log N (or 13 $N^{1/2}$) routing steps on an N element processor of the perfect shuffle (or mesh-type) variety.

If each resource in a multiprocessing environment is assigned one output of a GCN, private buses may be provided for any number of disjoint subsets of the resources. The partitioning construction derived from this paper's GCN has 6N log N switches, providing an alternative to "banyan networks" with O(N log N) switches but incomplete functionality.

1. Introduction

A generalized connection network (GCN) is a switching network with N inputs and N outputs capable of implementing any mapping of inputs onto outputs. In other words, each output may be connected to any one of the inputs, for a total of N^N different connection patterns. Thus a GCN is more powerful than the connection networks of Benes [1965] et al, for a connection network handles only one-to-one mappings of inputs onto outputs (N! settings). An important paramater of any GCN is its delay, that is, the maximum number of switches that separate any input-output pair. For example, an NxN crosspoint switch is a GCN with N^2 contact pairs and unit delay. Ofman's [1965] construction has 5 log N delay and 10N log N contact pairs (all logarithms in this paper are base 2). Ofman's construction is here improved to 4 log N delay and 8N log N contact pairs. Other GCN results are a construction with 8N log N contact pairs but O(N log N) delay (Pippenger [1973]), a construction with $O(N^{5/3})$ contact pairs (Masson and Jordan [1972]), and a non-constructive proof (Pippenger [1977]) that GCNs need only O(N) more switches than connection networks. (The best connection network construction has (6/log 3)N log N = 3.8 N log N contact pairs--Benes [1965].)

Any GCN construction leads to an algorithm for the transfer of data among processing elements of an SIMD (Single Instruction stream Multiple Data stream: Flynn [1966]) computer. This data transfer is modeled as the routing of messages, each originating at a processing element and destined for some subset of the other processing elements. There have been many papers treating particular message routing patterns on particular networks (Stone [1971], Siegel [1976], Orcutt [1976], . . .). The algorithm based on the GCN of this paper performs near-optimally on any message distribution pattern in which each processing element receives at most one message, on several popular SIMD interconnection networks. For an N element computer, the algorithm requires $13N^{1/2}$ routing steps on a square mesh-type array, 8 log N routing steps on the perfect shuffle, PM2I, and WPM2I networks, and 4 log N routing steps on the Cube (see Section 3 for descriptions of these networks). All other known GCN constructions lead to slower routing algorithms.

Finally, any GCN construction applies to the partitioning of multiprocessor systems in the sense of Goke and Lipovski [1973]. If each resource is assigned one output of a GCN, proper switch settings provide a private conductive path for each of any number of disjoint subsystems. The banyan networks originally proposed for this task do not implement all partitions when O(N log N) switches are employed. When used for partitioning, 1/4 of this paper's GCN can be omitted, so that unrestricted partitioning may be obtained with less than 6N log N switches. No other known GCN construction leads to smaller partitioners.

The new GCN construction is described in Section 2, its application to message routing is elaborated in Section 3, and its related partitioning network is derived in Section 4.

2. A GCN Construction

A GCN may be represented as a graph with two ordered sets of N vertices each ("inputs" and "outputs"), such that for any sequence $j_1,j_2,\ldots,j_N$ $(1 \le j_k \le N)$ there exists a subgraph in which input vertex i is connected with output vertex k iff $j_k = i$. In this model, vertices are wires and edges are switches; edges included in the subgraph for a particular $j_1,\ldots,j_N$ are precisely those switches that must be closed to connect the ith input to those outputs k for which $j_k = i$. A trivial GCN construction is the complete bipartite graph on 2N nodes, which corresponds to the NxN crosspoint switch. Since there are N^N different ways of choosing the j_k sequence, at least $\lg(N^N) = N \log N$ contact pairs (edges) are required in any GCN.

A GCN construction may be obtained from the following schema (Ofman [1965]).

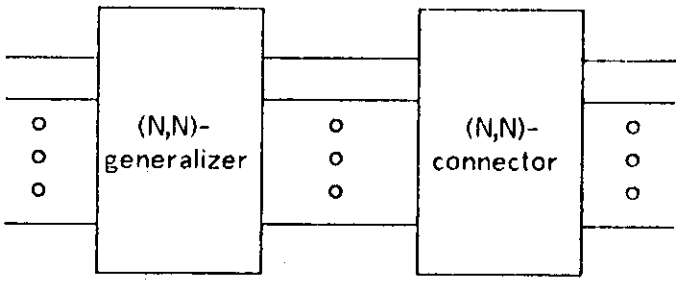
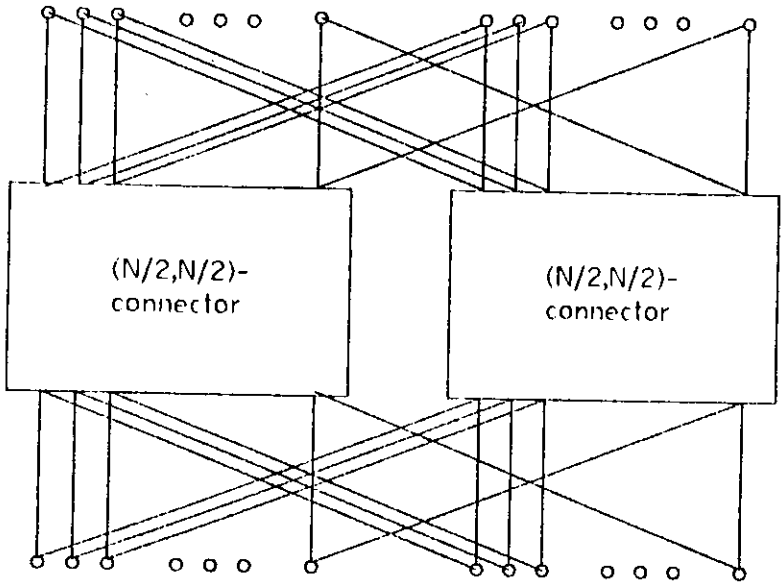


Figure 1. Schema for a GCN construction.

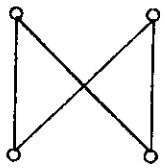
The left-hand network produces the correct number of copies of each of the inputs, which are then permuted to the proper outputs by the right-hand network. It is now necessary to examine connection and generalization networks in more detail.

2a. Connection Networks

An (N,N)-connection network is a switching network with N inputs and N outputs capable of passing any of the N! one-to-one mappings (permutations of inputs onto outputs). This is of course strictly less powerful than a GCN, in which the same input may be connected to more than one output at a time. Benes [1965] published the following 4N log N construction in which an N-input connection network is synthesized from 2 N/2-input connectors and 4N additional contact pairs.



(N,N)-connector



(2,2)-connector

Figure 2. Benes' connection network.

The proper switch settings for any desired connection pattern may be found by the method of Waksman [1968] in O(N log N) time on a serial computer, the best result known. The author has developed a divide-and-conquer approach that would run on many N element SIMD computers in O(N) time, but does not know how to reduce this to an acceptable (sublinear) figure. Thus it would seem that lengthy preprocessing time will be required for each GCN setting. In some cases, it may be feasible to tabulate precomputed GCN settings, although it would seem necessary to store O(N log N) bits for each setting.

It should be noted that this connector construction is symmetric about a horizontal axis. In fact, the top log N layers and the bottom log N layers comprise Omega networks (see Lawrie [1973]) that share a common level of switches.

2b. Generalization Networks

An (N,N)-generalizer passes input i to m_i different outputs, where $\sum m_i = N$ and $m_i \geq 0$. Thus it provides a particular number of copies of each input somewhere among the outputs. The existence of (N,N)-generalizers with O(N) switches has been demonstrated non-constructively by Pippenger [1977]. Construction of a generalizer can be accomplished by the following schema, due to Ofman [1965].

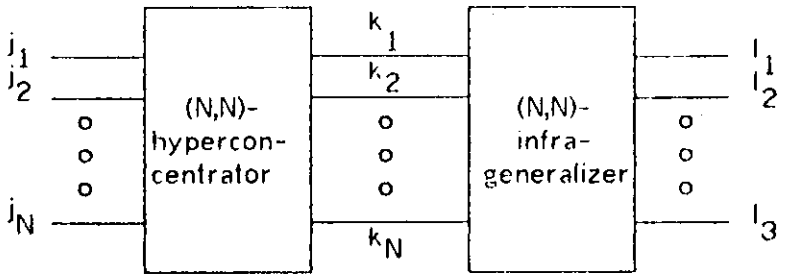


Figure 3. Schema for a generalizer construction.

The left-hand network routes all important inputs to its uppermost output lines. More precisely, if m of the inputs will not appear on any output of the generalizer, the other N-m inputs must appear on lines k_1 through k_{N-m} of the hyperconcentrator. The right-hand network is responsible for producing the correct number of copies of each of its inputs, but there must exist some integer p such that $k_1, k_2, \ldots, k_p$ will appear in the output at least once, while $k_{p+1}, k_{p+2}, \ldots, k_N$ will be ignored. Ofman demonstrates that the following network is an infrageneralizer.

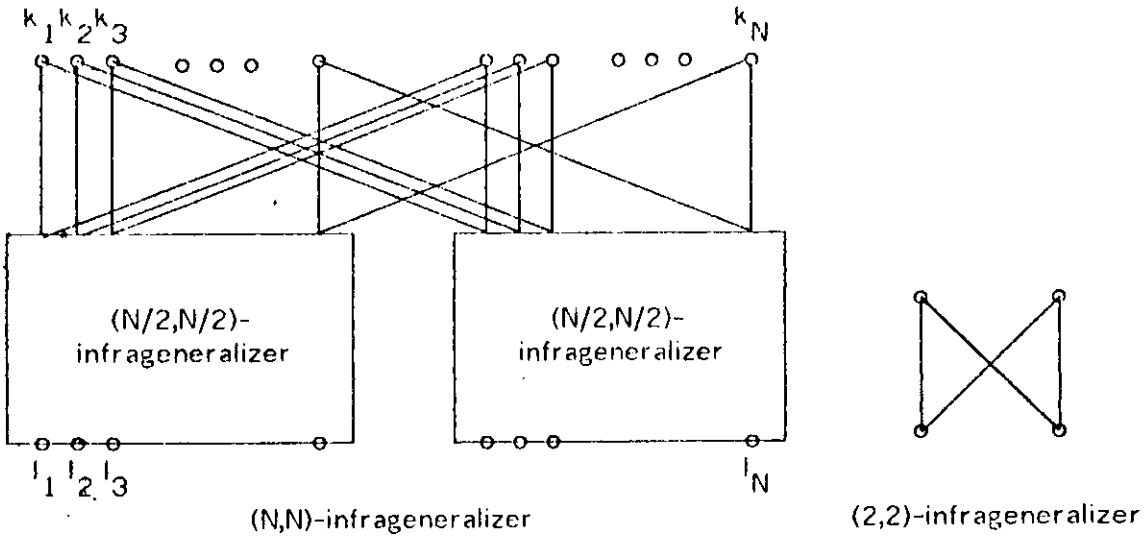


Figure 4. Ofman's infrageneralizer.

A more complete specification of Ofman's network is required. The leftmost m_q output lines will bear copies of input k_q, the next m_{q+1} outputs bear input k_{q+1}, . . ., and the rightmost m_p outputs bear input k_p, where $m_i>0$ for $1\le q\le i\le p\le N$, $m_i=0$ for $i<q$ or $i>p$, and $\sum m_i=N$. If q is chosen to be 1, this specification can be seen to satisfy the requirements for an infrageneralizer. The proper switch settings for Ofman's network may be obtained recursively by using the upper switches to give each half-sized infrageneralizer half the input signals. It should be noted that the signals required by

each half-sized infrageneralizer form a consecutive subsequence of the original inputs (if the leftmost input is considered to "follow" the rightmost one).

An (N,N)-connection network could be used for hyperconcentration, since a hyperconcentrator merely permutes its inputs. This is in fact Ofman's approach, yielding an (N,N)-generalizer with 6N log N contact pairs. Ofman's construction can be improved by using fewer switches in the hyperconcentrator portion. Somewhat surprisingly, Ofman's infrageneralizer is an "upside down" hyperconcentrator--the direction of signal flow through the network is reversed by turning inputs into outputs and vice versa. This equivalence will be verified by the demonstration of a correspondence between any desired hyperconcentration function and an infrageneralizer function. A hyperconcentration setting may be specified by a list of p integers, $n_1,n_2,\ldots,n_p$ with $1\le n_1<n_2<\ldots<n_p\le N$, corresponding to the indices of the inputs whose signals are to appear in the first p output lines. The corresponding infrageneralizer function is that input i should appear on m_i output lines, where $m_i = n_i - n_{i-1}$, $n_0 = 0$, and $n_{p+1} = n_{p+2} = \ldots = n_N = N$. Ofman's infrageneralizer will connect input i to outputs $n_{i-1}+1$ through n_i; if switches are opened to disconnect all but output number n_i for $1\le i\le p$, then the required hyperconcentration function is implemented by the reversed infrageneralizer.

An example should clarify matters. A (8,8)-hyperconcentrator setting for n_i = (2,3,6,7,8) corresponds to a (8,8)-infrageneralizer setting for m_i = (2,1,3,1,1,0,0,0). In other words, the problem of finding the proper switch settings to bring inputs 2, 3, 6, 7, and 8 to outputs 1, 2, 3, 4, and 5 (a hyperconcentration) may be solved by setting Ofman's infrageneralizer to route input 1 to outputs 1 and 2; input 2 to output 3; input 3 to outputs 4, 5, and 6; input 4 to output 7; and input 5 to output 8.

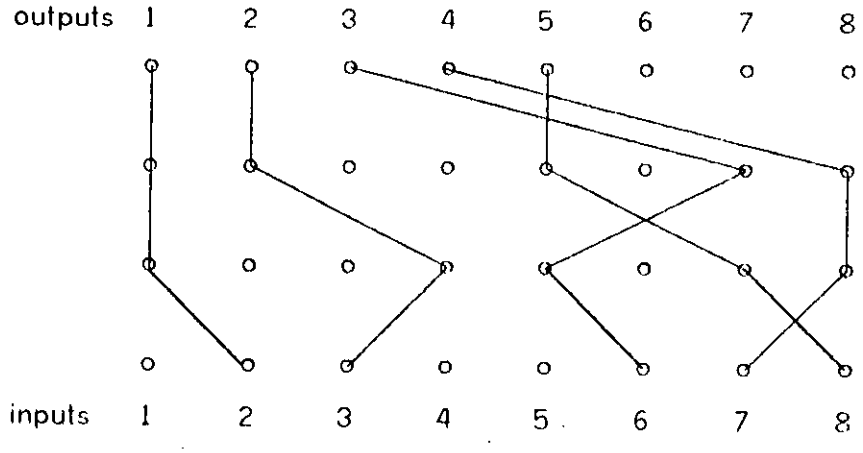


Figure 5. An upside-down hyperconcentrator set for (2,3,6,7,8).

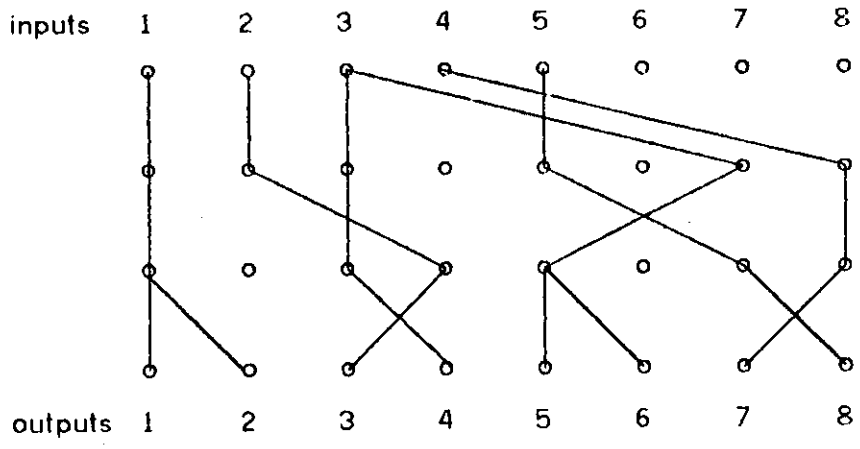


Figure 6. An infrageneralizer set for (2,1,3,1,1,0,0,0).

Since Ofman's (N,N)-infrageneralizer has 2N log N contact pairs, an (N,N)-generalizer can be built with 4N log N contact pairs by attaching a infrageneralizer to a reversed infrageneralizer (a hyperconcentrator). Since the last stage of the hyperconcentrator is identical to the first stage of the infrageneralizer, the combined functionality of these two levels of switches may be obtained with a single one, eliminating 2N contact pairs. The (8,8)-generalizer obtained in this way is illustrated below.

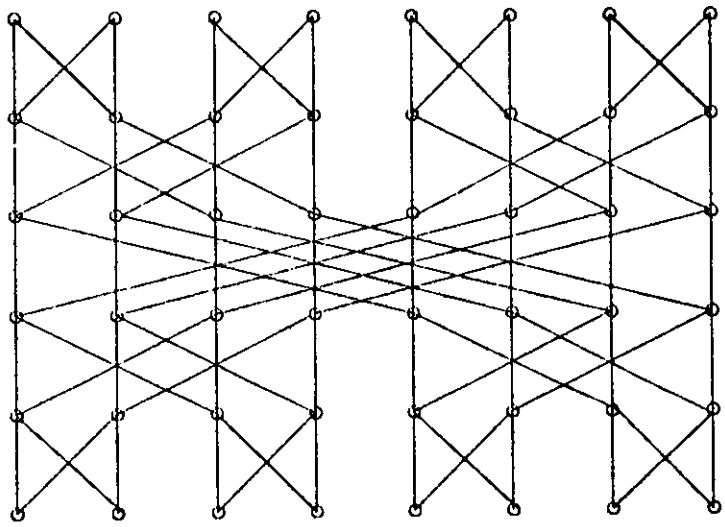


Figure 7. An (8,8)-generalizer.

2c. The complete GCN construction

The astute reader will have noticed that the (N,N)-generalizer of Subsection 2b is quite similar to the (N,N)-connection network of Subsection 2a. In fact, one merely needs to "unshuffle" the inputs and outputs of this (N,N)-connection network to make the two networks identical. Then, when concatenating the generalization and connection networks to obtain a GCN, the first stage of the latter can be combined with the last stage of the former to yield (for N=8)

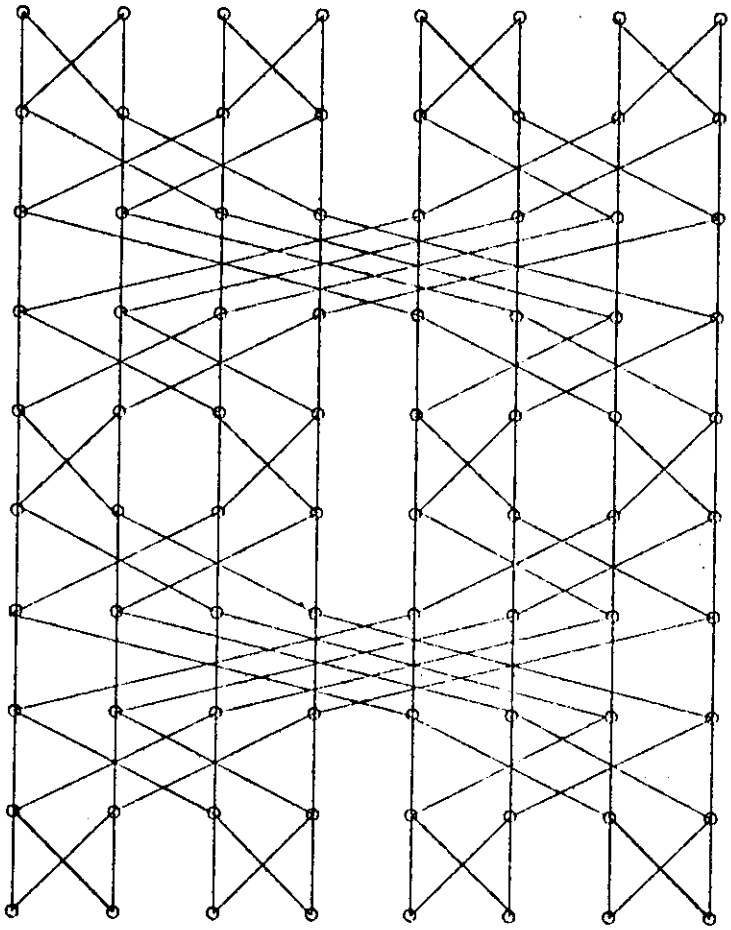


Figure 8. An (8,8)-GCN.

As it stands, there are 8N log N - 6N contact pairs in this GCN (N a power of 2). However, O(N) contacts may be stripped from the connector (see Waksman [1968]) and the generalizer leaving 8N log N - 10.5N + 11 contact pairs. These GCN constructions have 4 log N - 3 delay. An alternative construction based on three-way branching yields (N,N)-GCNs with $(12/\log 3)N \log_3 N - (71/6)N + 13.5$ contact pairs (N a power of 3), the author's best result (note: $12/\log 3 = 7.6$). This three-way branching construction has delay $(4/\log 3)\log N - 3$, which makes it the fastest known GCN.

3. Message Broadcasting

An SIMD computer may be considered to consist of three major parts: a central control unit, the processing elements, and an interconnection network. Each PE (processing element) operates on data in its own local memory according to the dictates of the central control unit. Data enters and leaves this local memory via the interconnection network, which typically connects each PE to one of several neighboring PEs. For example, in a mesh-type computer each PE has at most four neighbors. The situation may be depicted as follows, where the boxes are PEs and the lines are possible connections.

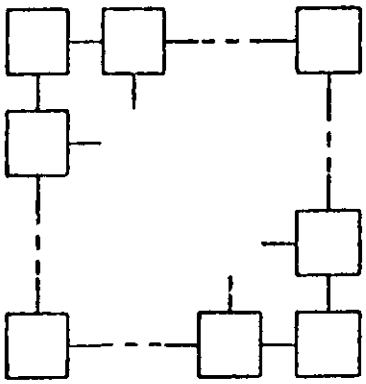


Figure 9. A mesh-connected computer.

Note that PEs on the edges have fewer than four neighbors. Given the strongly local nature of the connection pattern, efficient intercommunication algorithms would seem necessary for effective use of such a computer.

The message broadcasting problem may now be broadly stated. Initially, each PE has generated a message of interest to some (possibly empty) subset of the other PEs. Each PE is to receive exactly one interesting message. How long does it take to deliver all the messages, as a function of the total number of PEs and their interconnection pattern? Time is measured in the number of (parallel) unit-distance message routings, i.e., if the mesh-type interconnection network is set to "up", in one time unit each of the PEs may receive a copy of the message sent by its downward-adjacent PE. For simplicity, assume that no time is spent on selecting which message (of possibly several) will be sent from each PE. This assumption is valid on a computer with a sufficiently powerful control unit (each PE is explicitly told which message to send), and is nearly valid when routing decisions are made locally (for example, by examination of "routing tags" on the messages). The algorithms of this paper will place at most two messages in a PE at a time, so these routing decisions should not be time-consuming. As mentioned in Subsection 2a, substantial preprocessing time will be required for each distribution pattern, but will not be included in message delivery time.

The next three Subsections will solve the message broadcasting problem for several different interconnection networks.

3a. Message broadcasting on the mesh-connected computer

Referring to the N element mesh-connected computer of Figure 9, it may be seen that $4(N^{1/2}-1)$ time units may be needed by some message broadcasting patterns. It would take a message $2(N^{1/2}-1)$ time to travel from the upper-leftmost PE to the lower-rightmost PE. During all that time the interconnection network has been set to "left" and "down", so another $2(N^{1/2}-1)$ time units would be required to send a message along the reverse route. It is not known whether more complicated broadcasting patterns require more routing time. However, a large number of patterns can be completed in $4(N^{1/2}-1)$ time -- the so-called Omega permutations (Orcutt[1976]). Also, any one-to-one pattern (each message goes to exactly one PE) can be accomplished in about $6N^{1/2}$ time, when N is very large (Thompson and Kung [1977]). When N is small, $7(N^{1/2}-1)$ time is sufficient, as indicated later in this Subsection. The main algorithm of this Subsection demonstrates that no broadcast pattern need take more than $13N^{1/2}$ time units.

A relationship between a GCN construction and a message routing algorithm may be drawn in the following way. Each node of a GCN corresponds to a PE, and each arc to a message routing. If each of the N input nodes of a GCN corresponds to a different PE, and if the same condition holds for the output nodes, then any GCN setting indicates how to perform the corresponding message broadcast. Careful choice of the node-PE numberings will result in a fast message broadcasting algorithm. For example, on a 4x4 processor, the following indexing scheme seems natural.

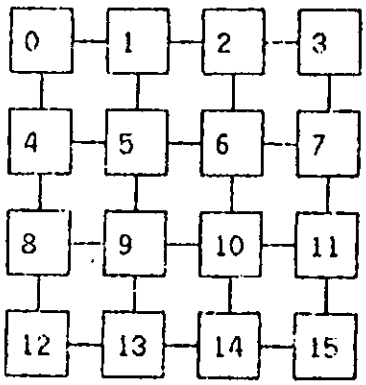


Figure 10. Natural indexing of a 4x4 mesh-connected processor.

If the 16 nodes on each level of the (16,16)-GCN built according to Section 2 are numbered from left(0) to right (15), then the corresponding routing algorithm may be drawn as follows.

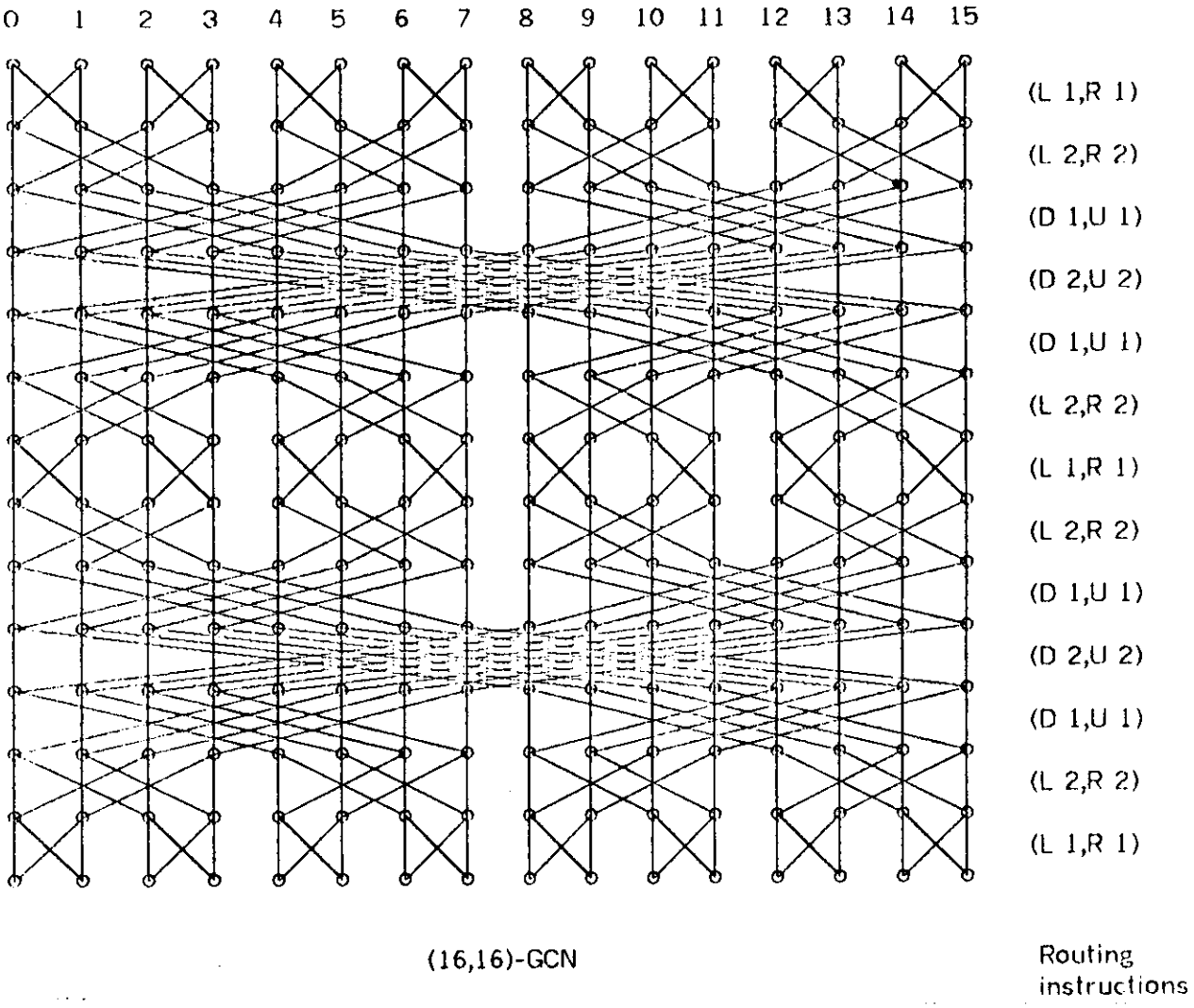


Figure 11. Routing on a 4x4 mesh-connected computer.

In general, this approach on an N element computer would require 3 (L 1,R 1) routings, 4 (L 2,R 2) routings, 4 (L 4,R 4) routings, . . ., 4 (L $N^{1/2}/2$,R $N^{1/2}/2$) routings, 4 (D 1,U 1) routings, 4 (D 2,U 2) routings, . . ., 4 (D $N^{1/2}/4$,U $N^{1/2}/4$) routings, and 2 (D $N^{1/2}/2$,U $N^{1/2}/2$) routings. Note that there are four routings of every type in the list except the first and the last. This list sums to $14N^{1/2}-18$ time units. However, the result can be improved to $13N^{1/2}-16$ by renumbering the nodes of the GCN. Since the first routing type in the list above only occurs 3 times, it should be a relatively long one, freeing a quick unit-distance routing for a step that is repeated 4 times. For example, the GCN nodes may be indexed from left to right as (0,2,1,3,4,6,5,7,8,10,9,11,12,14,13,15) on each level. This sequence was obtained from the binary representation of the natural sequence by exchanging the least significant bit with the (log N)/2 th least significant bit.

If a particular message distribution pattern happens to be one-to-one (each message goes to exactly one PE), then the full power of a GCN simulation is not required. Instead, a simulation of the connection network imbedded in the last half of the GCN can be accomplished in $7N^{1/2}-8$ time units, using the natural correspondence scheme.

The author cannot resist noting that Batcher's bitonic sorting network is a GCN when run "backwards" (proof supplied upon request). This leads to an alternative algorithm for message distribution that runs in time bounded by $14N^{1/2}$ (Thompson and Kung [1977]).

3b. Message broadcasting on a perfect shuffle computer

The perfect shuffle interconnection (Stone [1971]) is nicely suited for message broadcasting. A GCN may be simulated in 8 log N - 7 time units, giving a correspondingly low upper bound for the time required by any message distribution pattern.

Let the PEs of a perfect shuffle computer be numbered from 0 to N-1. Each index can be represented in log N = m binary bits, $b_m b_{m-1} \cdots b_3 b_2 b_1$. The perfect shuffle interconnection network has just three settings, so that PE $b_m \cdots b_1$ is connected to $b_m b_{m-1} \cdots b_2 \bar{b}_1$ ("exchange"), to $b_{m-1} b_{m-2} \cdots b_2 b_1 b_m$ ("shuffle"), and to $b_1 b_m b_{m-1} \cdots b_3 b_2$ ("unshuffle").

An optimal numbering of the nodes of Section 2's GCN construction is easily derived. Let the input nodes be labeled 0 (left) through N-1 (right). The labelings of the next log N - 1 rows of GCN nodes are obtained by unshuffling the binary representation of the labels of the previous row. For example, if N = 8, the first row is (0,1,2,3,4,5,6,7), the second row is (0,4,1,5,2,6,3,7), and the third row is (0,2,4,6,1,3,5,7). The (log N)th through the (2 log N - 1)th rows are labeled by shuffling the indices in the previous row. In the present example, the fourth row is (0,4,1,5,2,6,3,7) and the fifth row is (0,1,2,3,4,5,6,7). The (2 log N - 1)th through the (4 log N - 3)th rows are labeled identically to the 1st through the (2 log N - 1)th rows, while the output row (the (4 log N - 2)th) is numbered naturally.

This GCN numbering may be motivated by considering the corresponding perfect shuffle network settings. In the example above, the first two rows are (0,1,2,3,4,5,6,7) and (0,4,1,5,2,6,3,7). Thus, after the first stage of GCN simulation, each of PE 0 and PE 4 has one of the messages originally in PE 0 and PE 1; PE 1 and PE 5 have messages from either PE 2 or PE 3; PEs 2 and 6 have messages from PEs 4 and 5; and PEs 3 and 7 have messages from PEs 6 and 7. This result may be obtained with only two unit-distance routing steps: an exchange and an unshuffle. The exchange transmits messages between PEs 0 and 1, PEs 2 and 3, PEs 4 and 5, and PEs 6 and 7. At this point each PE has two messages, one of which is selected to be sent out on the unshuffle connection, while the other is ignored (destroyed). The message received by each PE during the unshuffle operation is in the desired place, ready for the next stage of GCN simulation. Succeeding stages of the GCN simulation are handled

similarly. The complete GCN simulation consists of (log N - 1) repetitions of (exchange, unshuffle), (log N - 1) repetitions of (exchange, shuffle), (log N - 1) repetitions of (exchange, unshuffle), (log N - 1) repetitions of (exchange, shuffle), and one final exchange; for a total of 8 log N - 7 time units.

The optimality of this GCN numbering follows from the following considerations. Each stage of the GCN consists of N/4 complete bipartite graphs on 4 nodes. The shuffle and unshuffle network connections are not in themselves sufficient to simulate any stage of the GCN since, for example, PE 0 is only connected to itself. Thus at least one exchange step must be executed during the simulation of each GCN stage. However, a shuffle or an unshuffle must occur between consecutive exchange steps (if not, the second exchange is superfluous). Since there are 4 log N - 3 stages in this paper's GCN construction, a simulation requires 4 log N - 3 exchanges interlarded with 4 log N - 4 shuffles or unshuffles. This Subsection's numbering and associated routing algorithm realizes this lower bound.

For the special case of one-to-one message distribution patterns, 4 log N - 3 time units are sufficient to simulate the last half of the GCN (a connection network).

3c. Message broadcasting on Cube, PM2I, and WPM2I computers

The nomenclature of this section is due to Siegel [1976]. The Cube network is similar to the one implemented in Staran, the PM2I network is similar to Feng's Data Manipulator, while the WPM2I is Siegel's brainchild.

As before, let the PEs be numbered from 0 to N-1 in m = log N bits: $b_m b_{m-1} \ldots b_2 b_1$. The cube has m settings, where setting i connects $b_m \ldots b_1$ to $b_m \ldots b_{i+1} \bar{b}_i b_{i-1} \ldots b_1$. Using the natural left (0) to right (N-1) numbering for the nodes on each level of the GCN, it should be clear that simulation of any stage of the GCN takes only one time unit, so that at most 4 log N - 3 time units are required by any message distribution pattern on the Cube.

The PM2I network has 2 log N = 2m settings, corresponding to the addition or subtraction mod N of 2^i for $0 \le i < m$. The WPM2I connections are similar to those of the PM2I network, except any "carry" or "borrow" will "wrap around" to the b_{i-2}th bit. Either network can simulate a naturally numbered GCN in two time units per stage, giving a total of 8 log N - 6 time units for worst-case message broadcasting.

Of course, these bounds are cut almost in half for the special case of one-to-one message distribution patterns. Only 2 log N - 2 time units are required for a Cube simulation of a connection network (4 log N - 4 time units on the PM2I or WPM2I), using the natural numbering scheme.

4. Partitioning

The use of switching networks in the partitioning of a multiprocessing system is treated in Goke and Lipovski [1973]. They propose connecting N resources to a network flexible enough to provide private buses for disjoint "subsytems" of the resources. For example, if a particular terminal, processing unit, and memory device are to be formed into an independent subsystem, the partitioning network is instructed to form a private connection between their respective I/O ports. The partitioning networks considered in this paper will merely connect appropriate I/O ports; management of the bus thereby created for each subsystem will be the responsibility of the member resources. The most straightforward partitioning network is based on an N by N/2 crosspoint switch: each of the N resources can be independently connected to any of N/2 internal buses. While this network is simple to configure and has only constant delay, it requires $O(N^2)$ switches. Another network considered by Goke and Lipovski is an (N,N)-connector whose inputs are connected to its outputs. Although this device has only O(N log N) switches, its delay may be O(N log N). Goke and Lipovski settled on "banyan networks" with O(N log N) switches and O(log N) delay, but incomplete functionality (not all partitions could be achieved). It should be clear that an (N,N)-GCN provides unrestricted freedom of connection between any of its N outputs. This paper's GCN construction thus immediately gives a complete partitioning network with O(N log N) switches and O(log N) delay.

Actually, a GCN is an unnecessarily complex partitioning network. The resources will only be connected to the outputs of the GCN, so that the ordering of the inputs is completely arbitrary. In terms of Section 2's construction, this implies that the hyperconcentrator "front end" is superfluous and may be removed. Since at most N/2 subsystems can have more than one resource, half of the inputs to the infrageneralizer may be deleted. This and similar optimizations to the infrageneralizer, coupled with connector optimizations (Waksman [1968]), yield a partitioning network with 6N log N - 8.5N + 8 switches. For example, the following is an (8)-partitioner.

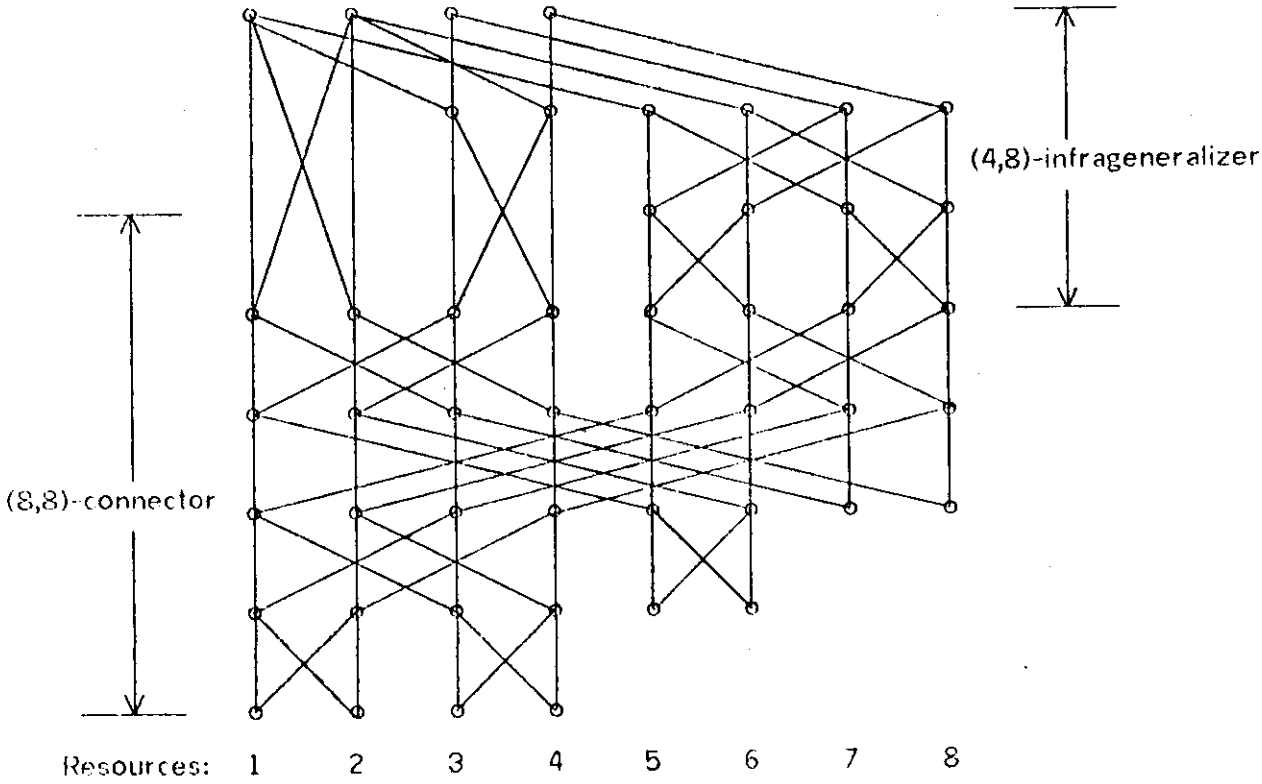


Figure 12. An (8)-partitioner.

Set-up algorithms for this network are relatively time-consuming, limiting its practicality (banyan networks can be essentially self-configuring, in O(log N) time). When a new subsystem with k resources (k>1) comes into existence, it is assigned the leftmost unused infrageneralizer input and the k leftmost unused connector inputs. The infrageneralizer can be configured in O(log N) time, since it is a banyan. However, the connector setting may need radical changes for which the best known algorithm (Waksman [1968]) requires O(N log N) time on a serial computer.

The three-way branching constructions mentioned in Section 2 lead to another partitioner with (9/log 3)N log N - (59/6)N + 10.5 switches, the best construction known to the author (note: 9/log 3 = 5.7).

The author is indebted to Nicholas Pippenger for several stimulating discussions.

References

Benes, V. E. [1965]. Mathematical Theory of Connecting Networks and Telephone Traffic, Academic Press, New York.

Flynn, M. J. [1966]. "Very High-Speed Computing Systems," Proc. IEEE, Vol. 54, pp. 1901-1909.

Goke, L. R. and Lipovski, G. J. [1973]. "Banyan Networks for Partitioning Multiprocessor Systems," First Annual Computer Architecture Conference, Gainsville, Florida, pp. 21-28.

Lawrie, D. E. [1973]. Memory-Processor Connection Networks, Ph. D. Dissertation, Dept. Computer. Sci., Univ. Illinois, Urbana, Report 557.

Masson, G. M. and Jordan, B. W. Jr. [1972]. "Generalized Multistage Connection Networks," Networks, Vol. 2, pp. 191-209.

Ofman, J. P. [1965]. "A Universal Automaton," Trans. Moscow Math. Soc., Vol. 14 (translation published by American Math. Soc., Providence, R. I., 1967, pp. 200-215).

Orcutt, S. E. [1976]. "Implementation of Permutation Functions on Illiac IV-Type Computers," IEEE Trans. on Computers, C-25, pp. 929-936.

Pippenger, N. J. [1973]. The Complexity Theory of Switching Networks, Tech. Rep. 487, Res. Lab. of Electronics, MIT.

Pippenger, N. J. [1977]. "Superconcentrators," SIAM J. Comput. Vol. 6, pp. 298-304.

Siegel, H. J. [1976]. SIMD Machine Interconnection Network Design, Tech. Rep. 198, Computer Sci. Lab., Dept. of EE, Princeton Univ.

Stone, H. S. [1971]. "Parallel Processing with the Perfect Shuffle," IEEE Trans. on Computers, C-20, pp. 153-161.

Thompson, C. D. and Kung, H. T. [1977]. "Sorting on a Mesh-Connected Parallel Computer," CACM, Vol. 20, pp. 263-271.

Waksman, A. [1968]. "A Permutation Network," JACM, Vol. 15, pp. 159-163.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* GENERALIZED CONNECTION NETWORKS FOR PARALLEL PROCESSOR INTERCOMMUNICATION		5. TYPE OF REPORT & PERIOD COVERED Interim 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) C. D. Thompson		8. CONTRACT OR GRANT NUMBER(s) N00014-76-C-0370 MCS 75-222-55
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie-Mellon University Computer Science Dept. Pittsburgh, PA 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Arlington, VA 22217		12. REPORT DATE May 1977 13. NUMBER OF PAGES 19
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

A generalized connection network (GCN) is a switching network with N inputs and N outputs that can be set to pass any of the N^N mappings of inputs onto outputs. This paper demonstrates an intimate connection between the problems of GCN construction, message routing on SIMD computers, and "resource partitioning." A GCN due to Ofman [1965] is here improved to use less than 8N log N contact pairs, making it the minimal known construction.

(continued)

20. abstract (Continued)

Any GCN construction leads to a new algorithm for the broadcast of messages among processing elements of an SIMD computer, when each processing element is to receive one message. Previous approaches to message broadcasting have not handled the problem in its full generality. The algorithm arising from this paper's GCN takes 8 log N (or 13 $N^{1/2}$) routing steps on an N element processor of the perfect shuffle (or mesh-type) variety.

If each resource in a multiprocessing environment is assigned one output of a GCN, private buses may be provided for any number of disjoint subsets of the resources. The partitioning construction derived from this paper's GCN has 6N log N switches, providing an alternative to "banyan networks" with O(N log N) switches but incomplete functionality.