計算機アーキテクチャ 70－2
（1988. 3. 10）

ベンズＩＮのための高速並列計算モデル

アイサム 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

本論文は、ベンズ *IN* (**Interconnection Network**) の並列処理への応用が、より実際的となるような高速制御アルゴリズムについて考察している。まず、サイクリック・キューブ・トポロジーに基づいた *CCE* (**Cyclic Cube Engine**) と呼ぶ高速並列計算モデルを構成する。但し、$\Phi \geqq \mathbb{N}$ とする (これを制限されない並列性と呼ぶ)。ここで、Φ と $\mathbb{N}$ は、それぞれ、計算モデルを構成する処理要素の数と入力の数を示す。次に、このモデルを用いて、任意のパーミュテーションを $O(\log_2 \mathbb{N})$ の時間で実現可能とするベンズ *IN* の並列セッティングアルゴリズムを与える。これは、前述の並列セッティングアルゴリズムを *Accelerator,* と呼ぶもう1つの高速アルゴリズムにより駆動することにより実現される。この結果は、*nonshared,* 並列計算モデルを用いて、ベンズ IN をセッティングするための時間の理論的な下限値 $O(\log_2 \mathbb{N})$ を与えている。

A New Fast Parallel Computation Model for Benes Interconnection Network – With Unrestricted Parallelism

Issam A. HAMID, Norio SHIRATORI and, Shoichi NOGUCHI

Research Institute of Electrical Communication,
Tohoku University, 1-1-2, Katahira, Sendai. 〒980.

This paper has given another consideration for Benes rearrangeable Interconnection Network (IN), to be controlled and set by very fast algorithms such that to make such networks more practical for parallel computations applications. We have presented here, for the Control Unit(CU) of Benes IN, a new fast nonshared memory parallel computational model named as; Cyclic Cube Engine (CCE), which depends on the cyclic Cube topology, where the number of processing elements Φ *are equal or more than the number of the* $\mathbb{N}$ *data items consisting the permutation(i.e., unrestricted parallelism). We have constructed on this model parallel algorithm for parallel settings of Benes IN in order to realize arbitrary permutation with a setting time of* $O(\log_2 \mathbb{N})$*; (assuming* $\mathbb{N}$ *is base 2). This bound could be achieved by accelerating the parallel setting algorithm by function call of another very fast algorithm named as the Accelerator. Consecquently, our construction could approach, the lower theortical bound of setting Benes IN of* $O(\log_2\mathbb{N})$*, in parallel using nonshared parallel computational model.*

I -Introduction

One of the key problems in parallel processing research deals with designing cost-effective rearrangeable networks. But Benes Interconnection Network (IN) is not easy to setup as the networks with $O(N^2)$ connections (N is the number of inputs). It runs in $O(N \log_2 N)$ time, which is worse than $O(N)$ setup time of networks with $O(N^2)$ connections, (assuming N is base 2). Such a long time has limited or restricted such types of networks to be useful for parallel computation [Ha87], inspite of its richness to support the mappings of N! permutations. There is a parallel algorithm[Na82] which sets the N-input in $O(\log^4 N)$ time. The main shortcoming of this algorithm as well as the others[Or87] is that it requires $O(N)$ processors with $O(N^2)$ connections among them, because of using a shared memory computation model. Also, those previous algorithms are time consuming and not suitable for supersystems[Ha87].

This paper further examines the setup problem of Benes IN with a new computational model of $O(N)$ processors with $O(3)$ connections among them. Hence, the main goal is to reduce the setting time of Benes IN to reach the lower bound of $O(\log_2 N)$. Such a goal motivate us to construct a suitable parallel computation model for Benes IN's Control Unit(CU), depended in this paper on Cyclic Cube topology. We have implemented on this computational model a powerful parallel permutation algorithms suitable to relocate dynamically arbitrary records of a permutation in PEs connected as Cyclic Cube topology. We have called these algorithms implemented on CCE as an accelerator, because it accelerates the parallel mapping of Benes IN. In other companion paper we have proven the lower computation bounds needed for this model.

In this paper, Chap. I, have discussed the background concerning with the use of Benes rearrangeable IN for parallel processing. Chap. II, has discussed a new computation model for the CU of Benes IN named as Cyclic Cube Engine(CCE). Chap. III has given a parallel permutation algorithm depends on the radix-sort, implemented on the CCE with processing time of $O(\log_2 N)$. Chap. IV, have discussed the realization of the parallel setting of Benes IN using the CCE computation model with the parallel permutation algorithms of Chap. III, to accelerate the setting time for arbitrary permutation. Chap. V has devoted for the conclusions.

Assumptions and Notations:

The following assumptions are used throughout this paper:

1) We assume that this model has unlimited or unrestricted parallelism where the number of modules is tolerable to the problem size(i.e., every processor has one record). However, this assumption is for simplicity. We have also, worked on the limited parallelism problem(i.e., $N \gg$ the number of processors such that more than one records are assigned to one processor) which were discussed in other companion paper[Ha88].

2) Each Processor Element; $PE(i)$ has three registers, $\beta(i)$, $\gamma(i)$, $\mu(i)$, which correspond to that PE, and enough memory to hold one record; $R(i)$, (this includes also, the field $F(i)$).

3) The distinction between null and a record in a PE is possible, and one bit tag could be used.

4) $(:=)$ represents an assignment, $(\leftarrow)$ represents the connection and routing of two neighboring processors (has one unit route $O(1)$).

5) i_ϕ, represents the bit ϕ of the binary representation of i. Consequently, the PEs can be selected according to the mask specified by the bit ϕ.

6) $i_{//\phi}$, represents the complementary of bit ϕ of the binary representation of i.

7) $i(\phi\ \psi)$ represents the selected bits, of the binary representation of i starting from bit ϕ to bit ψ, i.e., $i(\phi\ \psi) = i_\phi, \dots i_\psi$.

8) The complexity of an algorithm will be measured in terms of PE time needed and the number of unit routes; assuming that, the arithmetic operations can be performed in $O(1)$ time. The computation carried out locally at a PE is not counted here.

9) The computation model is a nonshared memory model of an SIMD type machine, where the number of PEs is denoted by Φ.

II. The Cyclic Cube Engine

The computation model used in our work is a nonshared memory model. All instructions and data are loaded to a specific number of processors which communicate through both cubic and cyclic configuration.

The CCE, is a network of identical processor elements, as shown in Fig. 2, where each *Processor Element (PE(i))* contains an operand register, a field memory locations, and basic arithmetic and logical capabilities.

Let N be the number of input or the element of a permutation, as base 2, i.e., $N = 2^n$. $R(i)$ represents a record located in $PE(i)$, and $F(i)$ represents the record's field. Let for some k such that, $2^k \geqq n$ and, $1 \leqq k \leqq n$. Then each *PE(i)* has $n+k$-bit address $F(i)$ expressed as (x,y) of integers, where x has n bits length $(0 \leqq x < 2^n)$, and y has k bits length $(0 \leqq y < 2^k)$, such that; $x \cdot 2^k + y = R(i)$, where $R(i)$ represents the address of $PE(i)$, whereas, $0 \leqq R(i) < 2^{n+k}$, $N = 2^n$, and the number of Φ processor elements; (PEs) consisting CCE is;

$\Phi = 2^{n+k} = \mathbb{N}^{(1+1/h)}$; where $h = \lceil n/k \rceil$, (where $\lceil .. \rceil$, represents the ceiling function,) such that, $2^k \geqq n$. Each PE has $(n+k)$ bit address $R(i)$ expressed as (x,y) of integers. Due to Fig. 2 each PE has *three* interconnection ports:
Next(x,y) is connected to *Before*$(x, (y+1) \bmod 2^k)$,
Before(x,y) is connected to *Next*$(x,(y+1) \bmod 2^k)$, and
Cylce(x,y) is connected to *Cycle*$(x+Y \cdot 2^y, y)$, where $Y = (1-2\ x^y)$. (Where mod; represents the modulus calculation.)

The ports *Next* and *Before* are within the cycles, (each cycle consists of 2^k PEs). While the ports *Cycle* is between cycles which being connected as the cube configuration. Theses cycles are in turn interconnected as a n-Cube.

III. The Parallel Permutation Algorithm- (The Accelerator)

The main purpose of this chapter is to permute the records of any arbitrary permutation, in parallel. It gives us how to accelerate the structure of the CU of Benes IN to reduce the setting time of Benes IN for arbitrary permutation and increase the efficiency of the parallel setting of Benes IN algorithm given in Chap.IV. Please note that, the setting time for this type of network is the lower bound which is proportional with the propagation delay of Benes IN itself, (i.e., $O(\log_2 \mathbb{N})$). The algorithm performs arbitrary permutation in $O(h \log \mathbb{N})$ time on the CCE for the case of unrestricted parallelism, such that Φ are tolerable to $\mathbb{N}$. Therefore, every PE may process only one record. Also, $\mathbb{N}$ should always match the PEs such that to not let more than one record applied to one processor(in this paper) and restricted parallelism by letting the PEs to have more than one record when we have $\mathbb{N} \gg$PEs, (of the companion paper) using, $\Phi = \mathbb{N}^{(1+1/h)}$ PEs with a connection of degree *three* per processor.

III.1. Partitioning CCE into 2^h-Group-Cycles

At first, let us observe the binary representation for the addresses of the PEs which consisted the CCE. We have tabulated the addresses of the PEs of the CCE model by a two dimensional table (as shown in Fig. 3). Every row represents the PEs' addresses for a cycle. While columns represents the addresses of PE of 2^n PEs per columns for the n-Cube between n-Cycles, such that every column represents one dimension related to the cube configuration. The k low significant bits for all PEs of every column are equal while they are different in their n most significant bits. In contrast, all PEs of every row have equal n most significant bits and different k most significant bits. Due to these observations we can partition CCE into obstructed groups, which is done according to the table's rows, i.e., the CCE's cycles. Therefore, *2^k-group-cycles,* is defined as a sequence of PEs ($h = \lceil n/k \rceil$, $0 \leqq h \leqq n$) of 2^h consecutive rows or cycles. Each column of this group is a 2^h-groups of PEs. All indices in *2^h-group-cycles* have the *same* h^{th} MSB (Most Significant Bit) of the h MSBs, and all other indices varies in only the other LSBs of the binary representation of PE. For example, if $h=2$, then the 2nd MSB of the $h=2$ MSBs, is the fourth bit i.e., 2^h-bit=fourth bit, which is the dominated partitioning bit such that we can have two groups of rows. The first four rows has the fourth bit of 0, and the other four groups has the fourth bit of 1. In the same way, we can have four groups of *2^{h-1}-group-cycles,* of which the bit, $(n+k-2) = 3^{rd}$ MSB; is either 0 or 1. Such partitioning assists us to construct our algorithms, as will be shown later.

III.2. The construction of the Parallel Permutation Algorithm

This section consists from three subsections. One represents an informal description with an example, the other gives the formal details for that construction, and the third, represents the analysis of those algorithms.

Our permutation algorithm is a parallel version of MSD (Most Significant Digit) radix sort. The radix$=2^k$. Then $\lceil n/k \rceil$ digits of $F(i)$ are used. The following rules corresponds to how to obtain the MSD radix digits.

1) The binary representation of F(i) is obtained.
2) The k MSB yield the MSD (Most Significant Digit).
3) The next to the k bits give the next digit and so on.

For $0 \leqq i < \mathbb{N}$, $2^n = \mathbb{N}$, $R(i)$ represents the record which is initially located in PE(i). $F(i)$ represents a field in record $R(i)$. $(F(0), \ldots\ldots. F(\mathbb{N}-1))$ define a permutation of $(0, \ldots.. \mathbb{N}-1)$. Recall that, $F(i)$ is related to the permutation element i, and $R(i)$ is related to a record in PE(i). The records are to be *permuted* so that following the permutation, $R(i)$ is in PE($F(i)$). This permutation is to be performed on the parallel computation model of $\Phi = \mathbb{N}^{(1+1/h)} = 2^{n+k}$ PEs, connected as cyclic cube topology where, $h = \lceil n/k \rceil$, $1 \leqq k < n$, $\mathbb{N}^{(1+1/h)}$.

III.2.1. The Informal Description

Before we discuss the formal representation of parallel permutation algorithms, we have presented here an informal discussion supported with an example in order to show our construction's strategy.

For example, if $k=2$, $n=3$, then the permutation elements; $0, \ldots.. \mathbb{N}-1$ are represented as k integers; $\sigma_1 \sigma_0$ with radix $2^k=4$, such that $00=0$, $01=1$, $10=2$, $11=3$, $20=4$, $21=5$, $30=6$, $31=7$, (according to the above three rules). We can permute for example, 8 records using 2^{3+2} PEs, as shown in the example given in Fig. 4, in which every box represents a PE, and the number inside it represents a record. Recall that, these processors are connected by CCE. Using radix 2^k in the representation of F(i) of the above example, the permutation algorithm will have $k=2$ passes, for re-locating records with respect to the MSD; σ_1, and LSD; σ_0 respectively. In the *first* pass, records are re-located with respect to σ_1 of F(i) such that, the record of PE(i) is routed to the PE of Row(i), Column($\sigma_1(i)$). Then the record are organized (or leveled) such that the record in PE(i) will be re-located in the PE which is located by the sum of the pervious locations.

While in the *second* pass, records are re-located with respect to σ_0 of F(i), and done in the *group-cycles*, due to σ_0, (low radix). This ordering will be done in parallel, for groups of records with the same value of σ_1. For example, there are four *group-cycles* of two rows $2i$, $2i+h$, (i.e., 2^{h-1}*-group-cycles.*). Then for each *group-cycles* records are routed to the row σ_0 within the same column. Then these records are leveled in parallel within each *group-cycles* and re-located to the appropriate column without changing their rows. Then to obtain the desired final records distribution, all the records are routed to column 0 without changing their rows. Hence the permutation algorithm consists of $\lceil n/k \rceil$ route-level-gather (passes) followed by a routing passes so that all N records are in the column 0.

III.2.2. The Formal Description

The algorithm makes the records in each *group-cycles* to be leveled and gathered in parallel using CCE of $N^{(1+1/h)}$ PEs. The leveling of a record in a 2^h*-group-cycles* represents the number of the records preceding it in that group. Therefore, the Procedure *LEVEL*, shown in Fig. 5, which works recursively determines this number for each record in every group of the *group-cycles*, in parallel. It divides a 2^h*-group-cycles* into two 2^{h-1}*-group-cycles*. If L(i) is the level of a record (if any) in PE(i) and stored in its register $\beta(i)$. $\Sigma(i)$ is the total number of records in the 2^{h-1}-group-cycles containing PE(i) and stored in its register $\gamma(i)$, then the level of a record in a 2^h*-group-cycles* is L(i) if $i_{h-1}=0$, (note that $i_{h-1}=0$ for the upper 2^{h-1} of a 2^h*-group-cycles*, in Fig.3 and Fig.4.), or $L(i)+\Sigma(i_{//(h-1)})$, if $i_{h-1}=1$. (Where $\Sigma(i)$ represents the total number of records in the 2^{h-1}*-group-cycles* including PE(i). L(i) is the level of the record in the register $\beta(i)$ of the PE(i) (if any) within the 2^{h-1}*-group-cycle.*) Then, unfolding the recursion yields the iterative procedure *LEVEL*. We can see that *LEVEL* uses O(h) PE time and exactly h unit routes. After leveling the records the procedure *GATHER* shown in Fig. 6, gathers records with each 2^h*-group-cycles*, such that they are moved to consecutive PEs. Let L(i) be such that the record (if any) in PE(i) is be re-located to the R(i)th PE in the 2^h*-group-cycles.* (The L(i) used here differs from the L(i) obtained by *LEVEL* by an additive constant to be determined later). *GATHER* is achieved by first re-locating all records in the *group-cycles* to PEs such that the PE indices and L(i) agree in bits 0 and 1; and so on until records have been routed to the correct PE. Fig. 5, shows the formal recursive construction of the algorithm named by the procedure *LEVEL* . Whereas, Fig. 6, shows the procedure *GATHER(h).*

The total realization of the parallel permutation is shown in Fig. 7, which shows the algorithm named as the procedure *PARALLEL-PERMUTATION(h)*, that works to accelerate the parallel setting' algorithm of Benes IN given in the next chapter. Fig. 7, represents the total realization of the parallel algorithm performed in $\lceil n/h \rceil$ passes. The routing in each pass is done in parallel at each 2^h*-group-cycles, (*note that, at each pass there is only one record per row, because of the assumption of limited parallelism). As shown in Fig. 7, at first the algorithm works to re-locate R(i) in the pass; p to column $\sigma(i)$, where $\sigma(i)=F(i({}^{h-1}{}_l))$ such that, $h=n-(p-1)k$, *and* $l=\max(h-k,0)$. Therefore the record in each row is replicated over all PEs in that row, and then deleted except for the one in the proper row.

III.2.3. Analysis

As we have seen that arbitrary permutation can be permuted in $\lceil n/h \rceil$passes. For instance, in the pass p, the record R(i) are re-located according to the digit $\sigma(i)=F(i({}^{h-1}{}_l))$, as the sort key. For the algorithm shown in Fig. 7, the block ⓐ, on the figure represents the records at the rows are duplicated over all PEs in that row. The number of unit-routes for this block is $\lceil n/k \rceil \cdot k = n1$. Also, in the block ⓑ, all copies of these records are deleted except for the proper one. The call of the procedure *LEVEL(h,k,*n), levels the 2^h-records in the 2^h*-group-cycles*. The call of the procedure *GATHER*(h,k,n) partitions the records from the 2^h*-group-cycles* into 2^l*-group-cycles*, each partition is containing records which their field differ only in the most significant l bits. The number of unit routes needed to execute the call of these two procedures have h per iterations of the for loop in the block ⓓ. Therefore substituting for h, then the unit routes needed for calling these procedures with the block ⓓ is; $(h+1)n = n3$. The block ⓒ relocates the record into the PEs of column 0 of the same row. Therefore, this block has a number of unit routes of $n-(\lceil n/k \rceil - 1)k = n2$. Then the total unit routes needed for the accelerating algorithm; *PARALLEL-PERMUTATION(*n,k,h) is $\xi(n,k) \leqq n1+n2+n3$ this enquality is because of $k \leqq n$. Substiuting with the above, we have

$\xi(n,k) \leqq n + k + (\lceil n/k \rceil + 1)n \leqq (\lceil n/k \rceil + 2)n$. Therefore, the number of unit routes is of $O(h \log_2 N)$, where $h = \lceil n/k \rceil$, if we have Φ PEs of 2^{n+k}.

IV. Parallel Setting Algorithm for Benes IN

In this chapter we will represent the main parallel version of Benes IN's realization algorithms implemented on CCE model of Chap. II, and accelerated by parallel permutation algorithm given in Chap. III.

IV.1. Notations and Characteristics

We have presented here characteristics and realizations concerning with Benes IN; as following:

1) Let $\Pi(i)=j$ represents arbitrary permutation of $\mathbb{N}$ elements such that; $i=(0,1,....\mathbb{N}-1)$, where $0 \leqq i,j < \mathbb{N}$, assuming $\mathbb{N}$ is base 2, i.e., $\mathbb{N}=2^n$. (Note that, the small brackets; (....), represents an ordered set of elements.) Therefore $\Pi(n)$ means the permutation as a function of n. For instance, $i=(0,1,2,3,4,5,6,7)$, $j=(0,2,4,6,1,3,5,7)$, which represents the perfect shuffle permutation. Hence, for $\Pi(0)=0$, $\Pi(1)=2$, $\Pi(2)=4$, and so on. Also, let the inverse of this permutation be represented as $V(\Pi(i))=V(j)=i$.

2) According to the recursive structure of Benes IN, $\Pi(n)$ is divided into two parts of subpermutations. $\Pi_{UP}(n-1)$ represents the upper subpermutation of $0 \leqq i,j < N/2$, at the input of the upper middle stage, named as $B_{UP}(n-1)$, of Benes IN; B(n). Also, the $\Pi_{LOW}(n-1)$ represents the lower subpermutation of $0 \leqq i,j < N/2$, at the input of the lower middle stage, named as $B_{LOW}(n-1)$ of the B(n) as shown in Fig.1.

3) We have also represented the required permutation; $\Pi(n)$ by an undirected graph [Na82]; $G(\Pi(n))$, in which every vertex; v represents a SW of B(n) IN, while the edges represent connections between the vertices according to the permutation required. Consequently, we have two disjoint set of vertices. One is the vertices corresponding to the input stage, and are denoted by; $V_{in}=(a_1,a_2,a_3.......a_{(N/2-1)})$, where every vertex; $a_{\lfloor i/2 \rfloor}$; corresponds to the switch $\lfloor i/2 \rfloor$ of the first stage; 0. While the other set is the vertices corresponding to the output stage, and are denoted by; $V_{out}=(z_1,z_2,z_3....z_{N/2-1})$, where every vertex $z_{\lfloor j/2 \rfloor}$; corresponds to a switch $\lfloor j/2 \rfloor$ at the last stage; $2n-1$.

4) From the bipartite multigraph we can find the subpermutation named as; *complete-subpermutation-group*, which represents a cyclic path from vertex $\lfloor i/2 \rfloor$ with its edge connected to other disjoint vertex and so on till it return back to the same vertex $\lfloor i/2 \rfloor$. Knowing these *subpermutation-groups* we can find in parallel the switching states of the first and last stages, besides, the permutation of middle upper stages $\Pi_{UP}(n-1)$, and the permutation of middle lower stages $\Pi_{LOW}(n-1)$, of Benes IN.

5) Let us define $\Delta(\omega,\varepsilon)$, as the function's state of the SW; ε of the stage; ω. Hence, the parameter ω corresponds to the stage number, where $0 \leqq \omega < 2n-2$, and the parameter ε corresponds to the switch ε, where $0 \leqq \varepsilon < N/2$. $\Delta(\omega,\varepsilon)$ has only one of the two values, which represents either 0, or 1. The value 0; represents the setting of that SW to the straight connection, while the value; 1 represents the setting of that SW to the cross connection, Fig.1.

6) Let $H_{UP}(i/2)$ represents a subpermutation of $\Pi(i)$ which come from the output of the upper middle stage; $B_{UP}(n-1)$ of B(n). For example for the perfect shuffle permutation, the $H_{UP}(i/2)=(0,3,4,7)$ which represents a subpermutation of the output permutation $\Pi(i)$, which come from the output of the $B_{UP}(n-1)$; see Fig. 1. Please note that $H_{UP}(i/2)$ has the following characteristics, which are related to B(n) construction: $0 \in H_{UP}(i/2)$, $|H_{UP}(i/2)|=N/2$, and if $i_1, i_2 \in H_{UP}(i/2)$, where $i_1 \neq i_2$, then $i_1 \neq (i_2)_{//0}$, and $V(i_1) \neq (V(i_2))_{//0}$. However, the set $H_{UP}(i)$ defines a complete matching on the bipartite graph, $G(\Pi(i))$. In fact the set $H_{UP}(i/2)$, give us enough useful information about the switch settings for stages 0 and $2n-2$, $\Pi_{UP}(n-1)$, and $\Pi_{LOW}(n-1)$ using the following rules: If $i \in H_{UP}(i/2)$ then; ① $\Delta(2n-2, i/2) = i_0$, ② $\Delta(0,V(i)/2) = (V(i))_0$, ③ $\Pi_{UP}(V(i)/2)) = i/2$, ④ $\Pi_{LOW}(N/2+(V(i)/2)) = N/2 + \Pi((V(i)_{//0})/2)$. (please note that all divisions are integer division, i.e., $\lfloor i/2 \rfloor$).

IV.2. The Construction of The Parallel Algorithm for Benes IN

We have seen that the set $H_{UP}(i/2)$, are useful to find the Benes IN's SWs' setting. In result, we would like to find a fast parallel method to find $H_{UP}(i/2)$.

A) The first step in the parallel algorithm, is to detect the *complete-subpermutation-group*, which is defined above as a cyclic path from and into again a certain vertex. (IV.1 the realization 4.). Therefore the procedure; *SUBPERMUTATION-GROUP* shown in Fig. 8, performs this task by finding the *subpermutation-groups*; which represents the components of the *complete-subpermutation-group*. Please note that, if C represent a certain *subpermutation-group* resulted from the execution of procedure *SUBPERMUTATION-GROUP*, then C has the following characteristics: ① If $i \in C$, then $i_{//0} \notin C$. ② If C_1, and C_2 are *subpermutation-groups*, such that; $i \in C_1$, and $i_{//0} \in C_2$, then $C_1 + C_2$, represents a *complete-subpermutation-group* of the permutation graph; $G(\Pi(i))$. ③ $|C_1|=|C_2|$ if $i \in C_1$, then $i_{//0} \in C_2$ and $i \in C_2$, then $i_{//0} \in C_1$. Note that this algorithm works on the computational model presentd in Chap. II, where PEs in a *2^h-group-cycles*, are linked together using a field $F(i)$, (recall

F(i) is the link field in PE(i)). As has been presented in Sec.II.2, an arbitrary permutation is, initially distributed over Φ PEs, such that every record is in one cycle of the CCE model. Therefore, the record j is in PE(i) of the cycle; i, and $0 \leqq i < N$ of the CCE. If $j \in \Pi(i)$ then for some $j' \in \Pi(i)$ such that; $j' = (\Pi((V(j))_{//0})$, note that $V(\Pi(i)) = V(j) = i$. Therefore, applying the algorithm of Fig. 8, we have for instance, j and j' be in different *subpermutation-groups*, and j and $(j')_{//0}$ are in the same *subpermutation-group*.

The time complexity of *SUBPERMUTATION-GROUP* algorithm is of O(k), which represents the prallel computation time within the cycles of 2^k PEs of the CCE. The *subpermutation-groups* for the perfect-shuffle are; (0,3), (4,7), (1,2), (5,6).

B) The second step is to find $H_{UP}(i/2)$ and $H_{LOW}(i/2)$ from the *subpermutation-groups*, which are determined by the procedure *SUBPERMUTATION-GROUP*, in parallel. Here, these groups are localized to 2^h-*group-cycles*, (defined in Sec.III.1). Therefore, the procedure *DIMINUTION*(h)'s algorithm shown in Fig. 9 , diminute the permutation $\Pi(i)$ represented by the *subpermutation-groups*, such that to determine D(i); which represents the minimum integer in the *subpermutation-group* (containing i, $0 \leqq i < N$). This procedure takes in use the direct connection of CCE. Hence, after the iteration ϕ of the for loop, the register β which contains the field F(i) points to the register β of other PE at distance $2^{\Phi+1}$, (please note that, the distance is measured along the *subpermutation-group*, mapped on the cycles of CCE (i.e., the 2^h-*group-cycles*)). Therefore, completing all the iterations till $\phi = h-1$, we may have D(i), be as the minimum integer of the *complete-subpermutation-group* that also contain i. Using the same example of the perfect shuffle permutation, the set of *subpermutation-group*, will be mapped onto 2^{h-1}-*group-cycles*, of CCE, such that every *subpermutation-group* is on one 2^{h-1}-*group-cycles*. Due to Chap.III, we have (for our example) four groups of *group-cycles*, such that every *subpermutation-group* is mapped on one *group-cycle*. Operate all groups of the *group-cycle*, in parallel to determine $H_{UP}(i/2) = (0,3,4,7)$. The execution time is proportional to the partitioning time of the *group-cycle, i.e., O(h)*.

C) This is the complete algorithm named as the procedure *PARALLEL-MAP-SETTING*(n,h,k), which calls the procedures presented above as A and B, as shown in Fig. 10. This procedure represents the control mechanism for the CU of Benes IN to set its SWs for arbitrary permutation in time of O(h $\log_2$N). At each iteration the setting of the first and last stages of all B(h) Benes subnetworks of B(n) is determined in parallel. Thus, in the first iteration(i.e., h=n) the SWs' setting for stages 0 and 2n−2, according to the rules ①, and ② (presented at the end of sec.IV.1) as well as the $\Pi_{UP}(n-1)$, and $\Pi_{LOW}(n-1)$, are determined in parallel. In the next iteration, (i.e.,$h=n-1$) the SWs 'setting for stages 1 and 2n−3, as well as the $\Pi_{UP}(n-2)$, and $\Pi_{LOW}(n-2)$, (according to the rules ③, and ④) are determined in parallel, and so on. Hence, the parameter h which specify the size of the 2^h-*group-cycles* dominate the size of the *group-cycles* on which the permutation is mapped.

Please note that the statement denoted on Fig. 10, represents how to achieve the rule ③ and ④ of sec.IV.1. Whereas a division by 2 requires us to shift bits $h-1$,...1 one position right and define the new bit ($h-1$) to be zero. Also, adding 2^{h-1} requires changing bit $h-1$ to 1. Therefore, statement ⓐ and ⓑ on Fig. 11, implement the rules ③ and ④ for each B(h) network. If the least element in a *complete-subpermutation-group*, is even then all elements in that class must be routed through the upper B($h-1$) network. The statements ⓒ is executed on PEs whose register $\beta(i)$ is zero. In this statement the SW's settings for the first stage of all B(h) networks are determined by the rule ②. When $h=1$, then B(h) networks have only one stage, and then the statement ⓓ terminates the operation. But when $h \neq 1$, the SWs settings for the last stage of all B(h) networks are determined by the statement ⓔ.

V.Conclusions

The construction of the control unit of Benes interconnection network, appears to be highly sequential in nature, through the controlling algorithms presented by [Ca87], but it can be parallelized using efficient model like the CCE model presented in this paper. Therefore the time complexity for setting the SWs of Benes IN becomes comparable with the propagation gate delay, i.e., O(log N), which represents the lower setting bound. This bound could be achieved by using an accelerator which can realize the parallel permutation problem in time of O($\log_2$N), on the CCE parallel processor model of $\Phi = 2^{n+k}$ PEs, where $2^n = N$, and 2^k represents the number of PEs per cycle where $\Phi > N$.

REFERENCES:

[Ca87]Carpinelli, J.D., Oruc, A.Y.,"Parallel set-up algorithms for Clos networks using a tree-connected computer," *Second Int. Conf. on Supercomputing*, pp.321-327, May, 1987.

[Ha87] Hamid, I.A., et.al.,"A new fast control mechanism for rearrangeable interconnection network useful for supersystems," *Tran. IEICE*, vol. E70 No. 10, Oct. 1987, pp. 997-1008.

[Na82] Nassimi,D., et.al., "Parallel algorithms to setup the Benes permutation network," *IEEE Trans. Comput.* vol. C-31, pp. 148-154, Feb., 1982.

[Pr81] Preparata, F.P., Vuillemin, J.,"The cube-connected cycles: A versatile network for parallel computation," *CACM*, pp. 300-309, May, 1981

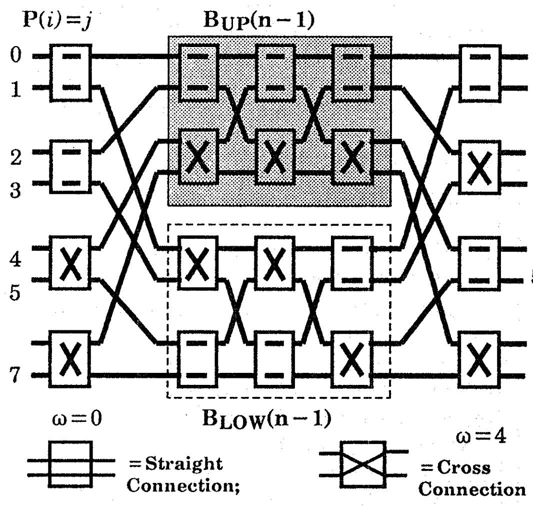

Fig. 1, Benes Interconnection Network; (B(n=3)), for N=8.

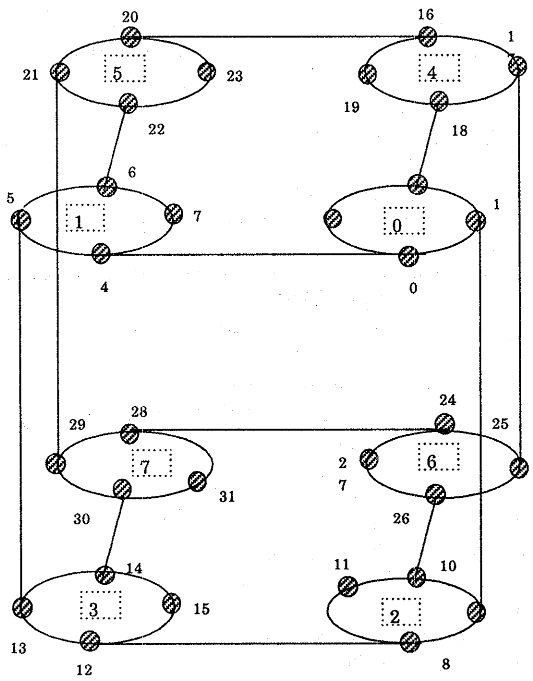

Fig. 2, The structure of the Cyclic Cube Engine computational model.

	0	*1*	*2*	*3*
0	0 00000	1 00001	2 00010	3 00011
1	4 00100	5 00101	6 00110	7 00111
2	8 01000	9 01001	10 01010	11 01011
3	12 01100	13 01101	14 01110	15 01111
4	16 10000	17 10001	18 10010	19 10011
5	20 10100	21 10101	22 10110	23 10111
6	24 11000	25 11001	26 11010	27 11011
7	28 11100	29 11101	30 11110	31 11111

Fig.3, The addresses of the CCE's PEsrepresented as two dimensional table.

	0	*1*	*2*	*3*
0	5			
1	2			
2	1			
3	4			
4	0			
5	7			
6	6			
7	3			

	0	*1*	*2*	*3*
0	1			
1	0			
2		2		
3		3		
4			5	
5			4	
6				7
7				6

	0	*1*	*2*	*3*
0			5	
1		2		
2	1			
3			4	
4	0			
5				7
6				6
7		3		

	0	*1*	*2*	*3*
0		1		
1	0			
2	2			
3		3		
4		5		
5	4			
6		7		
7	6			

	0	*1*	*2*	*3*
0	0			
1		1		
2	2			
3		3		
4	4			
5		5		
6	6			
7		7		

Fig. 4, An example of the parallel permutation algorithm using the CCE.

```
procedure LEVEL(h,k,n);
comment  make the records for each group of PEs. Recall
         that, $\beta(i)$, $\gamma(i)$, and $\mu(i)$ represent the registers of
         PE($i$), such that, L($i$) in register $\beta(i)$, $\Sigma(i)$ in $\gamma(i)$;
global integer array F($i$), L($i$), $\Sigma(i)$;
$\beta(L(i)) := 0$;
if F($i$)$\neq$ null then $\gamma(\Sigma(i)) = 1$ else $\gamma(\Sigma(i)) = 0$;
do foreach $\phi := h$ to $n+k-h$;
begin
    pardo $\mu(i_{//\phi}) \leftarrow \gamma(\Sigma(i))$;
    if $i_{//\phi} = 1$ then $\beta(L(i)) := \beta(L(i)) + \mu(i)$;
    $\gamma(\Sigma(i)) := \gamma(\Sigma(i)) + \mu(i)$;
    odpar;
end;
end LEVEL;
```

Fig. 5, The algorithm for the procedure *LEVEL*(n,*k*,*h*).

```
Procedure GATHER(h,k,n)
comment  Accpording to L($i$) determined by procedure
         LEVEL, relocate the records R($i$) to the PEs;
global integer array R($i$), L($i$);
comment  F($i$) is the field of the record R($i$);
do foreach $\phi = h$ to $n+k-h$
    if F($i$)$\neq$ null, and $\beta(L(i)) \neq i_{\phi}$ then
        begin
            pardo $(R(i_{//\phi})), \beta(L(i_{//\phi}))) \leftarrow (R(i), L(i))$;
            odpar;
        end;
end GATHER;
```

Fig. 6, The algorithm for the procedure *GATHER*(n,*k*,*h*).

```
procedure PARALLEL-PERMUTATION(n,k,h);
comment  On CCE of $2^{n+k}$ PEs, the parallel permutation of
         $2^n$ records, according to the field $F(i)$ of the record
         $R(i)$, can be done;
global integer array  $R(i)$, $L(i)$;
begin
    if $i(^k{}_0) \neq 0$ then $F(i) := null$;
comment initialize the remaining columns;
    $h := n$;
comment the number of column in a $2^h$-group-cycles, is $2^n$;
        for $p = 1$ until $\lceil n/k \rceil$;                              ⓓ
comment $p$ is a pass of $\lceil n/k \rceil$ passes of radix sort algorithm;
            pardo for each $\phi = 0$ to $k$;                       ⓐ
comment copy records over columns;
                pardo if $F(i) \neq null$ then $R(i_{//\phi}) \leftarrow R(i)$;
                odpar;
            odpar;
    $l := max(h-k, 0)$;
comment bits, $h-1, \ldots, l$ form the digit; ⓑ
    if $\lfloor i/2n \rfloor \neq F(i(^{h-1}{}_l))$ then
    $F(i) := null$  else  $F(i) := F(i)$;
    call LEVEL(h,k,n);
    $\beta(L(i)) := \beta(L(i)) + \lfloor i/2^n \rfloor \cdot 2^l$
    call GATHER(h,k,n);
    $g := h$; $h = l$;
comment Each partition is $2^l$-group-cycles columns;
end;
comment relocate records to the first column
begin
    pardo for each $\phi = 0$ to $k-1$;
        if $F(i) \neq null$ then $R(i_{//\phi}) \leftarrow R(i)$;   ⓒ
    odpar;
    end;
end;
end PARALLEL-PERMUTATION
```

Fig. 7, The algorithm for the procedure *PARALLEL-PERMUTATION*(n,*k*,*h*).

```
procedure SUBPERMUTATION-GROUP;
comment  find the subpermutations which represent the
         subpermutation-group.
global integer array  $F(i)$, $\Pi(i)$, $V(i)$;
$\beta(F(i)) := (P(i))_{//0}$;
comment  assign the register $\beta(i)$ which contains the field
         $F(i)$ with the value $(\Pi(i))_{//0}$;
$\beta(F(i_{//0})) \leftarrow \beta(F(i))$;
call PARALLEL-PERMUTATION(n,h,k)
$\langle V(P(i)), \beta(F(P(i))) \rangle \leftarrow \langle i, \beta(F(i)) \rangle$;
end SUBPERMUTATION-GROUP
```

Fig. 8, The algorithm for the procedure *SUBPERMUTATION-GROUP*.

```
procedure PARALLEL-MAP-SETTING(n,h,k);
commnet parallel algorithm for setting Benes IN;
global integer array   $\Pi(0{:}N-1)$, $D(0{:}N-1)$, $V(0{:}N-1)$,
                       $\Delta(0{:}2n-2, 0{:}(N/2)-1)$;
begin;
REPEAT: for $h := n$ to 1 step $-1$
    begin;
    $\omega := n - h$;
call SUBPERMUTATION(h);
call DIMINUTION(h);
comment  for permutation $\Pi(i) = j$, if $j \in H_{UP}(\lfloor i/2 \rfloor)$, then
$D(i) := 0$;
    $D(i) := D(i_{//0})$;
call PARALLEL-PERMUTATION(h);
comment this means $\beta(V(i)) \leftarrow D(i)$;
$\langle \beta(V(i)), D(i) \rangle \leftarrow \langle V(i), D(i) \rangle$;
comment  setting of the first stage;
    if $\beta(i) = 0$ then $\omega = i_0$ and $\Delta(\omega, \lfloor i/2 \rfloor) = i_0$ else   ⓒ
$\Delta(\omega, \lfloor i/2 \rfloor) := \Delta(\infty, \infty)$;
comment  setting of the first stage;                 ⓓ
    if $h = 1$ then exit;
comment assign the last stage;
    if $D(i) = 0$ then $\gamma(\Delta(2n-2-\omega, \lfloor i/2 \rfloor)) \leftarrow i_0$;   ⓔ
ccomment  Updates $\Pi(i)$ such that it corresponds to the
          $\Pi_{UP}(h-1)$ if it is even;
    if $\beta(i) \neq i_0$ then $\mu(\Pi(i_{//0}) \leftarrow \mu(\Pi(i))$;
    $\mu(\Pi)(i_{n-1}, \ldots i_h i_0 i_{h-1}, \ldots i_1)) \leftarrow \mu(\Pi(i))$;   ⓐ
comment   routes the $\Pi(i)$ to PEs whose index
          correspondsto the rules ③, and ④;   ⓑ
    $\Pi(i) := i(^{n-1}{}_{k-1})(D(i(^{h-1}{}_1)))$;
    end;
goto REPEAT;
end PARALLEL-MAP-SETTING
```

Fig. 10, The algorithm for the procedure *PARALLEL-MAP-SETTING*(*h*,*k*,n).

```
procedure DIMINUTION(h);
comment   this procedure diminute the permutation $\Pi(i)$ to
          find out the minimum element in every
          subpermutation;
global interger array $F(i)$, $D(i)$; integer array $\mu(i)$, $\gamma(i)$;
begin;
    $D(i) := i$;
    parado foreach $\phi = 0$ to $h-1$
comment  works in parallel on all $2^{h-1}$-group-cycles;
    $\gamma(i_{//0}) \leftarrow \beta(F(i))_{//0}$;
comment   this assign $F(i) = j$, during iteration $\phi = 0$,
          $F(j_{//0}) = i_{//0}$, therefore $\gamma(F(j_{//0})) = j_{//0}$;
call PARALLEL-PERMUTATION(h)
    $\langle \mu(\gamma(i)), \beta(F(\gamma(i))) \rangle \leftarrow \langle D(i), \beta(F(i)) \rangle$;
    $D(i) := min\{ D(i), \mu(i) \}$;
    odpar;
end DIMINUTION
```

Fig. 9, The algorithm for the procedure *DIMINUTION*(*h*)