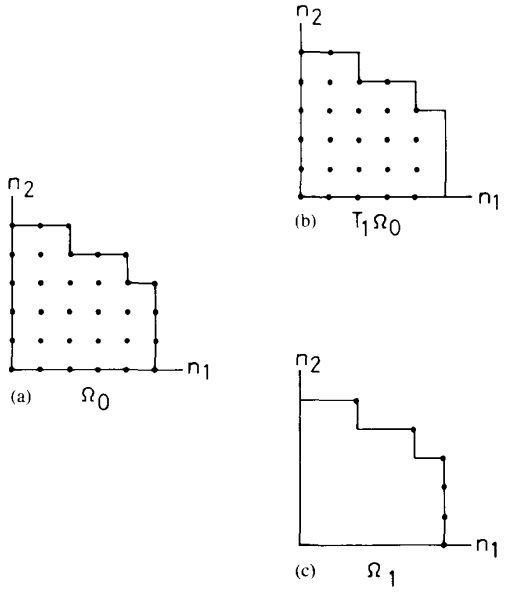

Fig. 1. A coordinate convex state space (a), the translated set (b), and the blocking states for calls of class 1 (c).

By the rule of quotient differentation, we then have from (5)

$$\frac{\partial B_i}{\partial a_j} = \frac{G(T_i\Omega_o)G(T_j\Omega_o) - G(\Omega_o)G(T_jT_i\Omega_o)}{G^2(\Omega_o)}. \quad (7)$$

As T_i and T_j obviously commute with each other, this expression is independent of the order of indexes i and j and the reciprocity relation (6) is proven. The proof is valid for any number of user classes, though Fig. 1 refers specifically to two classes only.

III. Concluding Remarks

We have shown that in a situation where several user classes compete for a common pool of resource the mutual differential influences on the blocking probabilities of the traffic streams are equal.

As a side result, we have obtained an expression, (7), for the derivative $\partial B_i/\partial a_j$ where each of the argument sets of the G expressions is coordinate convex. Such expressions lend themselves to a computation by the fast algorithms developed for the blocking problem, e.g., [4]. These derivatives are useful in the numerical inversion of the blocking functions to yield the traffic intensities as a function of the blocking probabilities.

References

[1] J. M. Aein, "A multi-user-class, blocked-calls-cleared, demand access model," *IEEE Trans. Commun.*, vol. COM-26, Mar. 1978.

[2] J. S. Kaufman, "Blocking in a shared resource environment," *IEEE Trans. Commun.*, vol. COM-29, Oct. 1981.

[3] J. W. Roberts, "A service system with heterogeneous user requirements—Application to multi-services telecommunications systems," in *Performance of Data Communication Systems and their Applications*, G. Pujolle, Ed. The Netherlands: North-Holland, 1981.

[4] B. Kraimeche and M. Schwartz, "Circuit access control strategies in integrated digital networks," in *Proc. IEEE INFOCOM '84*, San Francisco, CA, Apr. 1984, pp. 230–235.

[5] S. A. Johnson, "A performance analysis of integrated communications systems," *Brit. Telecom. Technol. J.*, vol. 3, Oct. 1985.

The Batcher-Banyan Self-Routing Network: Universality and Simplification

MADIHALLY J. NARASIMHA

Abstract—**It is shown that the Batcher-banyan network performs as a universal self-routing switch when inputs with unassigned destinations are present. This fact is demonstrated by first proving that banyan networks can realize permutations represented by bitonic sequences, and then noting that the sorted output of the Batcher network can be viewed as a bitonic sequence. Two methods are proposed for reducing the complexity of the Batcher-banyan network. In the first method, one stage of the banyan network is eliminated by assigning proper destination tags to the unassigned inputs. Second, a self-routing switch based on the binary radix sorting scheme is shown to be more economical for small number of lines.**

I. Introduction

Self-routing networks are being considered for constructing the interconnection fabric of the next generation switching systems in telecommunications [1]–[3]. They allow distributed control of the individual switches in the network and can therefore be advantageous over conventional designs in high-speed systems.

A popular method for realizing the self-routing switching fabric is the Batcher-banyan network [1]. It consists of a bitonic sorting network [4] and a banyan routing network. Both of these are multistage networks where each stage performs a fixed permutation on the incoming lines, and then routes them through a column of 2 × 2 switching elements. The connection state of any switching element in the network is completely determined by the destination tags of the inputs—hence, the terminology "self-routing." The combination of the two can implement any pattern of interconnection between N inputs and N outputs. (N is assumed to be a power of two throughout this paper.)

The Batcher sorting network by itself is capable of realizing an arbitrary permutation of N inputs. When inputs with unassigned destinations (designated here as "inactive inputs") are present, however, the routing implemented by this network is not quite right. To alleviate this problem, Huang and Knauer [1] proposed the addition of a banyan network to route the sorted output of the Batcher network to the appropriate destinations. Although they state that the combination accomplishes the desired mapping, we have not seen a formal proof of this fact in prior publications.

We show in Section III of this paper that the Batcher-banyan network is indeed a universal[1] self-routing switch. That is, it can implement any desired interconnection pattern, with decentralized control of the switching elements, even when some of the inputs have unassigned destinations. The universality of the network is demonstrated by first showing that permutations represented by bitonic sequences can be realized by a banyan network, and then noting that the sorted output of the Batcher network, in the presence of inactive inputs, may be viewed as a bitonic sequence.

Paper approved by the Editor for Communication Switching of the IEEE Communications Society. Manuscript received December 28, 1987; revised February 18, 1988.

The author is with Space, Telecommunications, and Radioscience Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA 94305.

IEEE Log Number 8822948.

[1] Wu and Feng [5] use the term "universal" to refer to a network capable of realizing all the $N!$ permutations. Our usage of this term refers to a more general situation. We allow inactive inputs to the network, and this causes additional complications in self-routing switches.

Two methods of reducing the complexity of the Batcher-banyan network are discussed in this paper. First, we show that one stage of the banyan network can be elminated by assigning a destination address midway between the lowest and the highest to the inactive inputs. This assignment does not significantly increase the complexity of the comparators at the switching nodes.

Second, we point out that a self-routing network based on binary radix sorting is simpler for small values of N. When $N = 8$, for example, this method requires seven stages compared to nine for the Batcher-banyan. The switch settings in this network are based on individual bits of the destination tags whereas the Batcher sorting network requires comparators at the switching nodes.

II. Routing and Sorting Networks

Fig. 1 shows an 8-input routing network known as the omega network [6]. It has three stages; each stage consisting of a shuffle permutation [7] and an exchange permutation. The exchange is realized by the column of 2×2 switches which can be set to either a straight through connection or a crossed connection. The inputs to the network are labeled $d_0, d_1, \cdots, d_7$ where d_k denotes the destination of the kth input. It is assumed that the d_k sequence is a permutation of the integers $0, 1, \cdots, 7$. An N-input network has $\log_2 N$ stages, and is designated as the "Ω_N network."

Goke and Lipovski [8] define a general class of banyan routing networks in their paper. We are interested in only a special class denoted as the "SW structure" (with a spread and fanout equal to two). If we transpose the graph representing such a network, and add a shuffle permutation to the input, the resulting structure is known as the "generalized cube" network [9]. This is referred to as the "banyan" network here. (The only reason for using such a terminology is that it rhymes better with the word "Batcher"!). Fig. 2 shows an example of this network for eight inputs. It is well known [9] that an N-input generalized cube network is topologically equivalent to an Ω_N network. The network of Fig. 2, for example, can be obtained from that of Fig. 1 by rearranging the second-stage switches. Because of this, we use the two terms—banyan and omega—intechangeably in this paper.

Although the two networks are functionally equivalent, drawing them out differently often provides some insights. With the banyan structure, for example, it is easy to visualize a method of building an N-input network from two $(N/2)$-input networks.

According to the standard bit-controlled routing algorithm [6] for these networks, switches in the first stage are controlled by the most significant bit (MSB) of the destination tag, those in the second by the next bit, and so on. By convention, the control bit of the top input decides the state of the switching elements. If it is zero, the switch is in the straight connection state; otherwise it is crossed. A conflict occurs if the control bits for both the top and bottom inputs of any switch are identical. Because of these conflicts, an omega network is not capable of realizing an arbitrary permutation of N-inputs. Any d_k sequence that can be completely routed with the above algorithm is known as an "Ω_N-passable sequence" [10].

Batcher's sorting network [4] consists of a sequence of bitonic sorters of increasing dimensions. A typical system for eight inputs is shown in Fig. 3. The network labeled S_k sorts a k-length bitonic sequence.[2] It consists of a $\log_2 k$-stage shuffle-exchange network [7]. Hence its topology is identical to that of a Ω_k network. (Note that S_2 is simply a 2×2 switch). The 2×2 switching elements in the S_k networks, however, are set by comparing the destination tags of the two inputs. The input with the larger tag is routed to the output

[2] A bitonic sequence [4] is a juxtaposition of two monotonic sequences, one ascending and the other descending.

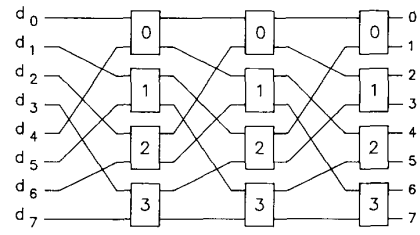

Fig. 1. A three-stage shuffle-exchange network for eight inputs (Ω_8 network). The d_k sequence denotes the destination tags.

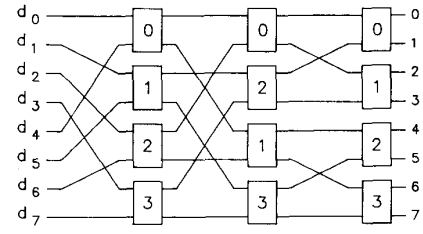

Fig. 2. A three-stage banyan network.

pointed to by the arrow. In any S_k network, the arrows for the individual switching elements point in the same direction as that shown for the corresponding box in Fig. 3. An N-input Batcher sorting network consists of $\log_2 N$ $(\log_2 N + 1)/2$ switching stages.

Although the S_k network and the Ω_k network have the same topologies, we designate them differently to emphasize the fact they are functionally different: magnitude comparisons are required to set the switches in the S_k network, whereas single bit comparisons are adequate to perform this function in the Ω_k network.

III. Universality of the Batcher-Banyan Network

In this section, we demonstrate first that a bitonic sequence can be routed properly by a banyan network. This important characteristic of bitonic sequences can be asserted by combining Batcher's results [4] on bitonic sorting, and Lawrie's observation [6] about the existence of a unique path between any input and any output in an omege network. Using this fact, we can prove the universality of the Batcher-banyan network by noting that the output of the sorting network can be viewed as a bitonic sequence in the presence of inactive inputs. These two results are stated here as a theorem and a lemma.

Theorem: A bitonic sequence that is a permutation of the integers $0, 1, \cdots, N-1$ is an Ω_N -passable sequence.

Proof: Refer to Fig. 4 which delineates Batcher's scheme for sorting a bitonic sequence. The S_N network performs a sorting operation, and can also accomplish the desired routing if the input sequence (which designates the destination tags) is a permutation of the integers $0, 1, \cdots, N-1$. This implies that the required permutation can be performed by the S_N network using a sorting algorithm to direct the switches. But the topology of the S_N network is identical to that of an Ω_N network, and it is well known [6] that, in an Ω_N network, there is one and only path for routing an input to a particular output, and that the set of paths for a given permutation is unique. From this it follows that if we set up the paths through the S_N network with the Ω_N bit-controlled algorithm, they would be identical to those set up by the sorting algorithm which controls the switching elements based on comparison of the tags. In other words, the required permutation can be accomplished by an Ω_N network (using a bit-controlled switch-setting algorithm).

Lemma: An omega network can completely route a sorted

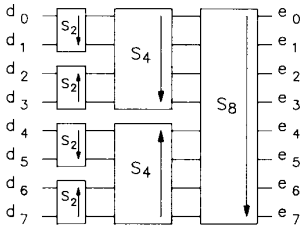

Fig. 3. Batcher's sorting network for 8 inputs. S_k denotes the sorter for a k-length bitonic sequence.

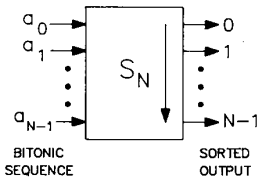

Fig. 4. A $\log_2 N$ stage sorting network for N-point bitonic sequences.

list when the inputs with unassigned destinations appear at either the high or low end of the list.

Proof: Refer to Fig. 5(a) which depicts the case where the inactive inputs are sorted to the high end by the sorting network. Let $m_o, m_1, \cdots, m_{r-1}$ be r of the missing destination tags. We can assume that they form a descending sequence without loss of generality. The Batcher network outputs an ascending sequence $a_o, a_1, \cdots, a_{N-r-1}, x, \cdots, x$ where the a_k corresponds to the active inputs and the x's denote the inactive ones. Now, the sequence $a_o, a_1, \cdots, a_{N-1-r}, m_o, m_1, \cdots, m_{r-1}$ is a bitonic sequence and can therefore be routed by the Ω_N network.

It is not necessary to actually assign the missing tags to the inactive inputs to accomplish the desired routing. This is taken care of automatically by using the modified bit-controlled routing algorithm [1] depicted in Fig. 5(b). This algorithm sets up the switches so that the active inputs have routing priority whereas the tags of the inactive inputs assume the appropriate missing values to make the combined sequence appear bitonic to the Ω_N network.

The case where the inactive inputs are sorted to the low end of the list can be treated similarly.

IV. Reducing the Complexity of the Batcher-Banyan Network

It is possible to eliminate one stage of the banyan network in the Batcher-banyan switch by assigning destination tags midway between the lowest and the highest to the inactive inputs. We demonstrate a method for accomplishing this in this section.

Fig. 6(a) depicts an N-input Batcher sorting network with some details shown for the S_N network. (The S_N network shown is based on the banyan structure.) The $(N/2)$-point sorters at the input consist of the usual $S_2, S_4, \cdots, S_{N/2}$ sequence of networks (refer to Fig. 3). The 2×2 switches in these sorters are normally set by comparing the destination tags at the input. If both active and inactive inputs appear at a switching element, however, we simply observe the MSB of the active input and set the switch state according to the scheme shown in Fig. 6(b). This is equivalent to assigning a tag value between $(N/2 - 1)$ and $N/2$ to the inactive inputs. The combined output of the two $(N/2)$-point sorters is a bitonic sequence.

The first stage of the following S_N network consists of a column of 2×2 switches. These switches route the input with

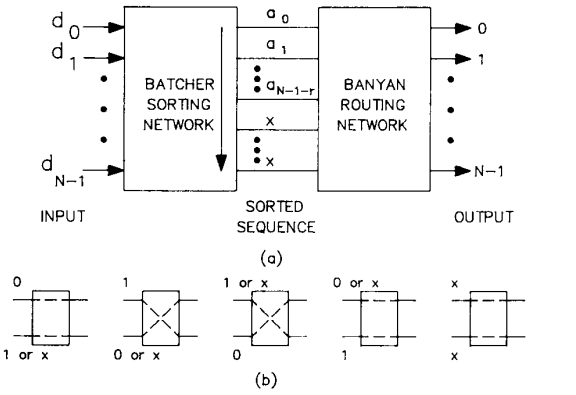

Fig. 5. (a) The Batcher-banyan network with the inactive inputs (denoted by x) sorted to the high end by the Batcher network. (b) Control of the 2×2 switching elements in the banyan network.

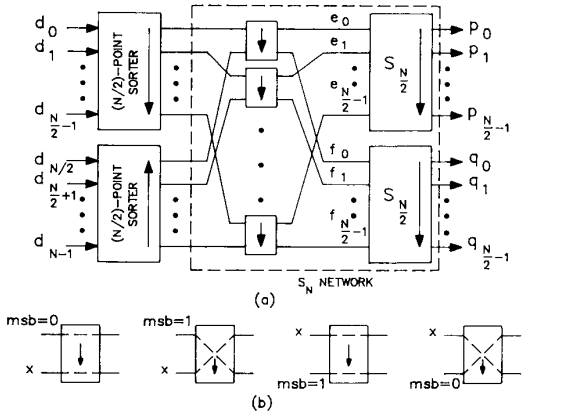

Fig. 6. (a) An N-point Batcher sorting network. (b) Switching of active and inactive inputs inside the $(N/2)$ point sorters.

the lower tag to the top output (the e_k sequence) and the one with the higher tag to the bottom output (the f_k sequence). Since the input sequence is the juxtaposition of ascending and descending sequences of equal length—a specific type of bitonic sequence—this first stage switching operation routes the $N/2$ inputs with the lowest tags to the upper half and the remaining to the lower half [11, p. 234]. Considering the above-mentioned constraints on the tags of the inactive inputs, we can conclude that the active inputs having zero for the MSB of the tag are switched to the e_k sequence while those with the MSB of tag equalling one are routed to the f_k sequence. Since both of these sequences are bitonic [4], they can be transformed to sorted lists by the following $S_{N/2}$ networks to yield the p_k and q_k outputs. Note that the inactive inputs are sorted to the high end of the p_k sequence whereas they appear at the low end in the q_k sequence. From the lemma of the previous section, it is obvious that two $\Omega_{N/2}$ networks can route the p_k and q_k sequences to their proper destinations. This saves one stage of the banyan network. Also note that the first stage switches of the S_N network need to sort based on only the MSB of the tag.

V. A Self-Routing Network Based on Binary Radix Sorting

The idea behind this method is to first sort the inputs into two lists based on the MSB of the destination tag. These lists are sorted again, but now based on the next bit, and so on. The inactive inputs are assigned a tag value such that the active

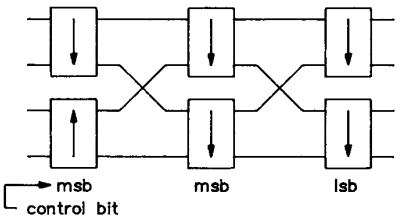

Fig. 7. Self-routing network based on binary radix sorting for four inputs.

inputs are always routed to the proper half at each step of sorting. To begin with, for example, destination addresses with MSB = 0 are considered to be lower than the tags assigned to the inactive inputs whereas those with MSB = 1 are considered to be larger.

Referring again to to Fig. 6(a), if the input is sorted based only on the MSB of the destination tags by the front end $(N/2)$-point sorters, constraining the tags of the inactive inputs as mentioned above, the desired splitting would be achieved after the first stage switching in the S_N network. That is, the e_k sequence consists of those inputs having the MSB of the destination tag equal to zero while the f_k sequence constitutes the remaining.

Routing networks for an arbitrary number N of inputs can be constructed by using this scheme iteratively. The next step, for example, consists of four $(N/4)$-point sorters operating on the e_k and f_k sequences, followed by a column of comparison-directed 2×2 switching elements. The sorting and comparison is now based on the bit next to the MSB, and a tag value between $(N/4 - 1)$ and $N/4$ is assigned to the inactive inputs. After this step the inputs would be routed to four groups according to the two most significant bits of the tags. And we can continue this procedure till all the inputs are properly routed.

Figs. 7 and 8 show self-routing networks for $N = 4$ and $N = 8$ based on this principle. The switching elements in these networks are controlled according to the scheme depicted in Fig. 9. Three stages are sufficient for the 4-line switch while seven stages are needed for the 8-input case. These are more economical than the corresponding Batcher-banyan networks which require five and nine stages, respectively.

When $N = 2^n$, this method requires $n(n^2 + 5)/6$ stages, each stage consisting of $N/2$ switching elements. For the case $N = 16$, both the Batcher-banyan network and the present network need 14 stages. The control of the switching elements, however, is simpler for the binary radix sorting network. This network does not appear to be attractive for larger values of N.

VI. Conclusion

Switching fabrics of future telecommunication systems should be capable of interfacing with very high-speed fiber optic transmission rates. This necessitates some form of distributed control to avoid overloading of the central processor. A self-routing network is attractive in these systems because, in such a network, the switching elements by themselves determine their connection states.

Huang and Knauer [1] first proposed the combination of a Batcher sorting network and a banyan routing network to realize the self-routing interconnection fabric. Since then, others [2], [3] have studied its applications in both circuit and packet switching. Integrated circuit chips have been designed to realize this function as well.

The universality of the Batcher-banyan network has been tacitly assumed so far. It is not obvious (at least to this author!) why the Batcher network produces a sequence that can be routed by the banyan network, when there are inputs with unassigned destinations. We have given theoretical justifications to this assumption by proving an important property of bitonic sequences—they can be routed through banyan networks—and then showing how to view the output of the sorting network as a bitonic sequence.

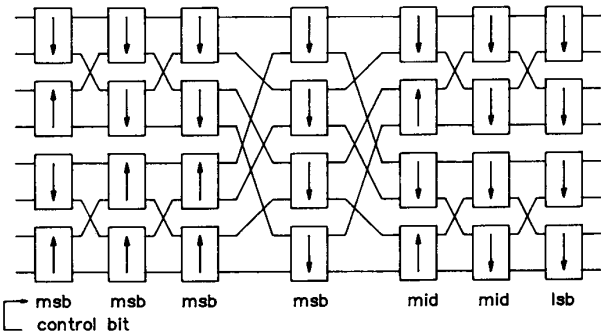

Fig. 8. An 8-input self-routing network based on binary radix sorting.

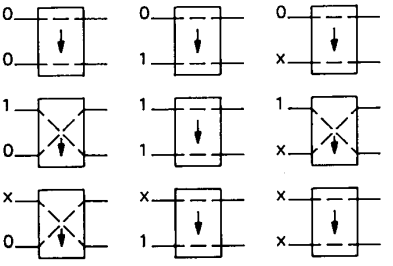

Fig. 9. Control of the switching elements in binary radix sorting. Inactive inputs are denoted by x.

This self-routing network, however, is significantly more complex than a switch with a central control unit. To realize a 1024×1024 switch, for example, the Batcher-banyan network requires 65 switching stages, whereas 19 stages are adequate in a globally controlled Benes binary network. This great disparity motivated us to look at methods for simplifying the Batcher-banyan network and to study modified architectures of reduced complexity. This paper has reported on incremental gains in both these directions.

References

[1] A. Huang and S. Knauer, "Starlite: A wideband digital switch," presented at Proc. GLOBECOM'84, Nov. 1984.

[2] C. Day, J. Giacopelli, and J. Hickey, "Applications of self-routing switches to LATA fiber optic networks," presented at Conf. Proc. Int. Symp. Switching, Phoenix, AZ, Mar. 1987.

[3] J. Hui and E. Arthurs, "A broadband packet switch for integrated transport," *IEEE J. Select. Areas Commun.*, vol. SAC-5, pp. 1264–1273, Oct. 1987.

[4] K. E. Batcher, "Sorting networks and their applications," in *1968 Spring Joint Computer Conf., AFIPS Proc.*, vol. 32, pp. 307–314.

[5] C. Wu and T. Feng, "The universality of the shuffle-exchange network," *IEEE Trans. Comput.*, vol. C-30, pp. 324–332, May 1981.

[6] D. H. Lawrie, "Access and alignment of data in an array processor," *IEEE Trans. Comput.*, vol. C-24, pp. 1145–1155, Dec. 1975.

[7] H. S. Stone, "Parallel processing with the perfect shuffle," *IEEE Trans. Comput.*, vol. C-20, pp. 153–161, Feb. 1971.

[8] L. R. Goke and G. J. Lipovski, "Banyan networks for partitioning multiprocessing systems," in *Proc. 1st Annu. Comput. Architect. Conf.*, Dec. 1973, pp. 21–28.

[9] H. J. Siegel and S. D. Smith, "Study of multistage SIMD interconnection networks," in *Proc. 5th Annu. Symp. Comput. Architect.*, Apr. 1978, pp. 223–229.

[10] K. Y. Lee, "On the rearrangeability of $2(\log_2 N) - 1$ stage permutation networks," *IEEE Trans. Comput.*, vol. C-34, pp. 412–425, May 1985.

[11] D. E. Knuth, *The Art of Computer Programming: Vol, 3: Sorting and Searching*. Reading, MA: Addison-Wesley, 1973.