計算機アーキテクチャ 64－5
（1987. 1. 27）

再構成可能な結合網を用いた大規模システムのための高速制御方式

イサム A. ハミド　　白鳥 則郎　　野口 正一

東北大学電気通信研究所

〒980 仙台市片平2-1-1

IN(Interconnection network)は、大規模システムの構成要素である多数のプロセッサ間における同時通信を可能とし、大規模並列システムの核と考えられており、システムの性能に大きな影響を及ぼす。ここでは、多くの長所をもつ再配置可能なベンズ(Benes)INの制御法について検討する。このようなINの制御法は非常に複雑でかつ実現性に乏しい。そこで、本論文では、並列処理でよく使用される操作に注目し、ベンズINのセッティングを簡単で効率的に行う制御アルゴリズムを提案する。さらに、この制御法の実現が容易なことを具体的な例題を用いて示す。

A new fast control mechanism for supersystems using a rearrengeable class of interconnection network

Issam A. HAMID,　Norio SHIRATORI　and　Shoichi NOGUCHI

Research Institute of Electrical Communication,

Tohoku University, 1-1-2, Katahira, Sendai. 〒980.

The Interconnection Networks(IN's), are capable of allowing simultaneous communication between thousands of processing elements(PE's) and hence, are considered the heart of the parallel and supersystems. But if there are many types of manipulations, it becomes necessary to have a rearrengeable IN like the Benes IN so that we can increase the number of stages to 2n-1, where $N=2^n$, and N is the number of input terminals which is base 2. But the control algorithm for this kind of IN is very complex. In this paper we have studied the mapping capabilities of Benes IN to use into super system and we give a control algorithms for setting this network for a groups of manipulation functions proved to be efficiently used for parallel computation, we have also shown the implementation of these control algorithm to be used by the system programmers with examples.

1-Introduction

The progressing in research on VLSI and VHSIC technologies is making it feasible to consider the construction of complex parallel architectures that comprise a number of processors communicating by means of a high-bandwidth Interconnection Network (IN). Recently, there have been extensive work on the capabilities of constructing architecture consisted from thousands of VLSI chips[Fe81][Ha86].

Several modes of operation (i.e., SIMD, MIMD, etc.), may be desirable in such architectures, like for example PASM[Si81]. Furthermore, it is widdely recognized that the communication complexity of parallel algorithms may be the principle determinant of performance.

Reconfiguration involves establishing different partitions and altering the interconnection between resources within a partition. The objective is to perform the reconfiguration in such away that takes the least total processing time. The problem of configuring architecture is that of the establishment of non-over lapping trees; that is the setting up of a system where instructive trees and data trees can exists simultaneously without interfering with each other.

The network is set up according the computation structures, so that certain topologies(such as ring, binary tree, near neighbour meshes, and so on) are created among sets of nodes,

Powerful pipeline computers are very expensive, because of the desired speed of components. In a SIMD computer, functional units are duplicated. N processing elements (PE's) perform the same instruction on different data, which are accessed in parallel memory banks through an IN(Interconnection Network). Tremendous computing power can be obtained because of the number of thousands of PE's. Then SIMD computers may be built with slower components and achieve the same performance as pipeline computers. But difficulties to design a suitable control unit to handle efficient mapping between the input and output of the IN become higher as the number of inlets(N) and outlets(M) of the IN's become big (several thousands). Full crossbar IN is not suitable to design SIMD, because the switches setting for $(N \times M)!$ permutations is very complicated to be a real design. An alternative is to divide IN into stages, and such a segmented network with each stage satisfying partial connection requirements yields to a multi-stage IN (MIN)[Fe81].

The organization of this paper is, section-1 has been devoted to introduction, and section-2 gives the background and motivations, while section 3 presents the new Benes like network, section-4 gives the set of manipulation showed to be useful for parallel computation. section-5 presents theorems and rules used for the setting of the new Benes network(i.e., the CU mechanism for IN), while section-6 gives examples for our mechanism. Finally section-7, is devoted to conclusions.

2-Background and Motivations

samples of MIN's are Omega IN [La82] and baseline IN[Fe81], Fig. 1, shows 8 inputs baseline IN. These networks are implemented with two states switching elements.Fig. 2, can show the four functions of the switching element used into Omega networkwhile the upper two states switching function have been used into baseline MIN,(see Fig. 2). These IN's are blocking MIN[Fe81], because they cannot perform or realize some important permutations between its $N=2^n$ inlets and its outlets, for instance Omega network cannot perform the shuffle, unshuffle, bit reversal, and etc.., [Lê78]

One solution to have a powerful permutation network is to increase the number of stages to $(2\log_2 N-1)$, like for example have a Benes IN(see Fig. 1, for 8×8 Benes IN), which posses a unique property of passing any input-output bijection. Benes IN; (B(n)), belongs to the class of rearrangeable IN [Op71], because it is able to realize N! permutations. But the control algorithm for B(n) is complex. Opferman[Op71] gave a looping algorithms which could calculate the control information according to the permutation function. With these algorithms Benes IN can realize arbitrary permutations in O(N log N) steps so these algorithms had a severe drawbacks to use for parallel computation[Lê78]. Feng[Fe81] has pointed out that when processing in parallel, the manipulation which are performed on vectors generally belong to some general family of manipulations but he did not specify what these families In this paper we have extended the manipulation family to a more strong set of permutations which is mostly used in parallel computation, and design parallel algorithms and setting theorems for realizing these manipulation set. This new mechanism can enhance the parallel computation for super system which use Benes type IN for achieving the complex mapping which are the most frequently used in O(log N) steps. If in case we need to use a manipulations that is not included in that set, we have suggested to use the conventional algorithms that is taking O(N log N) steps[Op71].

3- Benes like network

The nomenclature of this section is due to Benes network (B(n)) which is one kind of rearrangeable MIN[Bê65] capable to realize all permutations for mapping the input to output. The B(n) is symmetric ; the left half and the right half of the network are the

mirror image of each other, (see Fig. 1) with the middle stage as the plane of symmetry. The principle contributions of our paper relate mostly; to provide a new way to understand and prove the rearrangeability (or the universality; their ability to realize arbitrary permutations if multiple passes through them are permitted.) of 2(log N)-1 stages rearrangeable networks. We have considered N×N networks with $N=2^n$.

We will use the result of the following theorem[Ag83] to show the definition of Benes like network structure composites from two Baseline kind network.

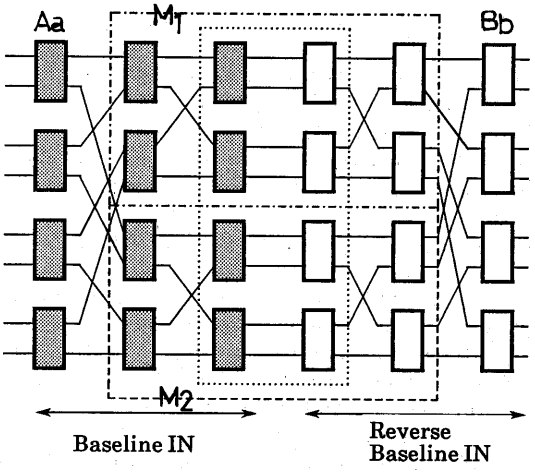

Fig. 1 Benes MIN constructed from cascading two baseline and reverse baseline IN, with the merging of the dotted block.

Theorem (Fig. 1)[Ag83]

A direct combination of n or (n-1) stages of the baseline and (n-1) or n stages of the reverse baseline MIN's; (we named it Benes like network) can pass any of the $2^n!$ possible permutations in a conflict free manner, i.e., Benes like network is equivalent to Benes network. We use these result to construct Benes like network such that we can build rearrangeable network from normal MIN like Omega or Baseline. Hence we consequently may have a rearrangeable network with a strong capability for reconfiguration.

Example of super computer based Benes network is GF11[Be85] constructed by IBM. The main objective was to increase the raw processing power rate of the PE's. This sample of SIMD machine has a capability of 11.52 GFlops.

But this machine has a draw back because the configuration of the connection scheme achieved by setting Benes network will be loaded previous to the run and remain constant through the whole execution of the problem.

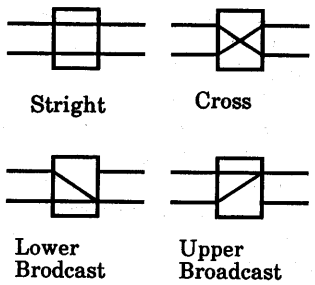

Fig. 2 Legitimate states of an interchange box(switching element).

Benes MIN consists in systematic way from the left and right stages which also, consist of N/2, two states switch elements (Fig. 2) which has two functions, either direct connection (I) or cross connection (π). In Fig, 1 the middle stages consist of two parts of rearrangeable networks with N/2 inlets. The connection pattern between left stage and middle stage is the reverse perfect shuffle $(\phi^n)^{-1}$. Symmetrically the middle and the right stage are connected by perfect shuffle connection function ϕ^n. Due to the rearrangeability definition we mentioned before, we can have the permutation P, then there are two permutations, A_a, B_b, in Benes IN which are subgroup of input set, also there are two other permutations M_1, M_2; such that the Benes IN can perform P, hence;

$P = (A_a, M_1, M_2, B_b)$

The control pattern for this network is a N/2 (2 log N-1) switches bit matrix which specifies the setting of each basic switch to perform the requested permutation.As we stated in the introduction that, most algorithms to compute the pattern setting for arbitrary permutation are required O(Nlog N) steps, thus the setup times for the network is much larger than the network delay which is O(log N).

Our trail are motivated in the way to find very strong manipulation functions and derive the control setting for these manipulation using special setting formulae for these manipulations. Hence according to these theorems we design the control unit for Benes MIN which can be able to realize that manipulation belonged to listed groups. The new mechanism can provide a setting steps in O(Log N), which is a new lower bound for Benes routing algorithm.

4- Selected Manipulation groups

Within this section we will embody the mostly used permutation function necessary for

manipulations used in parallel systems.

In the following we will give some definitions of groups of functions used for manipulation which will be used to design the control unit for either Benes or Benes like network.

Definition 1

Let $(x_n, \ldots x_1)$ is the binary representation of positive integer, X, whose most significant digit is ensigned by n, and least significant digit is ensigned by one, where $n=0$. Hence we can represent $(x_n, \ldots x_1)$ as; $X(_1^n)$ to avoid different manipulations entanglements.

Definition 2

The perfect shuffle permutation; ϕ, performs the following permutation;

$\phi(x_n, \ldots x_1) = (x_{n-1}, \ldots x_1, x_n)$, where $(x_n, \ldots x_1)$; is the binary representation of integer $X=(x_n,\ldots .x_1)=x_n2^{n-1}+\ldots\ldots x_1 2^0$, which is a subset of all postive integers, $\Sigma=\{0,1,\ldots\ldots 2^n-1\}$. The inverse of perfect shuffle; is $\phi^{-1}(x_n,\ldots x_1)=(x_1,x_n,\ldots\ldots x_2)$.

It is well known how the perfect shuffle has a big role in parallel processing as been noted in details by Stone[St71].

Definition 3

The bit reversal permutation function, ρ perform the following manipulation;

$\rho(x_n, \ldots x_1) = (x_1, \ldots x_n)=X(_1^n)$;

Bit reversal is also, very important permutation function especially for FFT algorithms used in signall processing[St71].

Definition 4

The flip permutation function α; performs the following manipulation ;$\alpha(x_n, \ldots\ldots x_1)= (x_n, \ldots\ldots x_1)\oplus \upsilon$; where υ is a fixed control vector, and $\oplus$ is a bit-by-bit EX-OR function.

Definition 5

The shift permutation function β, preforms the following permutation;

$\beta(x_n,\ldots\ldots x_1)_\upsilon= \beta(X(_1^n))_\upsilon= (X(_1^n))+\upsilon \mod 2^n$;

where υ is a fixed control variable vector, and $n=\log_2 N$, $0\leqq\upsilon<n$.

In fact, the flip and shift permutation functions were firstly, used in the design of STARAN computers[Ba82].

Definition 6

The permutation function A, which perform the following manipulation;

$A(x_n,\ldots\ldots x_1)_{j,\upsilon}=A(X(_1^n))_{j,\upsilon}= jX+\upsilon \mod 2^n$;

where $0\leqq j<n$, and $0\leqq\upsilon<2^n$, where j is odd and υ is a control vector.

However, the family defined by definition 6, is very useful for the implementation for the parallel Load and Store instructions.

Definition 7

The permutation function B, can perform segmented cyclic shifts, by j, such that;

$B(x_n, \ldots\ldots x_1)_{j,\upsilon}= B(x_n,\ldots\ldots x_{n-j+1}, \beta_\upsilon(x_{n-j},\ldots x_1))_{j,\upsilon}=(X(_{n-j+1}^n), \beta(X(_1^{n-j}))$; where $0\leqq\upsilon<2^n$, $j\leqq n$.

This permutation group, represent the circular shifts of amplitude υ, by varying j.

This permutation family can be useful in divide and conquer technique used frequently, in the design of parallel algorithms design, that is; when a computation on n items is replaced by two computations on N/2 items, and so on.

Definition 8

The permutation function; $C(X)_{j,h,m,\upsilon}$; can perform within the parameters; j,h,m,υ a different segmented perfect shuffle such that;

$C(x_n,\ldots\ldots x_1)_{j,h,m,\upsilon}=$

$C(x_n,\ldots\ldots x_{n-j+1},x_{n-j},\ldots\ldots,x_{n-j-h+1},x_{n-j-h},\ldots\ldots x_{n-j-h-m+1},x_{n-j-h-m},\ldots\ldots x_1)$;

hence; $C_{j,h,m,\upsilon}(x_n,x_{n-j}, x_{n-j-h}, x_{n-j-h-m})$, where;

$X_n=(x_n,\ldots\ldots x_{n-j+1})=X(_{n-j+1}^n)=X_a$

$X_{n-j}=(x_{n-j},\ldots\ldots x_{n-j-h+1})=X(_{n-j-h+1}^{n-j})=X_b$

$X_{n-j-h}=(x_{n-j-h},\ldots\ldots x_{n-j-h-m+1})=X(_{n-j-h-m+1}^{n-j-h})=X_c$

$X_{n-j-h-m}=(x_{n-j-h-m},\ldots\ldots x_1)=X(_1^{n-j-h-m})=X_d$,

then;

$C(X)_{j,h,m,k}=C(X)_{j,h,m,k}=C(X_a,X_b,X_c,X_d)_{j,h,m,k}=(X_a,X_d,X_c,X_b\oplus\upsilon)$.

This permutation function is very useful in parallel programming, especially; in FFT algorithms and parallel sorting, like Batcher's network, where has 2^{14} PE's[Ba82]. It is quite known, that the ability to effectively vectorize a problem and make optimal use of a vector structure depends on the availability of means for efficient data manipulations.

Definition 9

The identity permutation $I(X(_1^n))$ maps input to the same input, asif it is the output.

In the above case the switch will have a straight connection (see Fig. 2).

Definition 10

The exchange permutation $\pi(X(_1^n))$, will map input x onto $x+1$, if x is even or will map x onto x-1, if x is odd. i.e., generally x will be mapped onto $x\oplus 1$.

Fig. 2 shows the state of exchange. The upper two state in Fig. 2 presents the basic state for the switching element used in Benes IN, which needs only one bit of information from the control unit to set the these switches to either straight(0) or cross connection (1), but if we use the other lower two cases, as shown in the Fig. 2 then the CU should supply two bits of information in order to set the switch into one of the four states. so that the four functions switching element increase the complexity of the CU to set the MIN to realize arbitrary manipulation function.

Definition 11

The permutation $\lambda(X(_1^n))_j$; will map $X(n)=(x_n, \ldots\ldots x_1)$ onto $(x_n, x_{n-1},\ldots\ldots x_1 \oplus x_j)$; In other words, $\lambda(X)_j=X\oplus\{x_j\}$

***Definition* 12**

We may define a manipulation D(X(n)), which permutes different segmented bit reversal permutation; which depends on the basic definition of bit reversal given in definition-3.

If we suppose we can partition the string of X(n) into four sub strings according to the parameters; j,h,m,k; such that, $j \leqq h \leqq m \leqq n$; where;

$X(n)=\{x_n,.....x_1\}=X(_1{}^n)=\{x_{n-m}.....x_{m-h+1}, x_{h-m},....x_{h-j+1}, x_{h-j},.....x_{j+1},x_j,.....x_1\}$; where; $X(n-m)=\{x_{n-m},.....x_{m-h+1}\}=X(_{m-h+1}{}^{n-m})$;

$X(h-m)=\{x_{h-m},.....x_{h-j+1}\}=X(_{h-j+1}{}^{h-m})$;

$X(h-j)=\{x_{h-j},.....x_{j+1}\}=X(_{j+1}{}^{h-j})$;

$X(j)=\{x_j,.....x_1\}=X(_1{}^j)$, then we can represent the X(n) by the most significant digit; hence; X(n)= (X(n-m),, X(m-h), X(h-j), X(j)), and;

$D(X(n))=(X(n-m), \rho(X(m-h)), X(h-j), \rho(X(j)))\oplus v$.

5-The control unit mechanism

As we mentioned previously, that our control mechanism, is based on stability concept, as well as the partitionability of the permutation function, in a way suitable to be similar to Benes network. Hence to compute and analyze the control pattern for the most used manipulation mentioned in the pervious definitions, we have given in the following a statement of equations stated as theorems with its proofs in order to show its validity to be used as setting rules to compute the control pattern for the above permutation functions realized on Benes network. All theorems are based on some rules of algebra, and quotient groups[Bi77]. If v is a positive integer then v divided by two results two positive numbers, one is the quotient; v', and other is the remainder v_1; hence $v=2v'+v_1$.

Theorem 1

The Flip permutation (Definition 4); can be mapped by the Bnese IN using the following equation;

$\alpha(X(_1{}^n))_v=[I(_1{}^n), \alpha(X(_1{}^{n-1}))_{v'}, \alpha(X(_1{}^{n-1})_{v'}, \alpha(X(_1{}^n))_{v_1}]$; if $n \geqq 2$. But if $n=1$; then $\alpha(X(_1{}^1))_0=I^1$; $\alpha(X(_1{}^1))_1=\pi(X(_1{}^1))$.

Proof;

suppose $X=(x_n,.....x_1)$ which belongs to Σ^n, we can represent $X=(x',x_1)$, so that, x', represent the n-1 tauple of $(x_n,.....x_2)$, suppose the letter L represents the left stage of Benes network (see Fig. 5), M represents the middle stage, and R is the right stage. then by following the mapping performed by these stages we can have the following, $\phi^{-1} \circ L$ will map (x',x_1) onto (x_1, x'), then by $M \circ \phi \circ L$ results the mapping $(x_1, x'\oplus v')$; this because of Flip permutation function, then by shuffling this by ϕ, we can have the mapping, $(x'\oplus v',x_1)=(x',x_1)\oplus(v',0)$, then by the R stages, which is exchange, then we can have; $((x',x_1)\oplus(v',0))\oplus v_1=(x',x_1)\oplus(v',v_1)=x\oplus v$. □

Theorem 2

The permutation function defined by (Definition 6) can be setted by;

$A(X(_1{}^n))_{j,v}=[I(X(_1{}^n)), A(X(_1{}^{n-1}))_{j,v'}, A(X(_1{}^{n-1}))_{j,j^*}, \alpha(X(_1{}^n))_{v_1}]$; where $j^*=j'+v'+v_1$,, $n \geqq 2$, if $n=1$, then $A(X(_1{}^1))_{1,0}=I(X(_1{}^1)), A(X(_1{}^1))_{1,1}=\pi(X(_1{}^1))$.

Proof:

In order to proof the above theorem, consider $X=(x',x_1) = 2x'+x_1$; A word of data is broadcast to upper median switch or lower median switch according to the parity of the number of the line by which it enters the network., then integer X is mapped by the first stage($\phi^{-1} \circ L$) such that; we have; $x'+2^{n-1}x_1$ then by the middle stage, we will have two cases, one if $x_1=0$(even-case), then we have (jx' v') Mod. (n-1), the other case is when $x_1=1$ (odd-case), such that;

$(jx'+j'+v'+v_1)$ Mod. (n-1), then map the even case y the perfect shuffle ϕ, we may have;

$(2jx'+2v')$ Mod. $n=(jX+2v')$ Mod. n =, (because, $X=j2x'+jx_1$; as well as z is even;

then $(\kappa_1\oplus z)=(\kappa_1+z)$, in this case κ is either 0, or 1., hen by the last stage, we can have $(jX+2v'+2v_1)$ od n which can be equalled to $(jX+v)$ Mod n.

The other case is when $x_1=1$(odd-case), then; by he perfect shuffle mapping we may have, $j(2x')+(2j'+1)+2v'+2v_1)$ Mod.n $=(jX+2v'+2v_1)$ od. $n=w$; as well as w is odd, then $w\oplus v_1=w-v_1$, hen the odd part of X is mapped by the last stage s; $(jX+2v'+v_1)$ Mod. $n=(jX+v)$ Mod. n. □

Theorem 3

This theorem will set the segmented cyclic shift ermutation function defined in (Definition),according to the following setting:

$B(X(_1{}^n))_{j,v}=[I(X(_1{}^n), B(X(_1{}^{n-1}))_{j,v'}, B(X(_1{}^{n-1}))_{j,v^*}, \alpha(X(_1{}^n))_{v_1}]$; where $v^*=v'+v_1$; and $n \geqq 2$, $\leqq j<n$.

Proof;

suppose $X=(x_n,.....x_{n-j+1}, x_{n-j},...x_1)=(x_n, x_{n-j})$; here;

$x_n=(x_n,.....x_{n-j+1})$, and; $x_{n-j}=(x_{n-j},....x_1)$; then $x'_{n-j}=(x_{n-j},....x_2)$, and ; $x_{n-j}=2x'_{n-j}+x_1$, then the ntegr X may be mapped by the first stage so that we an get; (x_1, x_n, x'_{n-j}), there after using the middle tage we may get the following:

$(x_1, x_n, (x'_{n-j}+v_1x_1+v')$ Mod. n-j. But we may have wo conditions; one in case $x_1=0$; such that we get;

$(x_1, x_n, (x'_{n-j}+v')$ Mod. n-j-1) , the other case is hen $x_1=1$; such that;

$(x_1, x_n, (x'_{n-j}+v_1+'v)$ Mod. n-j-1) ; then; by apping the perfect suffle, we get;

$(x_n, (x'_{n-j}+v_1x_1+v')$ Mod. n-j-1, x_1); going on to ast stage we may have;

$(x_n, (x'_{n-j}+v_1x_1+v')$ Mod. n-j-1, $x_1\oplus v_1)=(x_n, v)$; here v is also integr, then; $v=(2x'_{n-j}+2v_1x_1+2v'$ $x_1+v_1-2x_1v_1)$ Mod. n-j

$(2x'_{n-j}+x_1)+(2v_1+v_1)$ Mod. n-j m = $(x_{n-j}+v)$, Mod n-j $=\beta_K{}^{n-j}.\ x_{n-j}$. □

Theorem 4

The segmented shuffle permutation defined in defenition-8, can be setted according to the following setting:

$C(X(_1{}^n))_{j,h,m,v}$= [$\lambda(X(_1{}^n))_{j^*}$, $C(X(_1{}^{n-1}))_{j,h-1,m+1,v'}$, $C(X(_1{}^{n-1}))_{j,h-1,m+1,v^*}$, $\alpha(X(_1{}^n))_{v\#} \circ \lambda(X(_1{}^n))_{j\#}$]; where j^*= n-j-h+1 ; $j\#$=h+m+1 ; v^*= $v'\oplus 2m+h+1$; $v\#=v_1\oplus v_{h+m+1}$

The value of h will make diffrent conditions for the proof; the case when h>0, n≧2, and j+h+m<n, will give us the above setting equation.

But the condition when h=0; we will get; $C(X(n))_{j,0,m,v}=C(X(n))_{j,m,0,v}$; hence; $C(X(n))_{j,0,0,v}=\alpha(X(n))_v$.

The condition h>0, j+h+m=n; then; $C(X(n))_{j,h,m,v}=C(X(n))_{j,h,0,v}$ then;

if n =1; then ; $C(X(n=1))_{j,h,m,v}=\alpha(X(n=1))_v$.

Proof

$\lambda(X(n))_j$ has been defined in definition ?; if n≧2, h>0, and j+h+m<n, then according to definition(7); we have X(n))= (X_a,X_b,X_c,X_d); let $X_d=(X_d',x_1)$; $X_b=(X_b',x_{n-j-h+1})$; then; $X(n))=(X_a,X_b',x_{n-j-h+1},X_c,X_d',x_1)$, will be mapped by first stage and the reverse shuffle; so that; we may have;

$(x_1\oplus x_{n-j-h+1},X_a,X_b',x_{n-j-h+1},X_c,X_d')$, then by the middle stage we get;

$(x_1\oplus x_{n-j-h+1},X_a,X_b',v,X_c,X_d')\oplus\kappa'$; where $v=x_{n-j-h+1}\oplus(x_1\oplus x_{n-j-h+1})=x_1$; actually the bit $v=x_{n-j-h+1}$, if $(x_1\oplus x_{n-j-h+1})=0$; otherwise $v= x_{n-j-h+1}\oplus 1$; then passing through perfect shuffle we get; $(X_a,X_d',x_1,\ X_c,X_b',x_1\oplus x_{n-j-h+1})\oplus 2v'$ =w; the $\lambda(X(n))_{j^*}$; where $j^*=h+m+1$, is the bijection used onto w; so that, we get;

$\lambda(X(n))_{j^*}\circ w= (X_a,\ X_d',\ x_1,\ X_c,\ X_b',\ x_{n-j-h+1})\oplus 2v'=(X_a,X_d,X_c,X_b)\oplus 2v'$; then buy the last stage we may have;

$(X_a,X_d,X_c,X_b)\oplus(2v'\oplus v_1)$. □

Theorem 5

We can set the permutation D(X(n)); defined in definition 12, according to the different parameters; j,h,m,v; such that; if j=0 or 1, and n≧2, then; we can have the following setting;

$D(X(_1{}^n))$=[$I(X(_1{}^n))$, $D_1(X(_1{}^{n-1}))$, $D_2(X(_1{}^{n-1}))$, $\alpha(X(_1{}^n))_{v1}$]; where

$D_1(X(_1{}^{n-1}))$= (X(n-(m-1)), ρ(X((m-1)-(h-1)), X(h-1))⊕v' ; $D_2(X(n-1))=D_1(X(n-1))$. But if we examine the case when j ≧2, we may have the following setting;

$D(X(_1{}^n))$=[$\lambda(X(_1{}^n))_j$, $D_1(X(_1{}^{n-1}))$, $D_2(X(_1{}^{n-1}))$, $\alpha(X(_1{}^n)_{v_2}\circ\lambda(X(_1{}^n))_j$];where $v_2=v_1\oplus v_j$;

$D_1(X(_1{}^{n-1}))$= (X(n-(m-1)), ρ(X((m-1)-(h-1))), X(h-1-(j-2)), ρ(X(j-2)))⊕v' =

$D_1'(X(_1{}^{n-1}))\ \oplus v'$; $D_2(X(_1{}^{n-1}))= D_1'(X(_1{}^{n-1}))\ \oplus v^*$; where $v^*= v'\oplus 2j\text{-}2$; we may have another case when j=h=0, then;

$D(X(_1{}^n))$= (X(n-m), ρ(X(m-0))) ⊕ v= (X(n-m), ρ(X(m))) ⊕ v, hence if n=1, we may have; $D(X(n=1))=\alpha(X(n=1))_v$

proof;

As we have expressed in definition 12, that we can represent X(n) as;

X(n)= (X(n-m), X(m-h), X(h-j), X(j); For the case of jj ≧2, going on to the first stage and by induction;

assume fitst; X(jj)= X(j)- $\{x_j,x_1\}$; where x_1; is the least siginficant digit, and x_j; the most significant digit; then;

X(n)= (X(n-m), X(m-h), X(h-j), $\{x_j\}$, X(jj), $\{x_1\oplus x_j\}$); then by theinverse shuffle we may have;

X(n)=($\{x_1\oplus x_j\}$, X(n-m), X(m-h), X(h-j), $\{x_j\}$, X(jj)), going on to the middle stage;

($\{x_1\oplus x_j\}$, X(n-m), ρ(X(m-h)), X(h-j), $\{x_j\}$, ρ(X(jj)) ⊕ (v'⊕ 0).

The value of $\{x_1\oplus x_j\}$, can inform us about the connection which to be setted through the upper or lower median switch; hence, if its value is zero, then the connection is setted across the upper median switch, and if its value is one we may set the lower median switch. hence the aboove equation becomes;

($\{x_1\oplus x_j\}$, X(n-m), ρ(X(m-h)), X(h-j), $\{x_j\oplus 1\}$, ρ(X(jj)) ⊕ (v'⊕ 0). the above conditioned mensioned above can be joined together such that;

($\{x_1\oplus x_j\}$, X(n-m), ρ(X(m-h)), X(h-j), $\{x_1\}$, ρ(X(jj)) ⊕ (v'⊕ 0). there after shuffling we may have;

(X(n-m), ρ(X(m-h)), X(h-j), $\{x_1\}$, ρ(X(jj) $\{x_1\oplus x_j\}$) ⊕ (v'⊕ 0). going on to the last stage ; and due to, $\lambda_j(X(n))$, we may have $(x_1\oplus x_j)\oplus x_j= x_j$.; hence;

E(X(n))= (X(n-m), ρ(X(m-h)), X(h-j), $\{x_1\}$, ρ(X(jj) $\{x_j\}$) ⊕ (v'⊕ v_1); becuase,

$\{x_1\}$, ρ(X(jj) $\{x_j\}$= ρ(x(j); then

E(X(n))=X(n-m), ρ(X(m-h)), X(h-j), ρ(x(j))⊕v □

As we mentioned before that our controlling mechanisms for the Benes IN's CU designs will enhance the capability for efficient routing for the Benes IN for reconfigurable architecture.

Consequently, if we want to permute a manipulation which is not included in the definitions presented in this work that has setting time of O(log N), then we have to use the conventional constructive algorithm[Op71] which has O(N log N) steps setting time. Of course, this inherently, depends on kind of the application. For example the Fast Fourier Transform (FFT) computational algorithms uses perfect shuffle interconnection pattern as an efficient execution for the algorithm transformation on a parallel process. The perfect shuffle interconnection pattern is sufficient for executing the transform algorithms on parallel processor. The entire processor repeats the sequence; *shuffle, multiply* and *transfer* result back to input network; Log N=n time steps to compute FFT. We also can apply such MIN for polynomial

evaluation, sorting and matrix transpostion.

We can estimaite the total computationl time complexity according to following corollary.

Corollary;

If μ is the percentage of the frequent usage for arbitrary permutation functions which are included in the previous mentioned theorems, and if ξ=100-μ is the percentage of the frequent usage of arbitrary permutations not included in the listed permutations (Section-4), then we may express the total computational time complexity of our Benes like network by the equation;

ξ O(N log N) + μ O(log N);

Proof;

The proof is an immediate consequence of the setting rules specified by the theorems of section-5, which can yield a setting time of O(log N). Thus the frequent occurance of this time is dominated by μ which has a bigger value while the time dominated by ξ is proportional to the frequency of using other permutation functions not listed in this paper, which have been supposed to have low percentage of occurance. □

Fig. 3 shows such a comparision. Hence due to our hypothesis, the value of μ will nearlly be equal to one(i.e., 100%), while the value of ξ is nearlly equal to zero. But in order to see the exact behaviour of the above factors we have computed the total complexity time dominated as a function of μ to see the effect of using our mechanism in routing control of Benes network for arbitrary permutation.

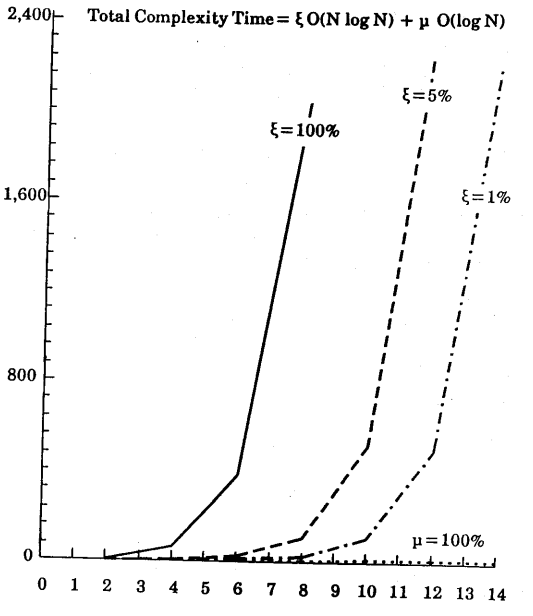

Fig. 3, The relation between the input size as; $Log_2 N$ and the total steps necessary for setting; represented as total complexity time.

6-Examples

In order to show, how shall we use the mechanism expressed in this paper. We have presented two examples representing how shall we set the Benes IN according to a given manipulation function using the above metioned mechanism.

The first example was to realize bit reversal manipulation function, such that $D(X(_1^n))$, where j=n=3, h,m,,υ=0, hence due to that, $\rho(X(_1^n)=D(X(_1^n))$. Then to map the 8-word vector of data, using Benes IN, and using Theorem 5 we may have;

$D(X(_1^3))=[\lambda(X(_1^3)_3,\ D_1(X(_1^2)),\ D_2(X(_1^2)),\ \alpha(X(_1^3)_0 \circ \lambda(X(_1^3)_3]$; as well as the υ value of $\alpha(X(_1^3))_0$ is zero, then this mapping represents identity function (according to theorem 1), also; according to sub condition of theorem 5, by further partiotioning we may have;

$D_1(X(_1^2))_{\upsilon=0}=[X(_1^2),\rho(X(_1^1))]=[I(X(_1^2)),D_{11}(X(_1^1))_0, D_{12}(X(_1^1))_0, I(X(_1^2)_2]$;

$D_{11}(X(_1^1))_0=\alpha(X(_1^1)_0=I(X(_1^1)$ and $D_{12}(X(_1^1))_0=\alpha(X(_1^1)_0=I(X(_1^1)$ (This is due to theorem 1 and 5).Then ;

$D_2(X(_1^2))_{\upsilon=2}=[X(_1^2),\ \rho(X(_1^1))] \oplus 2=[I(X(_1^2)), D_{21}(X(_1^1))_1, D_{22}(X(_1^1))_1, I(X(_1^2)_2]$;

$D_{21}(X(_1^1))_{\upsilon=1}=\alpha(X(_1^1))_1=\pi(X(_1^1))$ and $D_{22}(X(_1^1))_{\upsilon=1}=\alpha(X(_1^1))_1=\pi(X(_1^1)$; this is due to theorem 1 and 5.

Hence the control setting for the Benes network to realize bit reversal is shown in Fig. 4 due to the above setting.

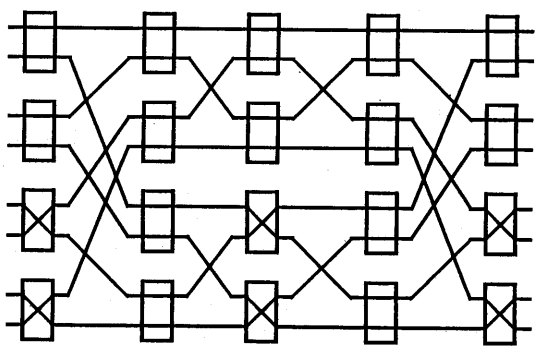

Fig.4 The setting of Benes IN to realize it reversal permutation function for input=8.for example-1

The second presented example is to realize for example perfect shuffle mapping function using the setting defined by theorem 4. It is well-known how much this setting is difficult for normal MIN. As we mentioned before in Def-7, that; if we set for n=3, the parameters j=m=υ=0, and h=1, then we have

$C(X(_1^n)) = \phi(X(_1^n))$;

this function).

Due to theorem 4 we may have;

$C(X(_1^3)) = [\lambda(X(_1^3))_3, C_1(X(_1^2))_0, C_2(X(_1^2))_4, \alpha(X(_1^3))_0 \circ \lambda(X(_1^3))_2] =$

$[\lambda(X(_1^3))_3, C_1(X(_1^2)), C_2(X(_1^2)), \lambda(X(_1^3))_2]$; where;

$C_1(X(_1^2))_0 = [\lambda(X(_1^2))_2, \quad C_{11}(X(_1^1))_0, \quad C_{12}(X(_1^1))_2, \lambda(X(_1^2))_2] =$

$[\lambda(X(_1^2))_2, I(X(_1^1)), \pi(X(_1^1)), \lambda(X(_1^2))_2]$; this the setting for the upper median switch.

$C_2(X(_1^2))_4 = [\lambda(X(_1^2))_2, \quad C_{21}(X(_1^1))_0, \quad C_{22}(X(_1^1))_2, \alpha(X(_1^2))_2 \circ \lambda(X(_1^2))_2] =$

$[\lambda(X(_1^2))_2, I(X(_1^1)), \pi(X(_1^1)), \pi(X(_1^1)) \circ \lambda(X(_1^2))_2]$; as shown in Fig. 5. By this way of seeting we can be able to realize any kind of permutation with a complexiety depends on degree of n.

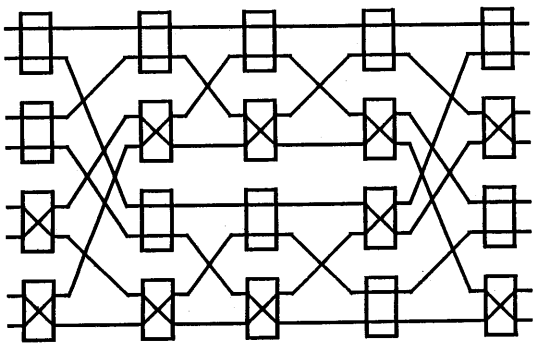

Fig.5 The setting of Benes IN to realize perfect shuffle permutation function for input=8.for the second example.

7- Conclusion

In this paper we have presented a class of multistage interconnection network based on Benes rearraengeable MIN, which has been proved on its capability to realize any kind of permutation setting, but because there are difficult problems facing the control algorithms for setting arbitrary manipulation function, has made the usage of this class of IN, narrow and very limited for only a small degree of n, ($n = \log_2 N$, N is the number of input terminals). Hence, to use this class of MIN, especially as an enhancement for the design of super systems, it becomes important to increase the mapping capabilities of Benes network to be very efficient to use for parrallel computations. Hence we have innovated a control algorithms to realize the manipulations functions used mostly in parallel computation. This mechanism can increase the capabilities of Benes IN because the setting time is proportional to degree of n, while the conventional methods used setting algorithms proportional to N times log N, which is rather very long for practical use in parallel computers.

The mechanism we presented here, has been depended on the symmetricality of Benes network. This may add a constraints for implementing such network on VLSI chips. Besides, the complexity of the CU design increases a we add more theorems for analyzing a more wide set of useful permutation groups. Hence, it becomes necessary to design rearrangeable MIN on another new criteria that do not depend on symmetricality, i.e., without relaying on Slephian-Duguid theorem[Bê64]. We also have been working on such problem, so that in coming paper we will give another methodology to analyze the rearreangeable IN's, so that the applications of such IN's for reconfigurable architecture become extensively, more practical.

REFERENCES:

[Ag83] Agarwal, D.P., "Graph theoretical analysis and design of multistage interconnection networks," IEEE *Tran. on Comp.*, vol. C-32, pp. 636-648.

[Ba82] Batcher, K.E., "Bit serial parallel processing systems," IEEE *Tran. on Comp.* vol. C-31, may 1982, pp. 377-384.

[Be85] Beetem, J., et.al., "The GF11 supercomputer," The 12th annual *Symp. on Comp. Architecture*, vol.13, June 1985, pp. 108-123.

[Bê64] Bênês, V. E., Permutation groups, complexes, and rearrangeable connecting networks," *Bell system Tech. J.*, 43, 1619-1640, July, 1964.

[Bi77] Birkhoof, G., et.al. *Modern Algebra*, 4th addition, 1977, Macmillan Publishing Co., Inc.

[Fe81] Feng, T., "A survey of interconnection networks," IEEE *Computer*, vol. 14, Dec. 1981, pp. 12-27.

[Ha86] Hamid, I.A., et.al. "Highly distributed architecture for a commanding executing systems," *Int. Conf. on computer technology*, Mar. 1986, A2-3.

[La82] Lawrie, D.H., "The prime memory system for array access," IEEEE *Tran. on Comp.* vol. C-31, May 1982, pp. 435-442.

[Lê78] Lênfant, J., "Parallel permutations of data: A Benes network control algorithm for frequently used permutations," IEEE *Tran. on Comp.* vol. C-27, pp 637-647, July, 1978.

[Op71] Opferman, D.C., et.al., "On a class of rearrangeable switching networks," *Bell System Tech. J.* vol. 50, May-June 1971, pp. 1579-1600.

[Si81] Siegel, H.J., et. al. "PASM: a pratitionable SIMD/MIMD system for image processing and pattern recongnition," IEEE *Tran. on Comp.* vol. C-30, Dec. 1981, pp. 934-947

[St71] Stone, H.S., "Parallel processing with the perfect shuffle," IEEE *Tran. on Comp.* vol. C-20, Feb. 1971, pp. 153-161.