ACKNOWLEDGMENT

The authors wish to acknowledge the suggestions of the late Dr. R. T. Gregory.

REFERENCES

[1] R. T. Gregory, *Error-Free Computation: Why It Is Needed and Methods For Doing It*. Huntington, NY: Krieger, 1980.

[2] N. S. Szabo and R. I. Tanaka, *Residue Arithmetic and Its Application to Computer Technology*. New York: McGraw-Hill, 1967.

[3] F. J. Taylor and C. V. Huang, "An autoscale residue multiplier," *IEEE Trans. Comput.*, vol. C-31, pp. 321-325, Apr. 1982.

[4] W. K. Jenkins and B. J. Leon, "The use or residue number systems in the design of finite impulse response digital filters," *IEEE Trans. Circuits Syst.*, vol. CAS-24, pp. 191-201, Apr. 1977.

[5] J. H. McClellan and C. M. Rader, *Number Theory in Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall Signal Processing Series, 1979.

[6] J. J. Thomas, J. M. Keller, and G. N. Larsen, "The calculation of multiplicative inverses over *GF*(*P*) efficiently where *P* is a Mersenne prime," *IEEE Trans. Comput.*, Mar. 1986.

[7] G. E. Collins, "Computing multiplicative inverses in GF(P)," *Math. Computat.*, vol. 23, p. 197, Jan. 1969.

[8] ——, "The computing time of the Euclidean algorithm," *SIAM J. Computat.*, vol. 3, Mar. 1974.

[9] J. J. Thomas and S. R. Parker, "A viable technique for calculating algorithms to any specified accuracy," in *Proc. 2nd Euro. Signal Processing Conf.*, Erlangen, West Germany, Sept. 1983.

[10] H. T. Kung and D. E. Leiserson, "Algorithms for VLSI processor arrays," in *Introduction to VLSI Systems*, C. Mead and I. Conway, Eds. Reading, MA: Addison-Wesley, 1980.

[11] D. E. Knuth, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*. Reading, MA: Addison-Wesley, 1981.

[12] E. V. Krishnamurthy, T. M. Rao, and K. Subramanian, "Finite segment *P*-adic number systems with applications to exact computation," in *Proc. Indian Academy Sci.*, 1975, vol. 81A, pp. 58-79.

[13] R. T. Gregory, "The use of finite-segment *P*-adic arithmetic for exact computations," *BIT*, vol. 18, pp. 282-300, 1978.

[14] ——, "Error-free computation with rational numbers," *BIT*, vol. 21, pp. 194-202, 1981.

[15] P. Kornerup and R. T. Gregory, "Mapping integers and Hensel codes onto Farey fractions," *BIT*, vol. 23, pp. 9-20, 1983.

[16] E. C. Hehner and R. N. Horspool, "Exact arithmetic using a variable-length *P*-adic representation," in *Proc. 4th IEEE Symp. Comput. Arithmetic*, 1978, pp. 10-14.

[17] J. A. Howell and R. T. Gregory, "Solving linear equations using residue arithmetic—Algorithm II," *BIT*, vol. 10, pp. 23-37, 1970.

[18] E. V. Krishnamurthy, "Matrix processors using *P*-adic arithmetic for exact linear computations," *IEEE Trans. Comput.*, vol. C-26, pp. 633-639, 1977.

[19] ——, "Exact inversion of a rational polynomial matrix using finite field transforms," *SIAM J. Appl. Math.*, vol. 35, pp. 453-464, 1978.

[20] T. M. Rao, "Error-free computation of characteristic polynomial of a matrix," *Computat. Math. Appl.*, vol. 4, pp. 61-65, 1978.

[21] W. S. Brown, "Euclid's algorithm and computation of polynomial GCD's," *J. Ass. Comput. Mach.*, pp. 478-504, Oct. 1971.

[22] W. S. Brown and J. F. Traub, "On Euclid's algorithm and the theory of subresultants," *J. Ass. Comput. Mach.*, pp. 505-514, Oct. 1971.

[23] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[24] E. Isaacson and H. B. Keller, *Analysis of Numerical Methods*. New York: Wiley, 1966.

[25] J. J. Thomas and S. R. Parker, "A highly efficient recursive digital filter using single-modulus Mersenne prime residue arithmetic," *IEEE Trans. Circuits Syst.*, submitted for publication.

[26] I. S. Reed and T. K. Troung, "The use of finite fields to compute convolutions," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 208-213, Mar. 1975.

[27] R. T. Gregory and E. V. Krishnamurthy, *Methods and Applications of Error-Free Computation*. New York: Springer-Verlag, 1984.

[28] G. Bioul, M. Davis, and J. J. Quisquater, "A computational scheme for an adder modulo ($2^N - 1$)," *Digital Processes*, vol. 1, 1975.

A New Benes Network Control Algorithm

KYUNGSOOK YOON LEE

Abstract—**A new Benes network control algorithm is presented. Unlike the original looping algorithm, the new algorithm is not recursive. In this algorithm ($N \times N$) Benes network is viewed as a concatenation of two subnetworks SN1 and SN2. The first (log $N - 1$) stages of a Benes network correspond to SN1, and the remaining log N stages correspond to SN2. SN1 is controlled by a full binary tree of set partitioning functions, called a Complete Residue Partition Tree, and SN2 is bit controlled. The new control algorithm sets switches one stage at a time, stage by stage.**

Index Terms—**Benes network, bit control, control algorithm, permutation, rearrangeable network, set partition, subnetwork.**

I. INTRODUCTION

The Benes network [1], which is a member of Clos' type networks [2], is a well known rearrangeable network. An interconnection network (IN) is rearrangeable if its permitted states realize every input-to-output permutation. A control algorithm is described in [7] for the Benes network. This control algorithm, called the "looping algorithm," is based upon the recursive configuration of the Benes network. As shown in Fig. 1, an ($N \times N$) Benes network with $N = p^n$, $p \geq 2$, consists of a switching stage, a ($p \times p^{(n-1)}$) unshuffle connection [8], followed by a stack of ($p^{(n-1)} \times p^{(n-1)}$) Benes networks as the middle stage, a ($p \times p^{(n-1)}$) shuffle connection [8], and a switching stage. The looping algorithm sets switches in the two outer switching stages, and recursively sets switches for smaller Benes networks of the middle stage. Thus, it works in the direction of outside-in toward the center stage working on two outer switching stages at a time until it reaches the single center stage consisting of $p^{(n-1)}$ copies of ($p \times p$) switching elements.

In this paper we present a new Benes network control algorithm which is not recursive. The new control algorithm sets switches one stage at a time, starting from the leftmost stage heading for the rightmost stage. The stage-by-stage switch setting localizes the control information within a stage, as we do not need the knowledge on the switch setting of another stage. This is not the case for the looping algorithm, and can be an advantage for the VLSI implementation of the Benes network. It will reduce the interchip (or interboard) information transfer significantly. The time complexities of the new algorithm are $O(N \log N)$ for a serial control and $O(N)$ for a parallel control, which are the same as those of the looping algorithm. Throughout the paper, logarithm of base 2 is assumed.

The new Benes control algorithm is a direct consequence of the earlier work [4] in which a uniform rearrangeability proof method was developed for (2 log $N - 1$) stage permutation networks. As the proof was a constructive one, it also yielded a uniform control algorithm for (2 log $N - 1$) stage rearrangeable networks.

In the new Benes control algorithm two different control schemes are used for the first (log $N - 1$) stages (SN1) and the remaining log N stages (SN2). A residue set partitioning control is used for SN1, and a bit control is used for SN2. Thus, SN2 is controlled in real time, and the slow control remains only within SN1. This fact can be regarded as another advantage compared to the looping algorithm.

This paper consists of six sections. After an introduction in Section

Manuscript received October 8, 1984; revised September 10, 1985. This work was supported in part by the National Science Foundation under Grant US NSF MCS80-01561, and in part by the University of California under Grant US DOE SBC UCAL 5498609.

The author is with the Department of Computer and Information Science, The Ohio State University, Columbus, OH 43210.

IEEE Log Number 8613385.

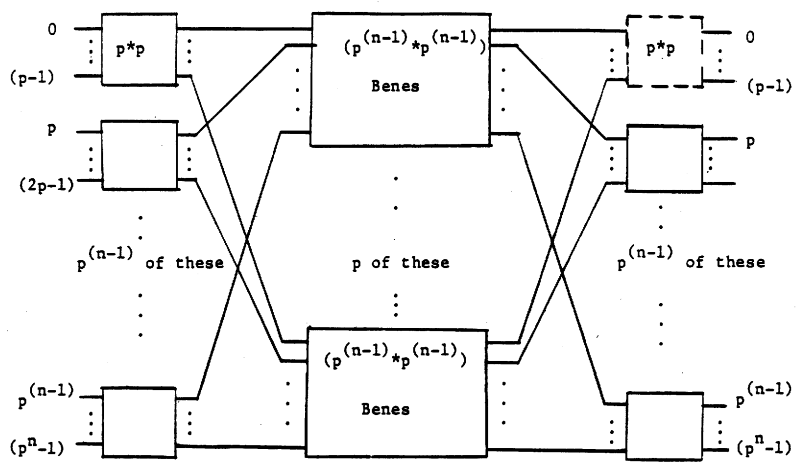

Fig. 1. A ($p^n \times p^n$) Benes network.

I, definitions are given in Section II. A new Benes network control algorithm is described in Section III for ($2^n \times 2^n$) Benes network. Hardware redundancy is discussed in Section IV. The results are generalized informally to ($p^n \times p^n$) case with $p > 2$ in Section V. Finally conclusions are drawn in Section VI.

II. Definitions

We define two key concepts used for the new control algorithm.

Definition 1: A *Complete Residue System modulo m, CRS-(mod m),* is a set of m integers which contains exactly one representative of each residue class mod m.

Definition 2: A *Complete Residue Partition, CRP,* is a partition of a CRS(mod 2^k) into two CRS's(mod $2^{(k-1)}$), $k \geq 1$.

A CRP can be performed in many different ways. As it is the same problem as dividing a pile of 2^k objects consisting of pairs of $2^{(k-1)}$ distinct objects into two piles of $2^{(k-1)}$ distinct objects, there are $2^{(2^{(k-1)}-1)}$ different ways to perform a CRP.

As an example, $\{7, 5, 0, 1, 4, 6, 2, 3\}$ is a CRS(mod 8). When a CRP is performed on this set, we get two CRS's(mod 4), $\{7, 5, 0, 2\}$ and $\{1, 4, 6, 3\}$ (or $\{7, 5, 4, 2\}$ and $\{0, 1, 6, 3\}$, etc.) If we apply two independent CRP's on these two CRS's(mod 4), we have four CRS's(mod 2). In general starting from a CRS(mod 2^n), a full binary tree of CRP's, called a CRPT, generates a full binary tree of CRS's with the leaves being $2^{(n-1)}$ CRS's(mod 2).

Definition 3: A *Complete Residue Partition Tree, CRPT,* is a full binary tree of CRP's. The root node of a CRPT is a CRP on a CRS(mod 2^n), and two sons of a node are two CRP's on the two CRS's produced by the parent node. The leaves of a CRPT are $2^{(n-2)}$ CRP's each working on a CRS(mod 4) independently to create $2^{(n-1)}$ CRS's(mod 2).

A CRPT is shown in Fig. 2.

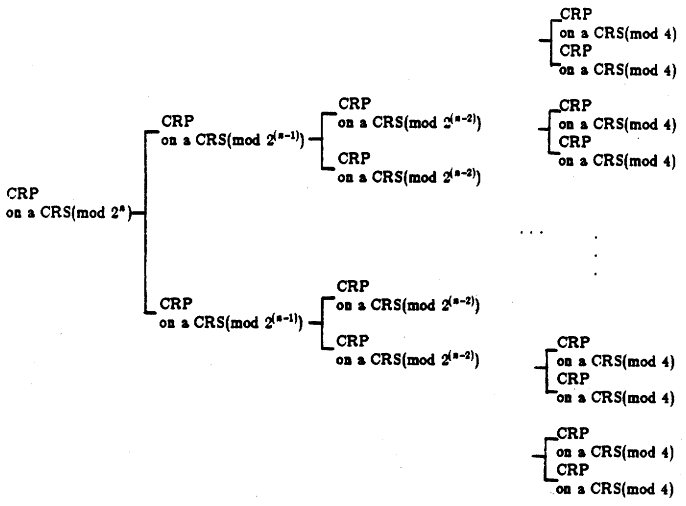

Fig. 2. A CRPT: a full binary tree of CRP's.

III. A New Benes Network Control Algorithm

For the new control algorithm an ($N \times N$) Benes network (BN) is regarded as a concatenation of two subnetworks, SN1 being the first ($\log N - 1$) stages of BN, and SN2 being the remaining $\log N$ stages. First the control algorithm will be described for $N = 2^n$ case where the basic elements of the network are (2×2) switches. It can be generalized easily for the $N = P^n$ case with ($p \times p$) switches, $p > 2$.

The BN can be represented as

$$\begin{aligned} \mathrm{BN} &= (E^0\sigma_n^{-1} \cdot E^1\sigma_{(n-1)-1}^{-1} \cdots E^{(n-2)}\sigma_2^{-1}) \\ &\quad \cdot (E^0\sigma_2 \cdot E^1\sigma_3 \cdots E^{(n-2)}\sigma_n \cdot E^{(n-1)}) \\ &= \mathrm{SN1} \cdot \mathrm{SN2} \end{aligned}$$

where σ_k^{-1} and σ_k stand for a segment unshuffle connection and a segment shuffle connection with the segment size equal to 2^k, respectively, and E represents a switching stage.

In the new control algorithm SN2 is bit controlled. Each of n destination tag bits is used once as a control bit. If the control bit of the upper input number is "0," the switch is set straight. If it is "1," the switch is set cross.

As pointed out in the earlier work [4], this kind of a bit control is a natural control for $\log N$ stage IN's; it realizes all permutations and only permutations that can pass through the network without a conflict. A formal proof of this fact is given in [4] for the inverse omega network [3]. An intuitive argument may be the fact that any $\log N$ stage IN in the literature can be viewed as N copies of 1-to-2^n demultiplexer trees overlaid in the certain manner [8], [10]. To reach an output point from an input point, we only have to follow the unique path decided by the demultiplexer tree with the input point as the root node. The bit control does this.

The selection of n control bits should be based on the particular configuration of the overlaid demultiplexer trees, and can be regarded as an important characteristic of an n stage IN. Refer to [5] for further elaboration on this point.

In this paper we state the following without a formal proof.

Control Bit Sequence for SN2 of the Benes Network

Let $b = b_{(n-1)}b_{(n-2)} \cdots b_0$ denote a destination tag. The control bit C_i for the ith switching stage E^i of SN2 should be the bit $b_{(n-1-i)}$ to realize all and only permutations that can pass through SN2 without a conflict.

$$C_i = b_{(n-1-i)}.$$

This is a direct consequence of the segment shuffles used as connection patterns in SN2.

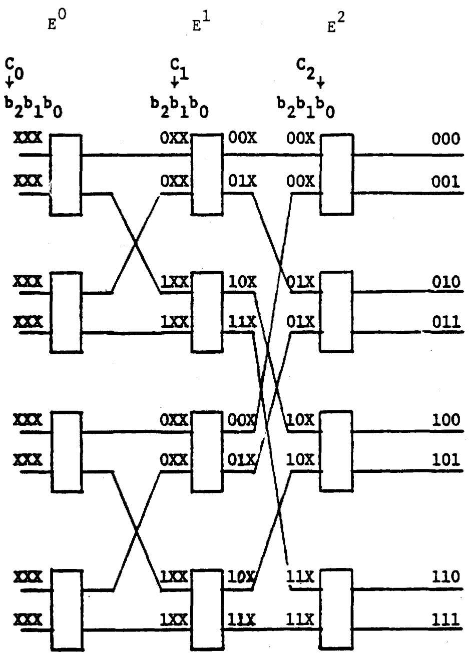

Fig. 3. The control bit selection for SN1 of a Benes network.

We can see the correctness of this control sequence intuitively in Fig. 3. To be connected to the output ports 000 through 111 for $N = 8$ case, the high order two bits of the input numbers to E^2 should have already been sorted in ascending order. The least significant bit b_0 can be in any order. When b_0 is used as the control bit C_2 for E^2, that bit is sorted by the bit control to reach the right output port. Similarly, the most significant bits of the input sequences to upper and lower halves of E^1 should be in ascending order. When b_1 is used as the control bit C_1 for E^1, the two high order bits of the output sequences will be in ascending order. In general $b_{(n-1-i)}$ should be used as C_i for E^i of SN2. The same bits are used as the control bits for the last n stages of the self-routing Benes network [6].

Now the SN2-passable condition for an input sequence to SN2 can be informally stated as follows.

SN2-Passable Condition

An input permutation is SN2-passable if and only if groups of 2^j consecutive input numbers form CRS's(mod 2^j) for the j control bits $C_{(j-1)}C_{(j-2)} \cdots C_0$, $n > j \geq 1$. In other words the groups of two input numbers should have ("0," "1,") pairs for C_0. The groups of four input numbers should have ("00," "01," "10," "11") quadruples for C_1C_0, and so on.

The SN2-passable condition is given here very informally, just to convey the ideas and to avoid necessary formalization of notations. Readers interested in a more formal approach are referred to [4], [5].

As the SN2-passable condition has been defined now, we set the switches of SN1 to transform an arbitrary input permutation to an SN2-passable permutation. A CRPT is used to do this. It is shown in [4] how a group of $2^{(k-1)}$ switches can be set to perform a CRP on the 2^k input numbers which form a CRS(mod 2^k), so that the resulting two CRS's(mod $2^{(k-1)}$) are separated into the upper and the lower output ports. For the completeness of our discussion we cite an example from [4] and repeat it in Fig. 4. In this example a CRP is performed on a CRS(mod 8) by setting four switches in such a way that the resulting two CRS's(mod 4) from the CRP reside on the upper and the lower output ports. The switch marked as ① is set straight arbitrarily. By doing that we missed the residue class 1(mod 4) for the upper output port. So we find another residue class 1(mod 4) which is number 1 in the switch marked as ② and set the switch ② cross. By doing this we missed the residue class 0 for the upper output port. So we connect the switch marked as ③ cross, and so on.

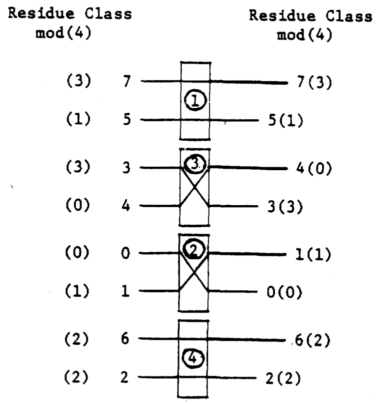

Fig. 4. A CRP can always be realized on a set of switches. (Circled numbers show a possible order of switch setting, where the input CRS(mod 8) {7, 5, 3, 4, 0, 1, 6, 2} is partitioned to two CRS's(mod 4), {7, 4, 1, 6} and {5, 3, 0 2}).

Observe that the switch ① is redundant, and the time complexity of a CRP control on N numbers is $O(N)$.

A CRPT Control for SN1

SN1 is controlled by a CRPT. For E^0, a CRP is needed to form two CRS's(mod $2^{(n-1)}$) for bits $b_1b_2 \cdots b_{(n-1)}$. For E^1 two CRP's are needed to work on the two CRS's(mod $2^{(n-1)}$ for bits $b_1b_2 \cdots b_{(n-2)}b_{(n-1)}$. Continuing this process we form a CRPT of $(n-1)$ levels, one level per switching stage.

We summarize the above discussion as follows.

A New Control Algorithm for the $(N \times N)$ Benes Network, $N = 2^n$

1. The N numbers of the destination permutation in the binary representation are the input to the network.
2. Perform 2^i CRP's on the 2^i CRS's(mod $2^{(n-i)}$), formed by bits $b_i \cdots b_{(n-1)}$ for E^i of SN1 to get $2^{(i+1)}$ CRS's(mod $2^{(n-i-1)}$) on bits $b_{(i+1)} \cdots b_{(n-1)}$ for $0 \leq i \leq (n-2)$.
3. The remaining n switching stages are controlled by $b_{(n-1)}, b_{(n-2)}, \cdots, b_0$ used as control bits $C_0, C_1, \cdots, C_{(n-1)}$, respectively. If $C_i = 0$, set a switch in E^i of SN2 straight, else if $C_i = 1$ set it cross, $0 \leq i \leq (n-1)$.

An example of the new control algorithm is shown in Fig. 5 for the bit reversal permutation on (8×8) BN. For E^0 of SN1, we set the switches so that bits b_1b_2 of four numbers on four upper (lower) output ports form a CRS(mod 4)—($\underline{00}0$, $\underline{01}0$, $1\underline{01}$, $1\underline{11}$) and ($1\underline{00}$, $1\underline{10}$, $0\underline{01}$, $0\underline{11}$), respectively. For E^1 of SN1, we perform two CRP's independently on the two groups of switches marked 1 and 2 to form four CRS'(mod 2) for bit b_2 (000, 101), (010, 111), (100, 001), and (110, 011). SN2 is bit controlled. Another example is shown in Fig. 6 for a shuffle permutation on the (16×16) BN.

The time complexity of the SN1 control is $O(N \log N)$, since SN1 consists of $\log N - 1$ stages where each stage is controlled in $O(N)$ time required for the CRP control. The time complexity of the SN2 control is $O(\log N)$, since SN2 is bit controlled. Thus, the time complexity of the overall algorithm is $O(N \log N)$ for a single control. For a multiple control all the CRP's on the same stage can be done in parallel. Thus, the time complexity becomes $O(N)$.

Observe that the switches are set stage by stage starting from the leftmost stage of BN, and continuing to the rightmost stage. All the information required to set switches of each stage is contained within the input sequence and thus no other information exchange is necessary among stages. This property can be exploited to pipeline the switch settings for the consecutive permutations in MIMD environments to decrease the control time by a factor of $O(\log N)$. This is not the case for the looping algorithm. Also note that more than half of the network is bit controlled in real time. Thus, the bottleneck of the control time remains within SN1.

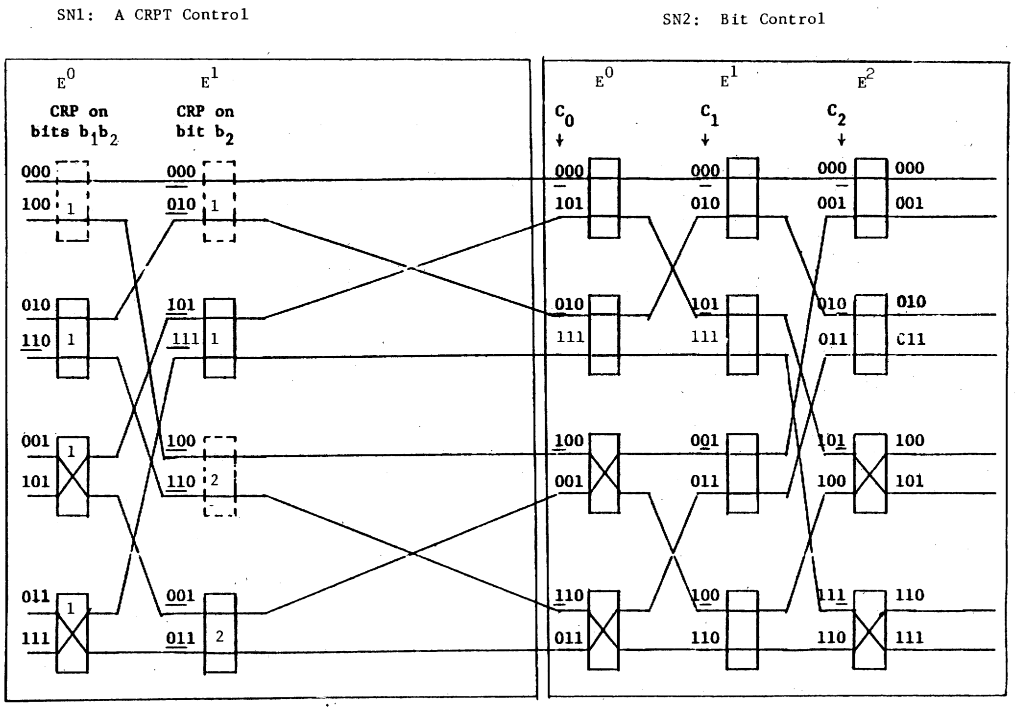

Fig. 5. A bit reversal on (8 × 8) Benes network by the new control algorithm. (Dashed switches are redundant.)

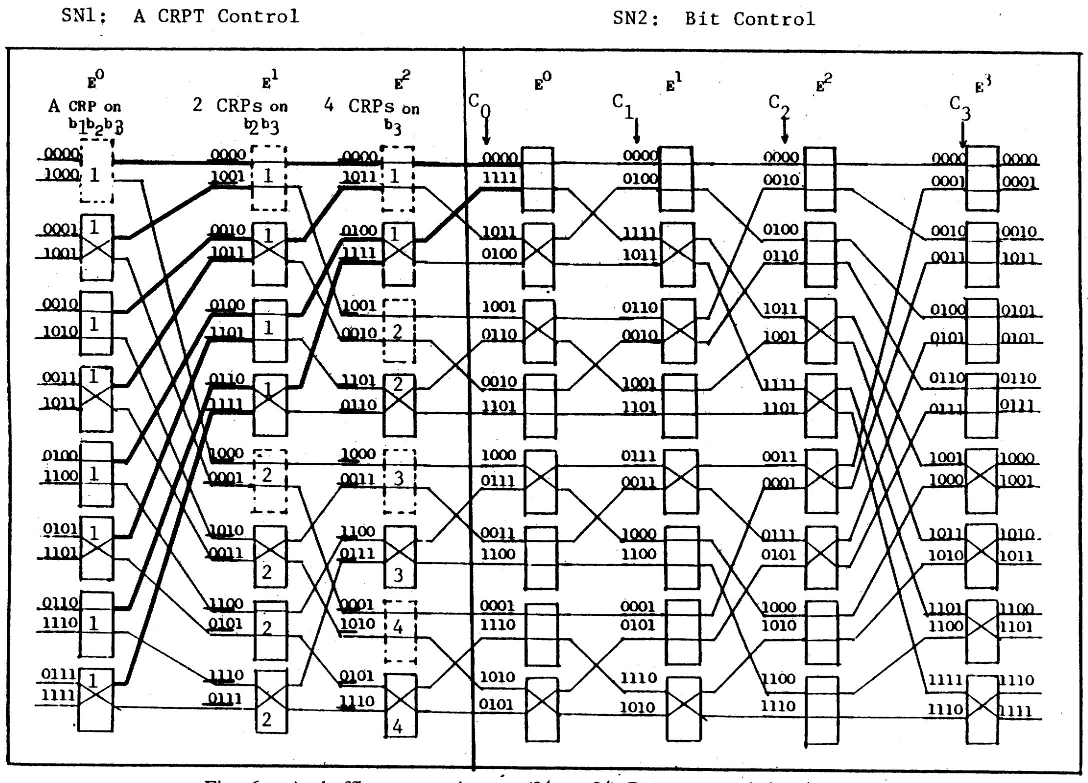

Fig. 6. A shuffle permutation on ($2^4 \times 2^4$) Benes network by the new control algorithm. (Dashed switches are redundant.)

IV. Hardware Redundancy in the Benes Network

As noted in the previous section one switch per CRP is redundant because we can set one switch arbitrarily. In a CRPT we have

$$\begin{aligned}1+2+\cdots 2^{(n-2)} &= 2^{(n-1)}-1\\ &=(N/2-1)\end{aligned}$$

CRP's. Thus, $(N/2 - 1)$ switches of the BN is redundant, which means we have

$$(2n-1)N/2-(N/2-1)=(N\log N-N+1)$$

switches in the BN. This matches the switch requirement given by Waksman [9]. In Figs. 5 and 6 redundant switches are marked with dashed lines.

V. Generalization to the $(p^n \times p^n)$ Benes Network, $p > 2$

The new control algorithm can be easily generalized for $N = p^n$ BN with $(p \times p)$ switches, $p > 2$. We need to generalize both CRP and the bit control to include a radix greater than 2.

Definition 4: A complete residue partition in radix r, CRP(r), is a partition of a CRS(mod r^k) into r CRS's(mod $r^{(k-1)}$).

Digit controlled switch setting sets a switch in such a way that the outcoming control digits are sorted in ascending order. It is a generalization of the bit control.

For $(p^n \times p^n)$ BN, SN1 is controlled by a CRPT consisting of CRP's(p), and SN2 is digit controlled. Refer to [5] for examples.

VI. Conclusions

Extending the earlier work [4] in which a general proof method for (2 log $N - 1$) stage rearrangeable networks was developed, a new Benes network control algorithm was presented. This new control algorithm does not view the Benes network as a recursive network, but rather as a concatenation of two subnetworks SN1 and SN2.

SN1 corresponds to the first (log $N - 1$) stages, and SN2 corresponds to the remaining log N stages of the Benes network. SN1 is controlled by a full binary tree of set partitioning functions, called a CRPT, and SN2 is bit controlled. Thus, the new control algorithm sets switches stage by stage.

The time complexity of the new control algorithm is the same as that of the looping algorithm. But it has some advantages compared to the looping algorithm. First, it sets switches stage by stage, one stage at a time. Thus, it eliminates the information exchange among different stages, making the pipelining of switch setting feasible for MIMD environments. Second, SN2 is bit controlled in real time, and the bottleneck remains only within SN1.

In the new algorithm one switch per CRP is redundant, which gives us exactly the same number of switch requirements of the reduced Benes network [9].

The new algorithm applies to any $(p^n \times p^n)$ Benes network for $p \geq 2$.

Acknowledgment

The author is grateful to Prof. D. Kuck. Without him this work would have not been possible.

References

[1] V. E. Benes, "Permutation groups, complexes, and rearrangeable connecting networks," *Bell. Syst. Tech. J.*, vol. 43, pp. 1619–1640, July 1964.

[2] C. Clos, A study of non-blocking switching networks," *Bell. Syst. Tech. J.*, vol. 32, pp. 406–424, Mar. 1953.

[3] D. H. Lawrie, "Access and alignment of data in an array processor," *IEEE Trans. Comput.*, vol. C-25, pp. 1145–1155, Dec. 1975.

[4] K. Y. Lee, "On the rearrangeability of a (2 log $N - 1$) stage permutation network," in *Proc. 1981 Int. Conf. Parallel Processing*, 1981, pp. 221–228; also in *IEEE Trans. Comput.*, vol. C-34, pp. 412–425, May 1985.

[5] ——, "Interconnection networks and compiler algorithms for multiprocessors," Ph.D. dissertation, Dep. Comput. Sci., Univ. Illinois, Urbana-Champaign, Rep. 83-1125, 1983.

[6] D. Nassimi and S. Sahni, "A self-routing Benes network and parallel permutation algorithms," *IEEE Trans. Comput.*, vol. C-30, pp. 332–340, May 1981.

[7] D. C. Opferman and N. T. Tsao-Wu, "On a class of rearrangeable switching networks, Part I: Control algorithm," *Bell. Syst. Tech. J.*, vol. 50, pp. 1579–1600, 1971.

[8] J. H. Patel, "Processor-memory interconnections for multiprocessors," in *Proc. 6th Ann. Symp. Comput. Architecture*, Apr. 1979, pp. 168–177.

[9] A. Waksman, "A permutation network," *J. Ass. Comput. Mach.*, vol. 15, pp. 159–163, Jan. 1968.

[10] C. L. Wu and T. Feng, "The reverse-exchange interconnection network," in *Proc. Int. Conf. Parallel Processing*, 1979; also in *IEEE Trans. Comput.*, vol. C-29, pp. 801–811, Sept. 1980.