The 17th Annual International Symposium on COMPUTER ARCHITECTURE
1990

A New Approach to Fast Control of $r^2 \times r^2$ 3-Stage Benes Networks of $r \times r$ Crossbar Switches

Abdou Youssef

Department of Elect. Eng. & Comput. Sci.
The George Washington University
Washington, DC 20052

Bruce Arden

College of Engineering and Applied Science
University of Rochester
Rochester, NY 14627

ABSTRACT

The routing control of Benes networks has proven to be costly. This paper introduces a new approach to fast control of $N \times N$ 3-stage Benes networks of $r \times r$ crossbar switches as building blocks, where $N = r^2$ and $r \geq 2$. The new approach consists of setting the leftmost column of switches to an apropriately chosen configuration so that the network becomes self-routed while still able to realize a given family of permutations. This approach requires that, for any given family of permutations, a configuration for the leftmost column be found. Such a family is called compatible and the configuration of the leftmost column is called the compatibility factor. In this paper, compabitibility is characterized and a technique to determine compatibility and the compatibility factor is developed. The technique is used to show the compatibility and find the compatibilty factor of Ω-realizable permutations, the permutations needed to emulate a hypercube, and the families of permutations required by FFT, bitonic sorting, tree computations, multidimensional mesh and torus computations, and multigrid computations. An $O(\log^2 N)$ time routing algorithm for the 3-stage Benes will also be developed. Finally, as only 3 compatibility factors are required by the above families of permutations, it will be proposed to replace the first column by 3 multiplexed connections yielding a self-routing network with strong communication capabilities.

§1. Introduction

Reconfigurable multistage interconnection networks have been the focus of intensive research due to their central role in the design and performance of large parallel processing systems [5], [6], [9], [18], [24]. The effectiveness of these networks depend on the efficiency of their routing control and the permutations they realise. Some of these networks, called banyan multistage networks, have efficient routing control but do not realize all permutations [1], [10], [14], [21], [23]. Benes networks, however, realize all permutations but have inefficient routing control [5], [16]. Controlling Benes networks involves either computing the switch configurations of permutations in advance and storing them, or computing them at run time. The first way is costly in space for large systems, while the second is costly in time as setting the switches to realize a given permutation takes $O(N \log N)$ sequential time [22] and $O(N)$ parallel time [11], where N is the number of input terminals of the network.

Two different approaches have been introduced to bypass this control complexity. The first, due to Nassimi and Sahni [15], consists of using destination addresses in a specified manner, and allows for the realization of a subset of permutations in optimal time. The second, due to Lenfant [13], identifies families of frequently used permutations, and develops a specialized control algorithm for each family to realize the permutations of the family efficiently. The first approach produces optimal control but allows for the realization of only a small fraction of permutations. The second approach is limited to a few families.

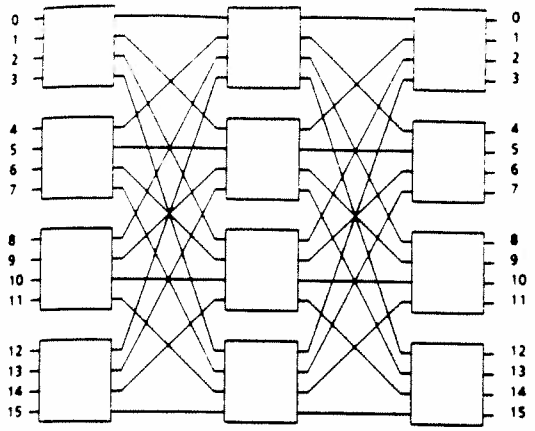
This paper introduces a new approach to controlling 3-stage Benes networks of r^2 input terminals, r^2 output terminals and $r \times r$ crossbar switches, for arbitrary $r \geq 2$ (Fig. 1). These networks can have up to 1024 input/output terminals as the current technology can provide 32×32 crossbar switches. IBM's GF11 [4] is an example of a 3-stage benes-connected parallel system of 24×24 crossbar switches and 576 processors.

The new approach of routing control consists of setting the first stage of the network to a fixed configuration so that the remaining network can be self-routed and can realize a given family of permutations. This approach requires that, for a given family of permutations, the family be examined to determine if there exists a configuration to which the first column of the network can be set so that the remaining network realizes the family (such a family is said to be *compatible*). This approach combines the advantages of the aforementioned two approaches in having optimal control for numerous large families of permutations.

In this paper compatibility is characterized and a technique to determine compatibility is developed. The technique is used to show the compatibility of the families of permutations required by many interesting classes of problems such as FFT, bitonic sorting, tree computations, multidemensional mesh and torus computations, and multigrid computations. Additional useful families of permutations, such as the permutations realizable by the omaga network, are also shown to be compatible.

The rest of the paper is organized as follows. The next section gives some definitions and fundamental concepts. Compatibility is characterized in Section 3. Section 4 shows the compatibility of several families of permutations. Section 5 identifies the families of permutations required by several interesting classes of problems and shows their compatibility. Finally, Section 6 gives an implementation of the new control scheme and draws some conclusions

regarding the possibility of replacing the leftmost column of the 3-stage Benes network with some fixed interconncetions which leads to self-routed networks of powerful communication capibilities.



3-stage Benes B(4,2)

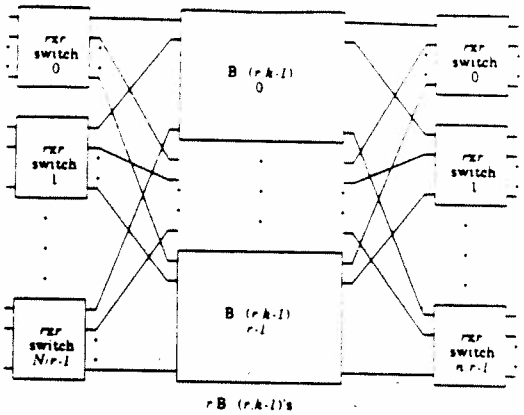
Figure 1

§2. Definitions and Fundamental Concepts

A Benes network [5], denoted here $B(r,k)$, of r^k input terminals, r^k output terminals, and $r \times r$ crossbar permutation switches as building blocks (where $r \geq 2$), is defined recursively as shown in Fig. 2. For $k = 1$, the network is simply the $r \times r$ crossbar switch. For $k \geq 2$, the connectivity between the leftmost column and the middle $B(r,k-1)$ networks is a permutation that maps (i.e., links) the q-th output port of the p-th switch to the p-th input terminal of the q-th $B(r,k-1)$ for $q = 0,1,...,r-1$ and $p = 0,1,...,r^{k-1}-1$. The inverse of this permutation connects the output terminals of the middle $B(r,k-1)$ networks to the input ports of the rightmost column. Note that $B(r,k)$ has $2k-1$ columns of r^{k-1} crossbar switches each. An interesting special case is when $k = 2$. The network $B(r,2)$ has then 3 columns as illustrated in Fig. 1.

The $B(r,2)$ networks for arbitrary r are the focus of this paper and referred to as ***3-stage Benes networks***. Let $N = r^2$ throughout this paper. The columns of $B(r,2)$ are numbered 0, 1, 2 from left to right, and will often be referred to as left, middle and right column, respectively. Each column has r switches numbered $0,1,...,r-1$ from top to bottom. The ports of each column are numbered $0,1,...,N-1$, from top to bottom. The q-th port of the p-th switch in any column is then labeled $pr+q$. The connectivity between the left column and middle column is a permutation f where $f(pr+q) = qr+p$, for $p = 0,1,...,r-1$ and $q = 0,1,...,r-1$. The connectivity between the middle column and the right column is f^{-1} which happens to be equal to f.

Let $R_N = \{0,1,...,N-1\}$, $R_r = \{0,1,...,r-1\}$, and S_N be the set of permutations of R_N. Every number $x \in R_N$ is uniquely expressed as $pr+q$ for some p and q, where $0 \leq p \leq r-1$ and $0 \leq q \leq r-1$. Let H be the subset of permutations that are realizable by any column of switches of $B(r,2)$. Observe that H is a subgroup of the symmetric group S_N, and $h \in H$ if $h(pr+q) = pr+q'$ for some q' function of p and q. Let then $q' = t(p,q) = h(pr+q) \bmod r$. Denote by $t(p,.)$ the mapping from R_r to R_r such that $t(p,.)(q) = t(p,q)$. Clearly, $t(p,.)$ is a permutation of R_r.



The recursive structure of the Benes network

Figure 2

If $h \in H$, a column of $B(r,2)$ is said to be *set to* h if the switches are set so that input port $pr+q$ is connected to output port $h(pr+q)$. Equivalently, for $0 \leq p \leq r-1$, the p-th switch of the column is set in such a way that its q-th input port is connected to its $t(p,q)$-th output port.

A permutation ϕ in S_N is ***realizable*** by $B(r,2)$ if the switches of $B(r,2)$ can be set such that the input terminal i is connected to the output terminal $\phi(i)$, for $i = 0,1,...,N-1$. A permutation is said to be ***h-realizable*** for some h in H if it is realizable by $B(r,2)$ with the left column set to h. A family of m permutations in S_N is said to be ***compatible*** if there is some permutation $h \in H$ such that all the permutations in the family are h-realizable. The permutation h is then called a ***compatibility factor*** of the family.

Every path from an input terminal s to an output terminal d in $B(r,2)$, denoted $s \rightarrow d$, goes through six ports $(x_1,x_2,x_3,x_4,x_5,x_6)$, where $x_1 = s$, x_2 is an output port of the left column, x_3 is an input port of the middle column, and so on. Note that $x_3 = f(x_2)$ and $x_5 = f(x_4)$, and therefore, the path is fully decided by x_2 and x_4. Two paths $(x_1,x_2,x_3,x_4,x_5,x_6)$ and $(y_1,y_2,y_3,y_4,y_5,y_6)$ are said to *conflict* if $x_i = y_i$ for some i.

If the left column is set to some configuration h in H, where $h(pr+q) = pr+t(p,q)$, and if the path $(x_1,x_2,x_3,x_4,x_5,x_6)$ between an input terminal $s = x_1 = pr+q$ and an output terminal $d = x_6 = lr+n$ is realizable with the left column set to h, then $x_2 = h(x_1) = pr+t(p,q)$, $x_3 = f(x_2) = t(p,q)r+p$, $x_4 = t(p,q)r+p'$ for some $p' \in R_r$, and $x_5 = f(x_4) = p'r+t(p,q)$. As x_5

and x_6 are linked to the same switch in the right column, it follows that $p' = l$. Thus we have:

$x_1 = pr + q,\ x_2 = pr + t(p,q),$
$x_3 = t(p,q)r + p,\ x_4 = t(p,q)r + l,$
$x_5 = lr + t(p,q)$, and $x_6 = d = lr + n.$

Therefore, if the left column is set to some configuration h in H, there exists a unique path between every input terminal and every output terminal. That path can be determined using the output terminal address [12] as follows: Each switch in the middle column, when receiving a request to connect to some output terminal $lr + n$, links the incoming request to its l-th output port, and each switch in the right column links the incoming request to its n-th output port. Therefore, $B(r,2)$ becomes self-routed when the left column is set to a configuration h in H. Consequently, if a family of permutations is compatible, where one compatibility factor is some h in H, the left column of the 3-stage Benes network can be set to h and the resulting network can realize all the permutations of the family in a self-routed fashion. This is the primary motivation behind our approach.

The main focus of this paper is to determine if a given family of permutations is compatible, and if so, to find one compatibility factor. Compatibility is characterized and various families of permutations that arise from many important classes of problems are identified and shown compatible, and their compatibility factors are found. These compatibility factors (permutations) can then be stored. When a 3-stage Benes-based computer system executes an algorithm whose communication requirements can be fulfilled by a compatible family of permutations, the left column is first set to the family's compatibility factor (if already found and stored) for the entire duration of the algorithm, allowing then for fast, self-routed realization of the permutations, and leading to speedy execution of the algorithm. Note that the amount of memory required to store the compatibilty factors is very small in comparison with the amount of memory needed to store the setting of the switches for all the $N!$ permutations.

§3. Characterization of Compatibility

In this section, necessary and sufficient conditions for a permutation to be h-realizabale will be given and then a compatibility characterization theorem is concluded which gives necessary and sufficient conditions for a given family of permutations to be compatible. This characterization will be used to show the compatibilty of several interesting families of permutations.

3.1 Lemma. *If the left column is set to some configuration $h \in H$, where $h(pr+q) = pr + t(p,q)$, and if s and s' are two distinct input terminals and d and d' two distinct output terminals, where $s = pr+q$, $s' = p'r+q'$, $d = lr+n$, and $d' = l'r + n'$, then the paths $s \to d$ and $s' \to d'$ conflict if and only if $t(p,q) = t(p',q')$ and $l = l'$.*

Proof. Let the path $(s \to d) = (x_1, x_2, x_3, x_4, x_5, x_6)$ and $(s' \to d') = (y_1, y_2, y_3, y_4, y_5, y_6)$. Since $x_1 \neq y_1$, it follows that $h(x_1) \neq h(y_1)$ and $f(h(x_1)) \neq f(h(y_1))$, and hence $x_2 \neq y_2$ and $x_3 \neq y_3$. We have also $x_6 \neq y_6$. Therefore, the two paths conflict if and only if $x_4 = y_4$ or $x_5 = y_5$. As $x_5 = f(x_4)$ and $y_5 = f(y_4)$, we have $x_4 = y_4$ if and only if $x_5 = y_5$. So the two paths conflict if and only if $x_4 = y_4$. It was shown in the preceding section that $x_4 = t(p,q)r + l$, and $y_4 = t(p',q')r + l'$. Hence, $x_4 = y_4$ if and only if $t(p,q) = t(p',q')$ and $l = l'$. ∎

3.2 Theorem. *Let h be in H where $h(pr+q) = pr + t(p,q)$. Let also ϕ be a permutation in S_N, and $\alpha(p,q) = \lfloor \frac{\phi(pr+q)}{r} \rfloor$. ϕ is h-realizable if and only if for every two distinct input terminals $s = pr+q$ and $s' = p'r+q'$, $\alpha(p,q) = \alpha(p',q')$ implies that $t(p,q) \neq t(p',q')$.*

Proof. Let $s = pr+q$ and $s' = p'r+q'$ be two arbitrary distinct input terminals, and let $\phi(s) = d = lr+n$ and $\phi(s') = d' = l'r+n'$, where $l = \alpha(p,q)$ and $l' = \alpha(p',q')$. Clearly, $d \neq d'$. If ϕ is h-realizable, then the paths $s \to d$ and $s' \to d'$ do not conflict in $B(r,2)$ with the left column set to h. After the preceding lemma, we must have either $t(p,q) \neq t(p',q')$ or $l \neq l'$. Therefore, if $\alpha(p,q) = \alpha(p',q')$, then $t(p,q) \neq t(p',q')$.
Conversely, if for every two distinct input terminals $s = pr+q$ and $s' = p'r+q'$, $\alpha(p,q) = \alpha(p',q')$ implies that $t(p,q) \neq t(p',q')$, then by the preceding lemma, for every two distinct input terminals s and s', the paths $s \to \phi(s)$ and $s' \to \phi(s')$ do not conflict in $B(r,2)$ with the left column set to h. Consequently, ϕ is h-realizable. ∎

3.3 The Compatibility Characterization Theorem. *Let $\{\phi_1, \phi_2, ..., \phi_m\}$ be a family of permutations in S_N, and $\alpha_i(p,q) = \lfloor \frac{\phi_i(pr+q)}{r} \rfloor$, for $i = 1,2,...,m$. The family $\{\phi_1, \phi_2, ..., \phi_m\}$ is compatible if and only if there is a mapping $t: R_r \times R_r \to R_r$ such that:*
(i) $t(p,.)$ is a permutation of R_r for every p in R_r.
(ii) If for some p, p', q, $q' \in R_r$ there exists i such that $\alpha_i(p,q) = \alpha_i(p',q')$ and $p \neq p'$, then $t(p,q) \neq t(p',q')$.

Proof. Assume first that $\phi_1, \phi_2, ..., \phi_m$ are compatible, and that h is a compatibility factor. Let t be such that $h(pr+q) = pr + t(p,q)$. Clearly, $t(p,.)$ is a permutation of R_r for every p. To show (ii), assume that for some p, p', q, q' in R_r there exists i such that $\alpha_i(p,q) = \alpha(p',q')$ and $p \neq p'$. Let $s = pr+q$ and $s' = p'r+q'$. Clearly $s \neq s'$ because $p \neq p'$. Since ϕ_i is h-realizable, $s \neq s'$ and $\alpha_i(p,q) = \alpha_i(p',q')$, it follows that $t(p,q) \neq t(p',q')$, after Theorem 3.2.
Conversely, assume that there is a mapping $t: R_r \times R_r \to R_r$ that satisfies (i) and (ii). Let h be the mapping from R_N to R_N such that $h(pr+q) = pr+t(p,q)$. Since $t(p,.)$ is a permutation of R_r for every p, h must a permutation in H. For every i, ϕ_i will be shown h-realizable using Theorem 3.2. Let $s = pr+q$ and $s' = p'r+q'$ such that $s \neq s'$ and $\alpha_i(p,q) = \alpha_i(p',q')$. Since $s \neq s'$, we have $p \neq p'$ or $q \neq q'$. If $p \neq p'$, it follows from (ii) that $t(p,q) \neq t(p',q')$. If $p = p'$, then $q \neq q'$ and therefore $t(p,q) \neq t(p,q')$ because $t(p,.)$ is a permutation. So in both cases $t(p,q) \neq t(p',q')$. After Theorem 3.2, ϕ_i is h-realizable. ∎

It can be seen from the proof of the previous theorem

that when there is a mapping t that satisfies the conditions (i) and (ii) of the theorem, one compatibility factor of the family is a permutation $h \in H$ where $h(pr+q) = pr+t(p,q)$.

It is worthwhile to note that Theorem 3.3 can be mapped into a graph-theoretic problem, namely, the node coloring problem. Let $\phi_1, \phi_2, ..., \phi_m$ be m permutations in S_N. Let G = (V,E) be the following undirected graph: V = $R_r \times R_r$ and E = $\{((p,q),(p',q'))|p = p'$ *or* $(\exists i)(\alpha_i(p,q) = \alpha_i(p',q'))\}$. The theorem can now be stated as follows: $\phi_1, \phi_2, ..., \phi_m$ are compatible if and only if G can be r-colored such that no two adjacent nodes have the same color. To prove this, let $t(p,q)$ be the color of node (p,q). It is clear that t satisfies condition (i) and (ii) of Theorem 3.3 if and only if t r-colors G in such a way that no two adjacent nodes have the same color.

The general coloring problem is NP-complete, but it remains open whether these graphs have any peculiarities that open the door to a fast (i.e., polynomial) algorithm to r-color them. Such an algorithm would automate deciding compatibility and finding a compatibility factor.

Using this graph formulation, it can be shown that not every family of permutations is compatible. Take for example $r = 2$, (and hence $N = 4$ and $R_r = \{0,1\}$), $\phi_1 = (0\ 1\ 2)(3)$ and $\phi_2 = (0)(1\ 2\ 3)$. It can be shown that $\alpha_1(0,0) = \alpha_1(1,0) = 0$, $\alpha_1(0,1) = \alpha_1(1,1) = 1$, $\alpha_2(0,0) = \alpha_2(1,1) = 0$, and $\alpha_2(0,1) = \alpha_2(1,0) = 1$. The corresponding graph G is depicted in Fig. 3. Since G has a 3-clique, it cannot be 2-colored and, therefore, ϕ_1 and ϕ_2 are not compatible.

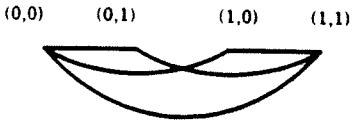


Figure 3

§4. Compatible Families of Permutations

Various families of permutations will be shown compatible in this section. In particular, two large families of permutations, namely, the family of ***pseudo bit translations*** and the ***L-family***, will be defined and shown to be compatible. The permutations realizable by the omega network Ω_N will also be shown compatible. In the next section, the families of permutations required by many application areas will be shown to be subfamilies of these three families and hence compatible. In the remainder of the paper, r is assumed to be a power of two ($r = 2^k$), and therefore, $N = r^2 = 2^{2k}$.

Every number $x = pr + q$ in R_N can be expressed in binary as $x = x_{2k-1}x_{2k-2}...x_0$ and conveniently also as $x = p_{k-1}p_{k-1}...p_0q_{k-1}q_{k-2}...q_0$, where $p = p_{k-1}p_{k-2}...p_0$ and $q = q_{k-1}q_{k-2}...q_0$. The bit positions $0, 1, ..., k-1$ of x (i.e., the k least significant bits) are said to form the ***q-wing*** of x, and the bit positions $k, k+1, ..., 2k-1$ (i.e., the k most significant bits) the ***p-wing*** of x. For $i = 0, 1, ..., k-1$, bits p_i and q_i are called ***siblings***. Also, p_i is the ***p-sibling*** of q_i and q_i the ***q-sibling*** of p_i.

If π is a permutation of $\{0, 1, ..., 2k-1\}$, then f_π denotes a permutation in S_N, called a ***bit permutation***, such that $f_\pi(x_{2k-1}x_{2k-2}...x_0) = x_{\pi(2k-1)}x_{\pi(2k-2)}...x_{\pi(0)}$. A bit permutation f_π manipulates the bits of its parameter x, moving the i-th bit of x to bit position $\pi^{-1}(i)$, for $i = 0, 1, ..., 2k-1$. A ***pseudo bit translation*** is a bit permutation f_π that satisfies the following condition: f_π moves a bit from the q-wing to the p-wing if and only if f_π moves the p-sibling of that bit to the q-wing.

4.1 Theorem. *All pseudo bit translations are compatible and their compatibility factor is h such that $h(pr+q) = pr + (p \oplus q)$, where $p \oplus q$ is the bitwise XOR of p and q.*

Proof. Let $t(p,q) = p \oplus q$. It is enough to show that t satisfies the conditions (i) and (ii) of Theorem 3.3.

(i) $t(p,.)$ is a permutation since $t(p,.)(q) = t(p,.)(q') \Rightarrow t(p,q) = t(p,q') \Rightarrow p \oplus q = p \oplus q' \Rightarrow q = q'$.

(ii) Assume that for some p, p', q, $q' \in R_r$ there is a pseudo bit translation ϕ such that $\alpha_\phi(p,q) = \alpha_\phi(p',q')$ and $p \neq p'$. We need to show that $t(p,q) \neq t(p',q')$.

Let E be the set of bit positions of the q-wing that are moved to the p-wing by ϕ. Since ϕ is a pseudo bit translation, E must also be the set of p-wing bit positions that move to the q-wing.

$\alpha_\phi(p,q) = \alpha_\phi(p',q') \Rightarrow [(\forall i \notin E)(p_i = p'_i)$ and $(\forall i \in E)(q_i = q'_i)] \Rightarrow q \oplus q'$ has 0's in all bit positions $i \in E$.

$p \neq p' \Rightarrow (\exists i_0)(p_{i_0} \neq p'_{i_0}) \Rightarrow p \oplus p'$ has '1' in bit position i_0. Clearly $i_0 \in E$ because $\forall i \notin E$ we have $p_i = p'_i$. Therefore $(p \oplus p') \oplus (q \oplus q')$ has '1' in bit position i_0. Consequently,

$t(p,q) \oplus t(p',q') = (p \oplus q) \oplus (p' \oplus q') = (p \oplus p') \oplus (q \oplus q') \neq 0$.

Hence, $t(p,q) \neq t(p',q')$. ∎

Fig. 4-a shows the setting of the left column for pseudo bit translations.

Next we define the ***L-family*** and show it to be compatible. For every permutation ϕ in S_N, let $\alpha_\phi(p,q) = \lfloor \frac{\phi(pr+q)}{r} \rfloor$ = the leftmost k bits of $\phi(pr+q)$. The ***L-family*** of permutations in S_N is the set

$L = \{\phi \in S_N |$ if $\alpha_\phi(p,q) = \alpha_\phi(p',q')$ and $p \neq p'$, then p and p' agree in all but one bit position, and q and q' agree in at least one bit position$\}$

To show that the L-family is compatible, the set of switches of the left column (i.e., the set R_r of the switch labels) is partitioned into two subsets E_k and F_k, which will be defined recursively such that the binary representations of any two numbers in each subset disagree in at least two bit positions. Let $E_1 = \{0\}$, $F_1 = \{1\}$ and recursively $E_i = 0E_{i-1} \cup 1F_{i-1}$ and $F_i = 0F_{i-1} \cup 1E_{i-1}$, where $aE_{i-1} = \{ax_{i-2}...x_0 |\ x_{i-2}...x_0 \in E_{i-1}\}$ for a=0 or 1, and aF_{i-1} is defined similarly. For example, $0E_1 = \{00\}$, $0F_1 = \{01\}$, $1E_1 = \{10\}$, $1F_1 = \{11\}$, and consequently, $E_2 = \{00, 11\}$ and $F_2 = \{01, 10\}$.

It can be easily shown by induction on $i = 1, 2, ..., k$ that $E_i \cup F_i = \{0, 1, ..., 2^i - 1\}$ (in decimal), $E_i \cap F_i = \emptyset$ and for $i > 1$, any two numbers in each of the sets E_i and F_i disagree in at least two bit positions. In particular, $E_k \cap F_k = \emptyset$ and $E_k \cup F_k = \{0, 1, ..., r-1\} = R_r$

4.2 Theorem. *The L-familiy is compatible and its compatibility factor is h such that* $h(pr+q) = pr+q$ *if* $p \in E_k$ *and* $h(pr+q) = pr+\overline{q}$ *if* $p \in F_k$, *where* $\overline{q}$ *is the bitwise complement of* q.

Proof. Let $t(p,q) = q$ if $p \in E_k$ and $t(p,q) = \overline{q}$ if $p \in F_k$. It will be shown that t satisfies the two conditions (i) and (ii) of Theorem 3.3.

(i) $t(p,.)$ is a permutation because $t(p,q) = t(p,q') \Rightarrow [q = q' \text{ or } \overline{q} = \overline{q'}] \Rightarrow q = q'$.

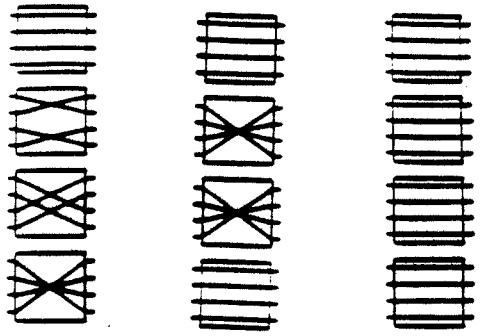
(ii) Assume that for some p, p', q, $q' \in R_r$ there is a permutation $\phi \in L$ such that $\alpha_\phi(p,q) = \alpha_\phi(p',q')$ and $p \neq p'$. By definition of L, it follows that p and p' agree in all but one bit position, and q and q' agree in at least one bit position. Consequently, p and p' cannot both be in the same E_k or F_k. Assume without loss of generality that $p \in E_k$ and $p' \in F_k$. Then $t(p,q) = q$ and $t(p',q') = \overline{q'}$. As q and q' agree in at least one bit position, q and $\overline{q'}$ must disagree in at least one bit position, and therefore, $t(p,q) \neq t(p',q')$. ∎

Fig. 4-b shows the setting of the left column for the L-family.

In addition to the above two families, two more families will be shown compatible, namely, the set H and the set of permutations realizable by the omega network Ω_N [10].

4.3 Theorem. *For every* h *and* $g \in H$, g *is* h*-realizable. In particular,* H *is compatible and its compatibilty factor can be any arbitrary permutation* $h \in H$.

Proof. Let h and g be two permutations in H, and let I be the identity permutation which is also in H. As H is a subgroup of S_N, $h^{-1}g \in H$. Set the left column of $B(r,2)$ to h, the middle column to I and the right column to $h^{-1}g$. This setting realizes the permutation $hfIfh^{-1}g$ which is equal to g (recall that $f = f^{-1}$ and hence $fIf = ff = I$). It follows that $B(r,2)$ can realize g with its left column set to h. ∎



for pseudo bit translations (a) | for the L-family (b) | for the omaga network (c)

The configurations of the left column for various families

Figure 4

4.4 Theorem. *The permutations realizable by the omega network* Ω_N *of* N *input terminals,* N *output terminals, and* 2×2 *crossbar switches as building blocks, are compatible and their compatibility factor is the identity permutation.*

Proof. Let $t(p,q) = q$. It will be shown that t satisfies conditions (i) and (ii) of Theorem 3.3.

(i) Since $t(p,.)(q) = t(p,q) = q$, $t(p,.)$ is the identity permutation of R_r.

(ii) Assume that for some p, p', q, and q' in H there is a permutation ϕ realizable by Ω_N such that $\alpha_\phi(p,q) = \alpha_\phi(p',q')$ and $p \neq p'$. It will be proved next that $t(p,q) \neq t(p',q')$. It was shown in [10] that a permutation ψ is realizable by Ω_N if and only if $(\forall s = s_{2k-1}s_{2k-2}...s_0 \in R_N)(\forall s' = s'_{2k-1}s'_{2k-2}...s'_0 \in R_N)(\forall l < \log N = 2k-1)$ (if $s \neq s'$ and $d_{2k-1}d_{2k-2}...d_{l+1} = d'_{2k-1}d'_{2k-2}...d'_{l+1}$, then $s_l s_{l-1}...s_0 \neq s'_l s'_{l-1}...s'_0$), where $\psi(s) = d_{2k-1}d_{2k-2}...d_0$ and $\psi(s') = d'_{2k-1}d'_{2k-2}...d'_0$. This will be used to show that $t(p,q) \neq t(p',q')$.

Let $s = pr+q$, $s' = p'r+q'$ and $l = k-1$. Thus $p = s_{2k-1}...s_k$, $q = s_{k-1}...s_0$, $p' = s'_{2k-1}...s'_k$ and $q' = s'_{k-1}...s'_0$. Let also $\phi(s) = d_{2k-1}d_{2k-2}...d_0$ and $\phi(s') = d'_{2k-1}d'_{2k-2}...d'_0$. Then $\alpha_\phi(p,q) = d_{2k-1}d_{2k-2}...d_k$ and $\alpha_\phi(p',q') = d'_{2k-1}d'_{2k-2}...d'_k$.

$\alpha_\phi(p,q) = \alpha_\phi(p',q') \Rightarrow d_{2k-1}d_{2k-2}...d_k = d'_{2k-1}d'_{2k-2}...d'_k$.

$p \neq p' \Rightarrow s \neq s'$. As ϕ is realizable by Ω_N, it follows that $s_{k-1}s_{k-2}...s_0 \neq s'_{k-1}s'_{k-2}...s'_0$ which implies that $q \neq q'$, that is $t(p,q) \neq t(p',q')$.

Therefore, by Theorem 3.3, all the permutations realizable by Ω_N are h-realizable, where $h(pr+q) = pr+t(p,q) = pr+q$, that is, the identity permutation. ∎

Fig. 4-c shows the setting of the left column for the Ω_N-realizable permutations.

§5. Applications

In this section the families of permutations of several problems are identified and shown to be subfamilies of the families in the last section, and hence compatible.

5.1 The Fast Fourier Transform

To compute FFT in the way described in [19], two permutations are needed: The shuffle (S) and the exchange (E), where $S(x_{2k-1}x_{2k-2}...x_0) = x_{2k-2}...x_0x_{2k-1}$ and $E(x_{2k-1}x_{2k-2}...x_0) = x_{2k-1}x_{2k-2}...x_1\overline{x_0}$. However, at the end of the computation, the components of the resulting vector are in bit reversed order. To restore the order, the bit reversal permutation (ρ) is needed, where $\rho(x_{2k-1}x_{2k-2}...x_0) = x_0x_1...x_{2k-2}x_{2k-1}$. Thus, the overall family of permutations needed by FFT is $\{S, E, \rho\}$.

5.1 Theorem. *The permutations needed by FFT are compatible.*

Proof. As $S(p_{k-1}p_{k-2}...p_0q_{k-1}...q_0) = p_{k-2}...p_0q_{k-1}...q_0p_{k-1}$, it follows that S moves only one bit from the q-wing to the p-wing, namely, bit q_{k-1}, and only one bit from the p-wing to the q-wing, namely, p_{k-1}. As these two bits are siblings, S is a pseudo bit translation. The bit reversal ρ moves every bit of the q-wing to the p-wing and every bit of

the p-wing to the q-wing. Therefore, ρ is trivially a pseudo bit translation. After Theorem 4.1, S and ρ are h-realizable where $h(pr+q) = pr + p \oplus q$. Since $E \in H$, E must be h-realizable for the same h, after Theorem 4.3. ∎

5.2 Bitonic Sorting

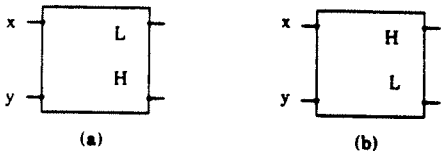
Although parallel sorting algorithms of $O(\log N)$ time have been found [2], they are not practical due to the extremely large multiplicative constant factor of $\log N$. Bitonic Sorting, though of time complexity $O(\log^2 N)$, is a practical parallel algorithm [19].

As shown in [19], N numbers can be sorted using a sorting network of $\frac{\log_2 N(\log_2 N+1)}{2}$ stages of comparison switches and based on Batcher's bitonic sorter [3]. Simulating this sorting network on 3-stage Benes networks involves realizing the interconnections (i.e., permutations) between the columns of the sorting network, as well as the columns of switches themselves.

Bitonic sorting and the sorting network based on the bitonic sorter are briefly reviewed next, and the permutations required to simulate the sorting network on $B(r,2)$ are identified and shown to be compatible.

A sequence of real numbers $a_0, a_1, ...a_{N-1}$ is *bitonic* if (1) there exists i such that $\{a_0, a_1, ..., a_i\}$ is increasing, and $\{a_{i+1}, ..., a_{N-1}\}$ is decreasing; or if
(2) the sequence can be shifted cyclically so that (1) is satisfied.

An $N \times N$ bitonic sorter is a recursive network where, for $N = 2$, it is a 2×2 comparison switch that takes two input numbers and puts the smaller in the upper output port and the larger in the lower output port (as in Fig. 5-a), or vice versa (as in Fig. 5-b), according to a control bit. For larger N, it is as depicted in Fig. 6. It is shown in [3] and [19] that if the input is a bitonic sequence, then this network sorts the input in increasing order. If the switches of the network of Fig. 6 are replaced by switches of the type in Fig. 5-b, the network sorts the bitonic input in decreasing order.



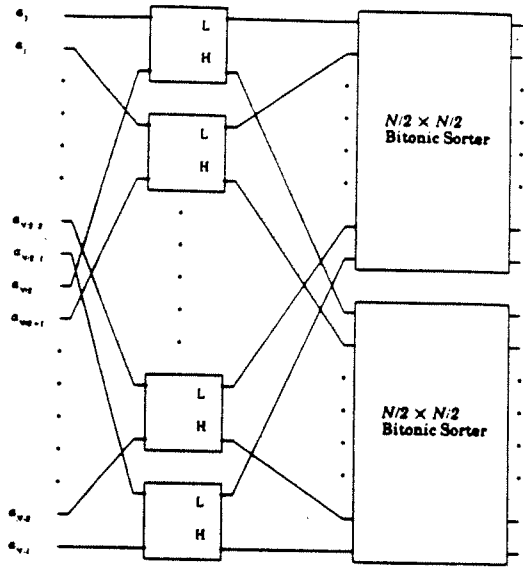
Bitonic switch, also a 2×2 bitonic sorter

Figure 5

A full sorting network that sorts any sequence of $N = 2^m$ numbers can be built in m steps, where the i-th step is a column of $\frac{N}{2^i}$ $2^i \times 2^i$ bitonic sorters as shown in Fig. 7. The network works as follows. The first step sorts pairs of numbers into alternately increasing and decreasing pairs so that each sequence of 4 numbers is bitonic. The second step sorts these bitonic sequences into alternately increasing and decreasing sequences so that each sequence of 8 numbers is bitonic. And so on to the last step which receives a bitonic sequence of length N. Since the last step is an $N \times N$ bitonic sorter, it can sort the incoming sequence. Fig. 8 shows an 8×8 sorting network where the shaded switches place the larger item on the top output, and the blank switches place the smaller on top.

The operations of the sorting network can be viewed as a sequence of data permutations. First the items are permuted by the first column of comparison switches, second they are permuted by the interconnection between the first column and the second column, and then permuted by the second column of switches, and so on. Therefore, the simulation of the sorting network on $B(r,2)$ is done by executing the above sequence of permutations on $B(r,2)$ in order, assuming that the input and output terminals of $B(r,2)$ are N processing elements $pe_0, pe_1, ..., pe_{N-1}$.

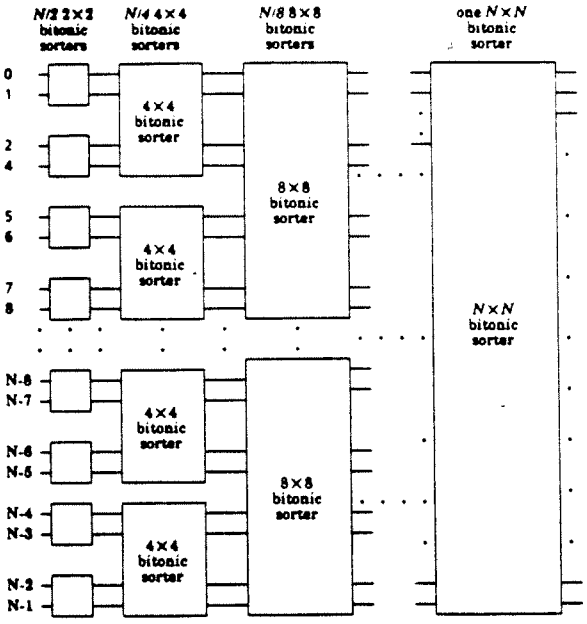
Note that when simulating a column of comparison switches (column j, say), more than permuting is required. Assume the numbers coming to this column are $b_0, b_1, ...b_{N-1}$ from top to bottom. Then, in our simulation, these numbers are in $pe_0, pe_1, ...pe_{N-1}$, respectively, when column j is due to be simulated. At comparison switch i of column j, the incoming numbers are b_{2i} and b_{2i+1}. Hence, in simulation, these numbers are in pe_{2i} and pe_{2i+1}, respectively. To be able to do comparison in the simulation, each of pe_{2i} and pe_{2i+1} must have both b_{2i} and b_{2i+1}, for $i = 0, 1, ..., N-1$. This can be accomplished by first executing the exchange permutation on the numbers, and then if switch i of the sorting network is in state (a) (as in Fig. 5-a), pe_{2i} keeps $\min(b_{2i}, b_{2i+1})$, while pe_{2i+1} keeps



The structure of the bitonic sorter

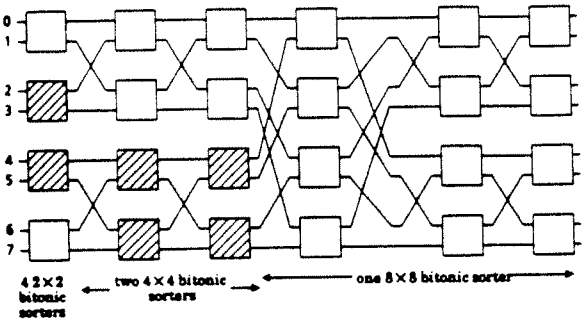
Figure 6

$\max(b_{2i}, b_{2i+1})$. If the comparison switch is in state (b), the opposite is done. Note that the states of the comparison switches of the sorting network can be known *a priori*.



The general structure of the $N \times N$ sorting network based on bitonic sorters

Figure 7



An 8×8 sorting network

Figure 8

The interconnection permutations between the successive columns of the sorting network are identified in the following lemma.

5.2 Lemma. *The inter-column interconnections of the sorting network form the set*

$$\{S_i | 0 \le i \le m-2\} \cup \{U_i S_{i+1} | 0 \le i \le m-2\}$$

where

$$S_i(x_{m-1}...x_0) = x_{m-1}...x_{m-i}x_{m-i-2}...x_0x_{m-i-1}$$

and

$$U_i(x_{m-1}...x_1x_0) = x_{m-1}...x_{m-i}x_0x_{m-i-1}...x_1.$$

Proof. Observe first that S_0 is the perfect shuffle on $\{0, 1, .. ., 2^m - 1\}$, S_1 is the perfect shuffle within two segments which are $\{0, 1, ..., 2^{m-1} - 1\}$ and $\{2^{m-1}, 2^{m-1} + 1, ..., 2^m - 1\}$, and in general, S_i is the perfect shuffle within 2^i contiguous segments of length 2^{m-i} each. Similarly, U_0 is the perfect unshuffle (i.e, the inverse of S_0) on $\{0, 1, ..., 2^m - 1\}$, and U_i is the perfect unshuffle (i.e., $U_i = S_i^{-1}$) within the same 2^i contiguous segments which S_i shuffles. A close inspection of the bitonic sorter in Fig. 6 shows that the leftmost interconnection is the perfect shuffle S_0, and the interconnection between the first column and the other $2\ \frac{N}{2} \times \frac{N}{2}$ bitonic sortes is the unshuffle U_0. The leftmost interconnection of these $2\ \frac{N}{2} \times \frac{N}{2}$ bitonic sorters must then be S_1. It follows then that the interconnection between the first two columns is U_0S_1. It can be easily shown by induction that the remaining interconnections are $U_1S_2, U_2S_3, ...U_{m-2}S_{m-1}$, due to the recursive structure of the bitonic sorter.

As described earlier, the sorting network (Fig. 7) has at the i-th block (from the left) $\frac{N}{2^i}$ $2^i \times 2^i$ bitonic sorters, whose interconnections are then
$S_{m-i}, U_{m-i}S_{m-i+1}, U_{m-i+1}S_{m-i+2}, ..., U_{m-2}S_{m-1}$,
for $i = 2, 3, ..., m$. Note that the first block is just the first column of comparison switches and has no interconnections. It follows that the interconnections of the sorting network are
$\cup_{2 \le i \le m} \{S_{m-i}, U_{m-i}S_{m-i+1}, U_{m-i+1}S_{m-i+2}, ...U_{m-2}S_{m-1}\}$
which is equal to $\{S_i | 0 \le i \le m-2\} \cup \{U_iS_{i+1} | 0 \le i \le m-2\}$. ∎

5.3 Theorem. *The permutations required by bitonic sorting are compatible.*

Proof. Note that $m = 2k$ and that for $i \ge k$, $S_i(x)$ does not alter the k most significant bits of x, and therefore, S_i is in the set H. Note also that

$$(x_{m-1}...x_0)U_iS_{i+1} = S_{i+1}(x_{m-1}...x_{m-i}x_0x_{m-i-1}...x_1)$$
$$= x_{m-1}...x_{m-i}x_0x_{m-i-2}...x_1x_{m-i-1}$$

and therefore U_iS_{i+1} does not alter the k most significant bits if $i \ge k$. Hence, for $i \ge k$, $U_iS_{i+1} \in H$. The exchange permutation E needed to simulate the columns of the sorting network is also in H. After Theorem 4.3, H is compatible and its compatibility factor is arbitrary. Consequently, it suffices to show that the remaining permutations $S_0, S_1, ..., S_{k-1}, U_0S_1, U_1S_2, ..., U_{k-1}S_k$ are compatible. In fact, they will be shown to be in the L-family.

For all $i < k$, $S_i(p_{k-1}...p_0q_{k-1}...q_0) =$
$p_{k-1}...p_{k-i}p_{k-i-2}...p_0q_{k-1}...q_0p_{k-i-1}$. Let $\alpha_i(p, q)$ be the leftmost bits of $S_i(p_{k-1}...p_0q_{k-1}...q_0)$, that is, $\alpha_i(p, q) = p_{k-1}...p_{k-i}p_{k-i-2}...p_0q_{k-1}$.

$[\alpha_i(p, q) = \alpha_i(p', q')$ and $p \ne p'] \Rightarrow [p$ and p' agree in all but bit position $k-i-1$, and q and q' agree in at least bit position $k-1] \Rightarrow S_i \in L$.

Similarly, for $i < k$, we have $(p_{k-1}...p_0q_{k-1}...q_0)U_iS_{i+1} = p_{k-1}...p_{k-i}q_0p_{k-i-2}...p_0q_{k-1}...q_1p_{k-i-1}$.

Let $\alpha_i(p,q)$ be the k leftmost bits of

$$(p_{k-1}\cdots p_0 q_{k-1}\cdots q_0)U_i S_{i+1},$$

that is, $\alpha_i(p,q)=p_{k-1}\cdots p_{k-i}q_0 p_{k-i-2}\cdots p_0$.
$[\alpha_i(p,q) = \alpha_i(p',q')$ and $p \neq p'] \Rightarrow [p$ and p' agree in all but bit position $k-i-1$, and q and q' agree in at least bit position $0] \Rightarrow U_i S_{i+1} \in L$.
Therefore, all the permutations required by bitonic sorting are compatible and their compatibility factor is h of Theorem 4.2. ∎

5.3 Tree Computations

Many parallel computations require full binary tree structures. These include semigroup computations such as addition and multiplication of N numbers, finding the maximum or minimum of N numbers, logical *and* and logical *or* operations on N boolean values, and any other associative operators.

If these algorithms are to run on 3-stage Benes systems, then the tree communication structure has to be emulated by permutations as explained next. Assume a full binary tree of $N-1$ nodes, where $N = 2^{2k}$. The nodes are labeled by level in a standard way where the root is labeled 1 and every internal node i has node $2i$ as its left child and node $2i+1$ as its right child. In the top-down tree communication, each node may send data to its chidren. In the bottom-up communication, each node may send data to its parent. The top-down communication can be accomplished in two steps, where each step can be carried out on $B(r,2)$ by a permutation. In the first step the parents send data to their left children, and in the second step the parents send data to the right children. As the shuffle permutation S maps i to $2i$ for all $i \leq \frac{N}{2}-1$, S can then carry out the first step. In the second step, where every node $i \leq \frac{N}{2}-1$ may send to node $2i+1 = E(2i)$, the permutation SE, which is the composition of S and the exchange E, can carry out the communication because $(i)SE = E(S(i)) = E(2i) = 2i+1$.

Similarly, the bottom-up communication can be accomplished by two steps, where in the first step the left children send data to their parents, and in the second the right children send data to the parents. The first step can be carried out by the unshuffle permutation U, where $U(x_{2k-1}x_{2k-2}\ldots x_1 x_0) = x_0 x_{2k-1}x_{2k-2}\ldots x_1$ (in particular, $U(2i) = i$ for $i \leq \frac{N}{2}-1$). The second step where $2i+1$ has to map to $i = U(2i) = U(E(2i+1) = (2i+1)EU$ can be done by the composition EU.

Therefore, the permutations required are

$$\{S, SE, U, EU\},$$

which will be shown to be in the L-family and hence compatible.

5.4 Theorem. *The permutations required by tree computations are compatible.*

Proof. From the definitions of S, SE, U and EU, it can be easily seen that $\alpha_S(p,q) = \alpha_{SE}(p,q) = p_{k-2}p_{k-3}\cdots p_0 q_{k-1}$, $\alpha_U(p,q) = q_0 p_{k-1}p_{k-2}\cdots p_1$, and $\alpha_{EU}(p,q) = \overline{q_0}p_{k-1}p_{k-2}\cdots p_1$, where $p = p_{k-1}p_{k-2}\cdots p_0$ and $q = q_{k-1}q_{k-2}\cdots q_0$.

$\alpha_S(p,q) = \alpha_S(p',q')$ clearly implies that
$p_{k-2}p_{k-3}\cdots p_0 q_{k-1} = p'_{k-2}p'_{k-3}\cdots p'_0 q'_{k-1}$
and, therefore, p and p' agree in the right $k-1$ bit positions, and q and q' agree in bit position $k-1$. If in addition $p \neq p'$, then p and p' must agree in all but one position because they can disagree in only bit position $k-1$. Therefore, S is in L. As $\alpha_S(p,q) = \alpha_{SE}(p,q)$, it follows that SE is in L too.
Similarly, $\alpha_U(p,q) = \alpha_U(p',q')$ clearly implies that
$q_0 p_{k-1}p_{k-2}\cdots p_1 = q'_0 p'_{k-1}p'_{k-2}\cdots p'_1$
and, therefore, p and p' agree in the left $k-1$ bit positions, and q and q' agree in bit position 0. If in addition $p \neq p'$, then p and p' must agree in all but one position. Therefore, U is in L.
A similar proof would show that EU is in L. It follows then that all the four permutations are compatible after Theorem 4.2. ∎

5.4 Multidimensional Torus and Mesh Computations

Toruses and meshes are very useful structures in image processing and scientific computing. The permutations required to emulate these structures on $B(r,2)$ will be shown to be realizable by the omega network Ω_N and therefore compatible.

An n-dimensional $l_1 \times l_2 \times \ldots \times l_n$ mesh (resp. torus) is a graph where the set of nodes is $R_{l_1} \times R_{l_2} \times \ldots \times R_{l_n}$ (recall that $R_z = \{0,1,\ldots,z-1\}$) and any two nodes $(x_1,x_2,\ldots,x_n)$ and $(x'_1,x'_2,\ldots,x'_n)$ form an edge if and only if there exists some j such that $(\forall i \neq j)(x_i = x'_i)$ and $x_j = x'_j + 1$ or $x_j = x'_j - 1$ (in the torus case, $+$ and $-$ are modulo l_j). Note that the only difference between meshes and toruses is that toruses have "wrap around" edges.

The communication in toruses and meshes is usually done along one dimension at a time, and also in the same direction. That is, in a communication step there exist some j and $a = 1$ or -1 such that every node $(x_1,x_2,\ldots,x_{j-1},x_j,x_{j+1},\ldots,x_n)$ may send data to node $(x_1,x_2,\ldots,x_{j-1},x_j + a,x_{j+1},\ldots,x_n)$, where in the case of torus the addition in the j-th dimension is modulo l_j. Therefore, a communication step in a mesh or torus can be carried out by a permutation $f_j^{(a)}$ such that $f_j^{(a)}(x_1,x_2,\ldots,x_n) = (x_1,x_2,\ldots,x_{j-1},(x_j + a) \bmod l_j, x_{j+1},\ldots,x_n)$, where $j = 1,2,\ldots,n$ and $a = 1$ or -1.

These permutations will be shown to be realizable by Ω_N when each l_j is a power of two for all $j = 1,2,\ldots,n$, and the number of nodes $l_1 \times l_2 \times \ldots \times l_n = N = 2^{2k}$. To do so we need the following two lemmas which are presented without proof because the first is straightforward and the second follows from the first. They will be used in the following subsection also.

5.5 Lemma. *Let m be a positive integer and $a = 2^i$ or -2^i, where $0 \leq i \leq m-1$. Let also x and x' be two m-bit binary numbers. If $(x+a) \bmod 2^m$ and $(x'+a) \bmod 2^m$ agree in the l most significant bits, then x and x' agree in the l most significant bits.*

5.6 Lemma. *Let $m_1, m_2, ..., m_n$ be n positive integers, j an integer in $\{1,2,...,n\}$ and $a = 2^i$ or -2^i, where $0 \leq i \leq m_j - 1$. Let also $x = (x_1, x_2, ..., x_n)$ and $x' = (x'_1, x'_2, ..., x'_n)$ where $(\forall c = 1,2,...,n)(x_c$ and x'_c are m_c-bit binary numbers). View the binary representation of x (resp. x') as the concatenation of the binary representations of $x_1, x_2, ..., x_n$ (resp., $x'_1, x'_2, ..., x'_n$). If $(x_1, x_2, ..., x_{j-1}, (x_j + a) \bmod 2^{m_j}, x_{j+1}, ..., x_n)$ and $(x'_1, x'_2, ..., x'_{j-1}, (x'_j + a) \bmod 2^{m_j}, x'_{j+1}, ..., x'_n)$ agree in the most significant l bits, then x and x' agree in the most significant l bits.*

5.7 Theorem. *The permutations of multidimensional toruses and meshes of 2^{2k} nodes are compatible.*

Proof. Assume the torus (or mesh) is an n-dimensional $2^{m_1} \times 2^{m_2} \times ... \times 2^{m_n}$ torus (or mesh), where $m_1 + m_2 + ...m_n = 2k$. It will be shown that $(\forall j = 1,2,...,n)(\forall a = -1,1)(f_j^{(a)}$ is realizable by $\Omega_N)$. Using the characterization in [10] of permutations realizable by Ω_N, it is enough to show that for all j and a

$(\forall s, s' \in R_N)(\forall l = 1,2...,2k-1)($ if $s \neq s'$ and $d_{2k-1}...d_{l+1} = d'_{2k-1}...d'_{l+1}$, then $s_l s_{l-1}...s_0 \neq s'_l s'_{l-1}...s'_0)$

where

$s = s_{2k-1}...s_0 = (x_1, x_2, ..., x_n)$,

$s' = s'_{2k-1}...s'_0 = (x'_1, x'_2, ..., x'_n)$,

$f_j^{(a)}(s) = d_{2k-1}...d_0$

$= (x_1, ..., x_{j-1}, (x_j + a) \bmod 2^{m_j}, x_{j+1}, ..., x_n)$

$f_j^{(a)}(s') = d'_{2k-1}...d'_0$

$= (x'_1, ..., x'_{j-1}, (x'_j + a) \bmod 2^{m_j}, x'_{j+1}, ..., x'_n)$.

After Lemma 5.6, $d_{2k-1}...d_{l+1} = d'_{2k-1}...d'_{l+1}$ yields that $s_{2k-1}...s_{l+1} = s'_{2k-1}...s'_{l+1}$, which in turn implies that $s_l s_{l-1}...s_0 \neq s_l s_{l-1}...s_0$ because $s \neq s'$. ∎

5.5 Hypercube Computations

Hypercubes are a special case of multidimesional toruses. Specifically, a hypercube of dimension n is the n-dimensional $2 \times 2 \times ... \times 2$ torus. In particular, the node labels $(x_1, x_2, ..., x_n)$'s are binary and $(x_j + 1) \bmod 2 = (x_j - 1) \bmod 2 = \overline{x_j}$. Consequently, $f_j^{(a)}(x_1, x_2, ..., x_n) = (x_1, ...x_{j-1}, \overline{x_j}, x_{j+1}, ...x_n)$, whether $a = 1$ or -1.

5.8 Theorem. *The permutations of the hypercube of dimension $2k$ are compatible.*

Proof. Since the hypercube is a torus, the theorem follows from Theorem 5.7. ∎

5.6 Multigrid Computations

A grid is a two-dimensional mesh. Multigrid computations are common in image processing [20] and scientific computing [7], [8]. The communications in these computations are between nodes that differ in only one dimension by 2^i for $i = 0, 1, 2, ...$ In terms of permutations, the permutations required by a $2^k \times 2^k$ multigrid computations are: $f_{i,1}^{(1)}(x_1, x_2) = (x_1 + 2^i, x_2)$, $f_{i,1}^{(-1)}(x_1, x_2) = (x_1 - 2^i, x_2)$, $f_{i,2}^{(1)}(x_1, x_2) = (x_1, x_2 + 2^i)$, and $f_{i,2}^{(-1)}(x_1, x_2) = (x_1, x_2 - 2^i)$, for $i = 0, 1, 2, ..., k-1$, where $+$ and $-$ are modulo 2^k.

5.9 Theorem. *The permutations of $2^k \times 2^k$ multigrid computations are compatible.*

Proof. The proof can be easily carried out using Lemma 5.6 and following the same line of reasoning as in Theorem 5.7. ∎

§6. Conclusions

A new approach to controlling 3-stage Benes networks has been developed. It consists of finding a configration to which the leftmost column can be set so that a given family of permutations can be realized in a self-routed fashion, leading to fast communication and speedy execution of algorithms. The speedup is high when the compatible family of permutations is large. Compatibility, that is, the existence of an appropriate configuration for the leftmost column, was characterized and a technique to show compatibility was derived from the characterization theorem. Various interesting families of permutations were shown compatible and an appropriate configuration of the leftmost column was found for each family. The unsolved case is how to proceed when the permutations cannot be functionally defined (as in FFT and bitonic sorting) but are irregular as in sparse linear systems. More generally, the problem of deciding compatibility in polynomial time remains open.

The implementation of the new approach of routing control is straightforward and can be integrated into the instruction set of the system. A one-bit flag is needed. For each known compatible family, its compatibilty factor h is stored in memory. Before a certain algorithm starts to run, if the family of permutations required by the algorithm is compatible, the compatibility factor is loaded to the leftmost column of the network and the flag is set to 1. Otherwise, the flag is set to 0. If the system has already an instruction REALIZE-PERM that takes a permutation (or a pointer to it) as operand and sets the network to it, then when the flag is set to 0, the same execution takes place; otherwise, the permutation is realized in the self-routed mode using destination addresses. This implementation shows the utility of the new control scheme when the algorithms require compatible permutations.

Among the families that were shown compatible was the family of permutations required by bitonic sorting. One consequence to this is a new $O(\log^2 N)$ routing control algorithm for 3-stage Benes networks: As sorting destination addresses brings the sources to their destinations, a permutation can be realized on the 3-stage Benes in as many passes as needed by bitonic sorting, that is $\frac{\log N(\log N+1)}{2} = O(\log^2 N)$. Another consequence is that the leftmost column of the 3-stage Benes network can be replaced by the interconnection (i.e., configuration) that is the compatibility factor of the permutations of bitonic sorting. The resulting network is cheaper and self-routed, and realizes any permutation in $O(\log^2 N)$ passes. It also realizes the permutations of bitonic sorting and tree computations in a single pass.

Along the same lines, the leftmost column can be replaced by the three compatibility factors (i.e., interconnections) that have been identified as needed by many interesting problem areas. Some multiplexers can be added to

choose one of the three interconnections as required. Then every one of the computation areas discussed in this paper and shown to need a compatible family of permutations can be run efficiently on the resulting self-routed network.

§7. References

[1] D. P. Agrawal and J. -S. Leu, "Dynamic accessibility testing and path length optimization of multistage interconnection networks," *IEEE Trans. Comput.*, C-34, pp. 255–266, Mar. 1985.

[2] M. Ajtai, J. Kanlos and E. Szemeredi, "Sorting in clog n parallel steps," *Combinatorica* 3, pp. 1-19, 1983.

[3] K. E. Batcher, "Sorting networks and their applications," *1968 Spring Joint Comput. Conf., AFIPS Conf.* Vol. 32, Washington, D.C.: Thompson, 1968, pp. 307–314.

[4] J. Beetem, M. Denneau, and D. Weingarten, "The GF11 Supercomputer," *The 12th ann. Int'l Symp. on Comp. Arch.*, 1985, pp. 108–113.

[5] V. E. Benes, *Mathematical theory on connecting networks and telephone traffic*, Academic Press, New York, 1965.

[6] L. N. Bhuyan and D. P. Agrawal, "Design and performance of generalized interconnection networks," *IEEE Trans. Comput.*, pp. 1081–1090, Dec. 1983.

[7] A. Brandt, "Multigrid Solvers on parallel computers," in *Elliptic Problem Solvers* (M. Schultz, ed.), New York, pp. 39–83, 1981.

[8] T. F. Chan and Y. Saad, "Multigrid algorithms on the Hypercube multiprocessor," *IEEE Trans. Comput.*, vol. C-35, pp. 969–977, Nov. 1986.

[9] T. Feng, "A survey of interconnection networks," *Computer*, Vol. 14, pp. 12–27, Dec. 1981.

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

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

[12] K. Y. Lee, "A Nerw Benes Network Control Algorithm," *IEEE Trans. Comput.*, C-36, pp. 768–772, May 1987.

[13] J. Lenfant, "Parallel permutations of data: A Benes network control algorithm for frequently used permutations," *IEEE Trans. Comput.*, C-27, pp. 637–647, July 1978.

[14] G. F. Lev, N. Pippenger and L. G. Valiant, "A fast parallel algorithm for routing in permutation networks," *IEEE Trans. Comput.*, C-,30 pp. 93–100, Feb. 1981.

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

[16] D. C. Opferman and N. T. Tsao-Wu, "On a class of rearrangeable switching networks, Parts I and II," *Bell Syst. Tech. J.*, pp. 1579–1618, May-June 1971.

[17] M. C. Pease, "The indirect binary n-cube multiprocessor array," *IEEE Trans. Comput.*, C-26, pp. 458–473, May 1976.

[18] H. J. Siegel and S. Smith, "Study of multistage interconnection networks," *Proc. Fifth Annual Symp. Comp. Arch.*, pp. 223–229, Apr. 1978.

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

[20] Q. F. Stout, "Hypercubes and Pyramids," in *Pyramidal Systems for Computer Vision*, edited by V. Cantoni and S. Levialdi, Springer-Verlag, Berlin, 1986.

[21] T. H. Szymanski and V. C. Hamacher, "On the permutation Capibility of multistage interconnection networks," *IEEE Trans. Comput.*, C-36, pp. 810–822, July 1987.

[22] A. Waksman, "A permutation network," *JACM*, Vol. 15, No. 1 pp. 159–163, Jan 1968.

[23] C. Wu and T. Feng, "On a class of multistage interconnection networks," *IEEE Trans. Comput.*, C-29, pp. 694–702, Aug. 1980.

[24] A. Youssef, *Properties of multistage interconnection networks*, Ph.D. dissertation, Princeton University, Feb. 1988.