상호 연결망이 단순화된 Batcher의 정렬망과 ATM 교환 시스템에서의 응용

이 재 동†

요 약

본 논문에서는 상호 연결망을 단순화시킨 Batcher의 정렬망(sorter)을 설계하고 ATM 교환 시스템에서 그것의 응용에 대하여 살펴본다. 많은 ATM 교환기에서 Batcher의 정렬망을 사용함으로써 조정회로의 구조나 전달망 구조의 설계를 단순화 할 수 있다. 알고리즘 CONSTRUCT-BSS를 설계할 수 있게 하는 패리티 전략을 소개하였다. 내부 상호 연결을 단순화하기 위하여 N/2개의 짝수 패리티 key들을 정렬망(sorter)에서 직선으로 연결하였다. 결과적으로, 이 논문에서 제안된 상호 연결 방법은 Batcher 정렬망의 내부 상호 연결을 단순화하였고 perfect-shuffle 상호 연결망과 비교하여 하드웨어 가격이나 속도 면에서 우위를 점한다. 또한, 여기서 제안한 정렬망은 전달 경로의 반이 직선으로 설계되므로 도안된 회로 보드나 VLSI 칩의 레이아웃이 보다 단순화될 수 있다.

BSS: Batcher's sorter with simpler interconnections and its applications for ATM switching

Jae-Dong Lee†

ABSTRACT

This paper presents the design of a Batcher's sorter with simpler interconnections between levels and its applications for ATM switching systems. Many ATM switches use the Batcher's sorter in order for simplifying the design of an arbitration circuit structure and a router. A parity strategy which leads to the algorithm CONSTRUCT-BSS is introduced. For simplifying inter-level wiring, N/2 even parity keys travel straight through the sorter. As a result, the proposed interconnection scheme simplifies the inter-level wiring through the Batcher's sorter and outperforms the perfect-shuffle interconnection scheme both in terms of cost and delay. The layout of this proposed sorter on a printed circuit board or a VLSI chip may be simpler since half of the routes are straight lines.

1. Introduction

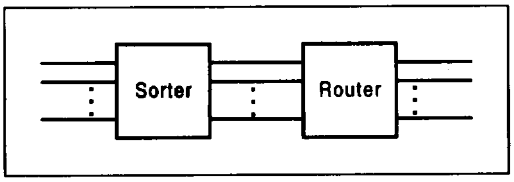
Asynchronous transfer mode(ATM) is the transport technique for the broadband ISDN recommended by CCITT[1]. Many switches have been proposed to accommodate the ATM, which requires fast packet switching [7,8,11]. A variety of switching fabrics have been proposed to implement an ATM switch, most of which are based on Banyan-type fabrics[2,5,6,7]. The Banyan networks that are based on multi-stage interconnection networks have features such as self-routing capabilities, cost-

※ This work was supported in part by DanKook University.
† 정 회 원 : 단국대학교 전자계산학과 교수
논문접수 : 1998년 1월 31일, 심사완료 : 1998년 5월 25일

effectiveness, low-delay and suitable for VLSI implementation[7]. These features have made Banyan networks a suitable candidate for implementing ATM switches. But Banyan networks are the internally blocking networks. In order to prevent the Banyan network from the internal blocking a sorter(or sorting network) is used. The most frequently used sorter is the Batcher's sorter[3,5,10].

As shown in Figure 1, a sorter is used to sort the cells which are transported to the output ports through a router in ATM switching systems. The Batcher's sorter which supports fast sorting capabilities and solves communication problems was introduced in 1968 [3,14]. Many ATM switches such as Starlite, Moonshine, SXmin and Sunshine use the Batcher's sorter which can be applied to simplify the design of an arbitration circuit structure and a router. The Sunshine network [6,11] shown in Figure 2 is an example of a Batcher-banyan network. Here, a trap considers the sorted cells and the second sorter separates the winners from the losers and sorts the losers by priority. A selector routes the highest-priority losers to the recirculation ports and passes the winners to the banyan network. A sorting circuit is used to sort the cells by destination address.

One difficulty in the design of an ATM switch based on Batcher-banyan configuration is the interconnection of the larger number of comparators(or switching elements) and links in order to accommodate the required cells transfers[3-6,14]. Batcher-banyan networks of significant size are physically limited by the possible circuit density and number of I/O pins of the integrated circuit. To interconnect the large number of comparators and links on the circuit board, interconnection complexity is an important factor for the circuit board design[2,5].



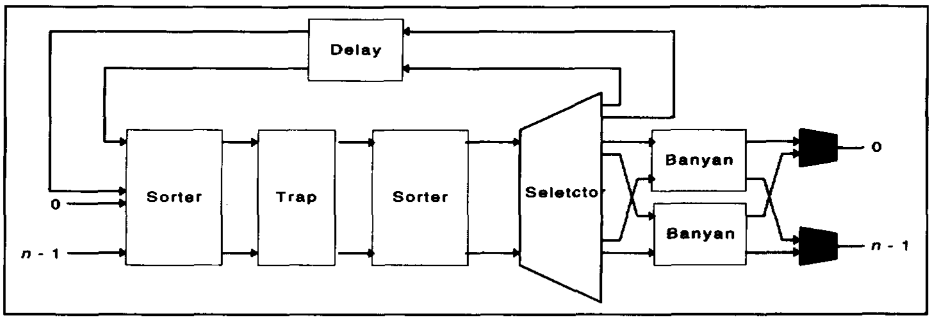
(Fig. 1) A Sorter-router ATM Switching System

The Batcher's Sorter with Simpler interconnections(BSS) will be presented here. The order of the comparators in each level can be arranged such that the inter-level wiring is simplified. For this purpose, the parity strategy is applied which results in straight network wires connecting the $N/2$ even-parity keys. The layout of this proposed sorter on a printed circuit board or a VLSI chip may be simpler since half of the routes are straight lines. Furthermore, compared with the perfect -shuffle interconnection scheme, a performance improvement is achieved by reducing the cost and time(or delay) by approximately one half. Throughout this paper, the terms a sorter and a (sorting) network, time and delay will be used interchangeably.

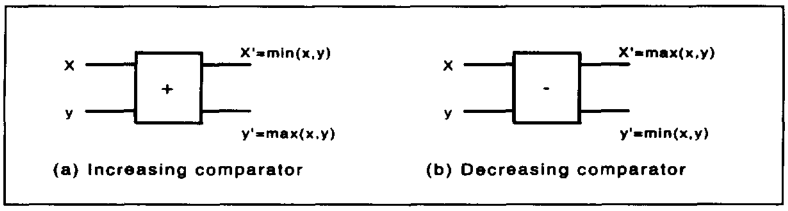
The remaining of this paper is organized as follows. Section 2 describes the basic description of the Batcher's sorter. Section 3 describes the basic approach for designing the Batcher's sorter with simpler interconnections, discusses the topology rules, introduces the algorithm CONSTRUCT-BSS which shows the interconnections scheme between stages and shows some examples. Section 4 shows the performance of the proposed sorter in terms of cost and delay. Section 5 presents the applications of the proposed sorter. Finally, section 6 states some conclusions.

2. Batcher's sorter

This section describes basic definitions



(Fig. 2) Sunshine network



(Fig. 3) A simple 2x2 comparator type.

which are used in this paper. The key component of the Batcher's sorter is a comparison-exchange element (or a *comparator*, for short), as shown in Figure 3. Two input keys, x and y, are compared and output in ascending order with min(x,y) output to x', and max(x,y) output to y'; for a decreasing comparator x' = max{x,y} and y' = min{x,y}. Throughout this paper, the uppercase letter N denotes a power of two, lowercase letter k denotes log$N$ and all logarithms are base two.

An ascending sequence, represented as /-sequence, is a non-decreasing sequence of keys $\{x_0, x_1, x_2, \ldots\ldots, x_{N-1}\}$ such that $x_0 \le x_1 \le \ldots\ldots \le x_{N-1}$. A descending sequence, represented as \-sequence, is a non-increasing sequence of keys $\{x_0, x_1, x_2, \ldots\ldots, x_{N-1}\}$ such that $x_0 \ge x_1 \ge \ldots\ldots \ge x_{N-1}$. Note that a single key, x, is both an /-sequence and a \-sequence.

An *ascending-descending* sequence (or ∧-sequence) is an /-sequence foliow by a \-sequence of key, $\{x_0, x_1, x_2, \ldots\ldots, x_i, x_{i+1}, \ldots, x_{N-1}\}$ such that $x_0 \le x_1 \le \ldots\ldots \le x_i \ge x_{i+1} \ge \ldots \ge x_{N-1}$.

Note that any /-sequence is an ∧-sequence. Similarly, a *descending-ascending* sequence (or ∨-sequence) is a \-sequence followed by an /-sequence.

A ***bitonic sequence*** is a sequence of keys $\{x_0, x_1, \ldots\ldots, x_{N-1}\}$ with the property that (a) *there exists an index i,* $0 \le i \le N-1$*, such that* $x_0 \le x_1 \le \ldots\ldots \le x_i \ge x_{i+1} \ge \ldots \ge x_{N-1}$ *or* (b) *there exists a cyclic shift of indices so that* (a) *is satisfied.*

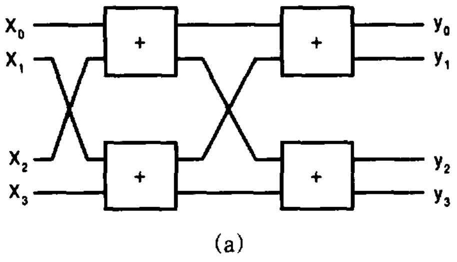
For example, {0 2 3 5 7 9} is a bitonic sequence since the sequence {0 2 3 5 7 9} may be defined as the /-sequence and { null } as the \-sequence. The sequence {0 1 3 5 7 9 8 6 4 2} is also a bitonic sequence, because it first increases and then decreases. Similarly, {3 5 7 9 8 6 1 0 2} is another bitonic sequence, because it is a cyclic shift of {0 2 3 5 7 9 8 6 1}. However, the sequence {0 2 3 7 6 4 8 5 1} is not a bitonic sequence since this sequence cannot be an ∧-sequence by the rotation. Notice also that, any sequence of three or less keys is bitonic and any subsequence of a bitonic sequence is also bitonic.

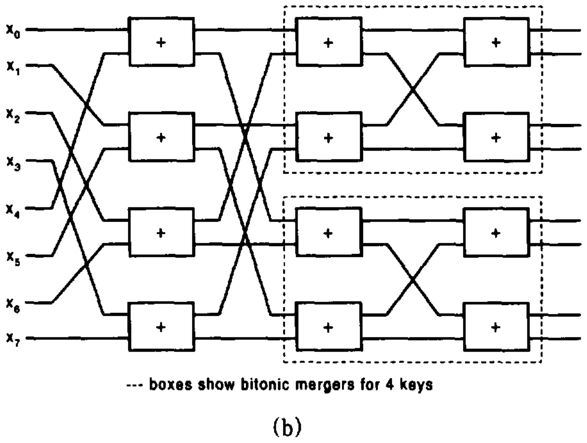
The definition of a Batcher's merger is a comparison network which accepts a bitonic sequence as input and produces a sorted sequence as output. A simple 2 x 2 comparator is the smallest Batcher's merger. Figure 4 shows the Batcher's mergers for 4 keys and 8 keys, respectively. The Batcher's sorter which sorts an arbitrary sequence is constructed from a series of the Batcher's mergers. Figure 5 shows the Batcher's sorter for 8 keys. The following theorem gives us a rule which is an essential method to rearrange a bitonic sequence into a sorted sequence in the Batcher's sorter.

Theorem 1: Let $X=\{x_0, x_1, \ldots\ldots, x_{N-1}\}$ be bitonic. If $B=\{b_0, \ldots\ldots, b_i, \ldots\ldots, b_{N/2-1}\}$ where $b_i=\min(x_i, x_{N/2+i})$ and $C=\{c_0, \ldots\ldots, c_i, \ldots\ldots, c_{N/2-1}\}$ where $c_i = \max(x_i, x_{N/2+i})$ for $0 \le i \le N/2-1$, then (1) both B and C are bitonic and (2) $\max\{b_0, b_1, \ldots\ldots, b_{N/2-1}\} \le \min\{c_0, c_1, \ldots\ldots, c_{N/2-1}\}$. (See [3])

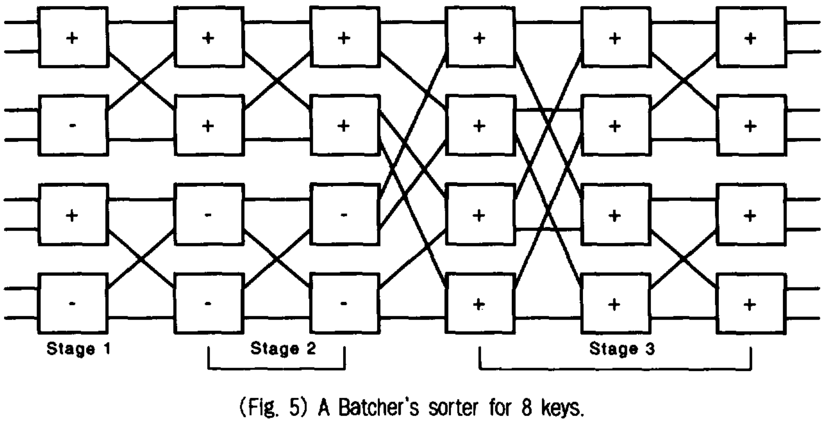
3. Batcher's sorter with simpler interconnections

In this section, the Batcher's sorter with simpler interconnections (BSS) is presented.





(Fig. 4) Batcher's merger for (a) 4 keys and (b) 8 keys



(Fig. 5) A Batcher's sorter for 8 keys.

3.1 The basic approach

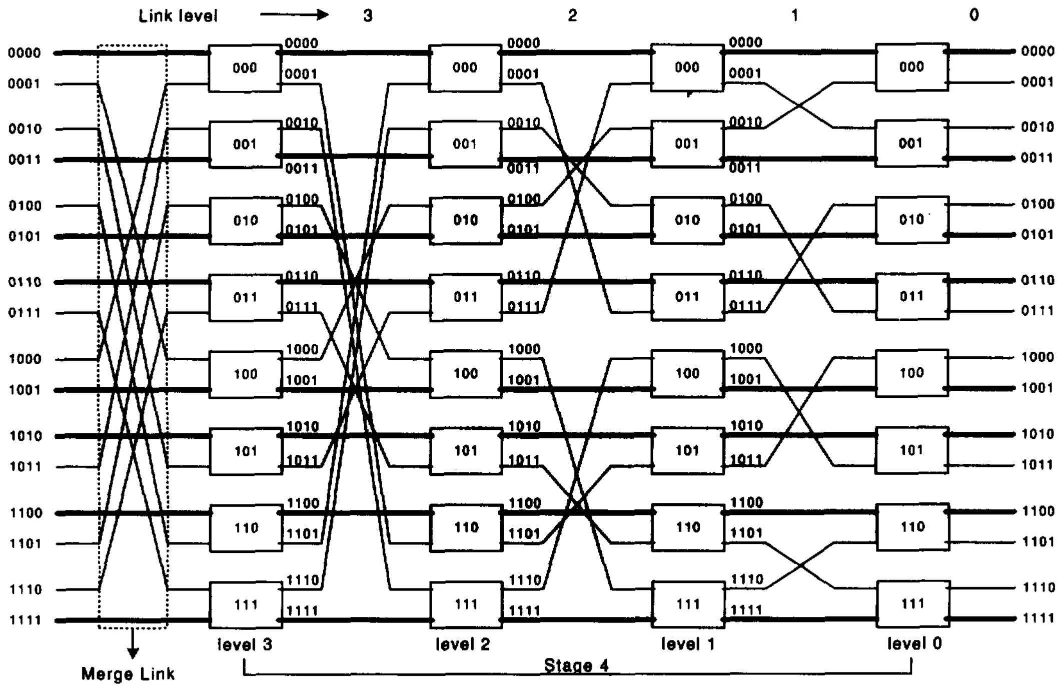
The basic idea employed here is that if $N/2$ even-parity keys are wired straight through the sorter, then only $N/2$ odd-parity keys are considered for inter-level wiring. This scheme leads to the algorithm CONSTRUCT-BSS to construct the BSS. In the following, the parity-strategy which leads to the algorithm CONSTRUCT-BSS is illustrated.

Parity Strategy : Let the parity of a key be defined by the number of 1-bits in its number; if the index has an even number of 1-bits then the key has even-parity (e.g., 0101); if the index has an odd number of 1-bits then the key has odd-parity (e.g., 1011). Note that in each paring of the keys during the bitonic sorting each comparator compares an even-parity key with an odd-parity key. The inter-level wiring is simplified by letting $N/2$ even-parity keys wire straight through the sorter and just wire the $N/2$ odd-parity keys through the sorter.

3.2 Batcher's merger with simpler interconnections

This section describes the connectivities of the interconnection paths between comparators using a set of mathematical rules, called *topology rules*, derived from the structure definition of the network. A physical name is given to each component (comparator, link, stage, level) for identifying its relative location in the network.

A Batcher's Merger with Simpler interconnections(BMS) accepts a bitonic sequence as input and produces a sorted sequence as output. There are $\log N(=k)$ levels of comparators and $\log N+1$ levels of links including the merge link. The BMS for 16 keys is illustrated in Figure 6, where thick lines show the even-parity keys which travel



(Fig. 6) A BMS with name presentation for 16 keys.

straight through the network.

The physical names are assigned as follows. The levels of comparators are labeled in a sequence from (k-1) to 0 with 0 for the output level and (k-1) for the input level. Similarly, the levels of links are labeled in a sequence from (k-1) to 0 excluding the merge link level. In each level, each comparator is indexed by a codeword of (k-1) binary bits, b_{k-1}, b_{k-2}, ..., b_1, which is the binary representation of its location in the level. Each link in each level is indexed by a codeword of k binary bits, b_{k-1}, b_{k-2}, ..., b_1, b_0, which is coded according to the following scheme. The first (k - 1) leftmost bits, b_{k-1}, b_{k-2}, ..., b_1, are the same as the binary representation of the comparator to which the link is connected to one of the two output keys of the comparator; the last bit, b_0, is equal to 0 if the link is connected to an upper key of the comparator and b_0 is equal to 1 if the link is connected to a lower key (see Figure 6). Throughout the rest of this paper, the binary representation of physical names of a comparator in level i and a link in level i are written as (b_{k-1}, b_{k-2}, ..., b_1) and $(b_{k-1}, b_{k-2}, ..., b_1, b_0)_i$, respectively.

The following topology rules of the BMS describe the connectivities of the interconnection paths between comparators. The procedure CONNECT-LEVEL illustrates the interconnection pattern between level d and level (d-1) in a stage and the procedure MERGE-STAGE describes the merge interconnection pattern between the column i of stage and the column (i+1) of stage$_{r+1}$. The function FPARITY takes the binary representation of the physical names of a comparator in level i as an input and returns 1 for an odd-parity comparator and return 0 for an even-parity comparator. The function C_i^p describe the interconnections by mapping a comparator in level i to two comparators in level (i-1). The function M_i^p describe the interconnections for a merge between stages by mapping a comparator in the output column i of stage to two comparators in the input column (i+1) of stage$_{r+1}$, one key per link. $\overline{b_i}$ and $\bar{p}$ represents the complements of the b_i and p, respectively.

```
CONNECT-LEVEL($d$)
For all comparators $(b_{k-1}, b_{k-2}, ..., b_1) = (0, 0, ....., 0)$ to $(1, 1, ..., 1)$ in parallel.
1  Obtain the parity value : $p \leftarrow$ FPARITY $[(b_{k-1}, b_{k-2}, ..., b_1)_d]$
2  $C_d^p[(b_{k-1}, b_{k-2}, ..., b_1)_d] = (b_{k-1}, b_{k-2}, ..., b_1)_{d-1}$, for link $(b_{k-1}, b_{k-2}, ..., b_1)_d$
3  $C_d^{\bar{p}}[(b_{k-1}, b_{k-2}, ..., b_1)_d] =$
     $\begin{cases} (b_{k-1}, ..., \bar{b}_d, \bar{b}_{d-1}, ..., b_1)_{d-1} & \text{if } 2 \le d < k \\ \text{for link } (b_{k-1}, b_{k-2}, ..., b_1, \bar{p})_d & \\ (b_{k-1}, b_{k-2}, ..., \bar{b}_1)_{d-1} & \text{if } d = 1 \\ \text{for link } (b_{k-1}, b_{k-2}, ..., b_1, \bar{p})_d & \end{cases}$
```

CONNECT-LEVEL illustrates how interconnection paths are established for all comparators to simplify inter-level wiring in a stage (see Figure 6). The parity value (0 or 1) of each comparator in level d is obtained in line 1. A straight link connection between the comparator of level d and the comparator of level (d-1) is illustrated in line 2. A non-straight connection between two levels is illustrated in line 3.

```
MERGE-STAGE($r$)
For all comparators $(b_{k-1}, b_{k-2}, ..., b_1) = (0, 0, ..., 0)$ to $(1, 1, ..., 1)$ in parallel.
1  Obtain the column number : $i \leftarrow 2^{r1} + r - 1$, for $1 \le i \le \frac{k(k+1)}{2}$
2  Obtain the parity value : p $\leftarrow$
```

```
  FPARITY[$(b_{k-1}, b_{k-2}, \ldots, b_1)_i$]
3 $M_i^p[(b_{k-1}, b_{k-2}, \ldots, b_1)_i] = (b_{k-1}, b_{k-2}, \ldots, b_1)_{i+1}$, for link $(b_{k-1}, b_{k-2}, \ldots, b_1, p)_i$
4 $M_i^{\bar{p}}[(b_{k-1}, b_{k-2}, \ldots, b_1)_i] = (b_{k-1}, b_{k-2}, \ldots, b_1)_{i+1}$, for link $(b_{k-1}, b_{k-2}, \ldots, b_1, \bar{p})_i$
```

MERGE-STAGE illustrates how merge connections are established between two stages (see Figure 7). It obtains the column number i which is the last comparator level of stage at line 1. The parity value is obtained using the function FPARITY st line 2. Line 3 and 4 determine the straight link connection and the non-straight link connection, respectively.

3.3 The Construction of a BSS

In this section, the Batcher's sorter with simpler interconnections (BSS) is considered, in which the inter-level wiring is simplified by letting the even-parity keys are wired straight through the sorter.

Setting comparator type

The procedure COMPARATOR-TYPE uses two flags ($OFLAG_{cid}$ and $EFLAG_{cid}$) to determine each comparator type at each level. A comparator acts as an increasing comparator if it is set to 0 from COMPARATOR-TYPE (i.e., $TYPE_{cid} = 0$), while it acts as a decreasing comparator if it is set to 1. CID is the comparator ID. The function CPARITY takes CID as an input and returns 1 if CID has an odd-parity (or 0 for an even-parity). $EIDX_{cid}$ and $OIDX_{cid}$ are used to set $EFLAG_{cid}$, in which $EIDX_{cid}$ represents an incoming even-parity key index and $OIDX_{cid}$ represents an incoming odd-parity key index. The procedure CTYPE is used to determine the final comparator type.

```
COMPARATOR-TYPE ($d$, $r$)
Global $OIDX_{cid}$, $EIDX_{cid}$, $OFLAG_{cid}$, $EFLAG_{cid}$, $TYPE_{cid}$ : integer
*For all comparators in parallel.*
1  if $\lfloor 2^d CID / 2^r \rfloor$ = Odd number
2  then $OFLAG_{cid} \leftarrow 1$
3  else $OFLAG_{cid} \leftarrow 0$
4  if $d = 0$
5  then Set $TYPE_{cid} \leftarrow OFLAG_{cid}$
6  else if ($EIDX_{cid} > OIDX_{cid}$)
7      then $EFLAG_{cid} \leftarrow 1$
8      else $EFLAG_{cid} \leftarrow 0$
9      if $d = r - 1$
10     then Get the index difference : Diff $\leftarrow |EIDX_{cid} \leftarrow OIDX_{cid}| + 2^r$
11     else Get the index difference : Diff $\leftarrow |EIDX_{cid} \leftarrow OIDX_{cid}|$
12     Obtain the parity value : $p \leftarrow$ CPARITY(CID)
13     if $OFLAG_{cid} = EFLAG_{cid}$
14     then $TYPE_{cid} \leftarrow$ CTYPE(Diff, $d$, $p$)
15     else $TYPE_{cid} \leftarrow$ CTYPE(Diff, $d$, $\bar{p}$)
```

The procedure works as follows. Lines 1-3 initially set $OFLAG_{cid}$ based on the parameter r. The last comparator level of each stage is checked at line 4. If it is the last level then each comparator type is set depending on $OFLAG_{cid}$ at line 5 ($OFLAG_{cid} = 0 \Rightarrow$ an increasing comparator). Otherwise, the indicies of two incoming keys (an even-parity key and an odd-parity key) are used to determine each comparator type in lines 6-15. During these steps, the procedure CTYPE is called to determine the final comparator type, which returns 0 for an increasing comparator or 1 for a decreasing comparator in line 13-15.

Having defined these procedures, the algorithm CONSTRUCT-BSS which constructs the Batcher's sorter with simpler interconnections (BSS) for $N(=2^k)$ keys proceeds as follows:

Algorithm **CONSTRUCT-BSS**

- Input : Unsorted $N(=2^k)$ keys with $N/2$ comparators.

- Output : Batcher's sorter with simpler interconnection description.

```
1  for $r$ = 1 to log$N$ (= $k$) do
2      for $d$ = $r$ - 1 downto 0 do
3          Perform COMPARATOR-TYPE($d$, $r$) to set each comparator type.
4          if $r \neq 1$
5          then Perform CONNECT-LEVEL ($d$) to establish links between levels.
6      Perform MERGE-STAGE($r$) to establish merge links between stages.
```

CONSTRUCT-BSS shows how to construct the BSS. Lines 1-2 show that the BSS is composed of $k(k+1)/2$ comparator levels. Line 3 sets each comparator type in each level. Line 4 checks whether it is the first level of the first stage (see Figure 7). If it is the first level of the first stage then it only requires the merge link at line 6.

The following theorem shows that the considered BSS sorts N keys correctly.

Theorem 2 : A list of $N(=2^k)$ unsorted keys can be sorted using a BSS of $2^{k-2}(k^2 + k)$ comparators in $O(\log^2 N)$ time.

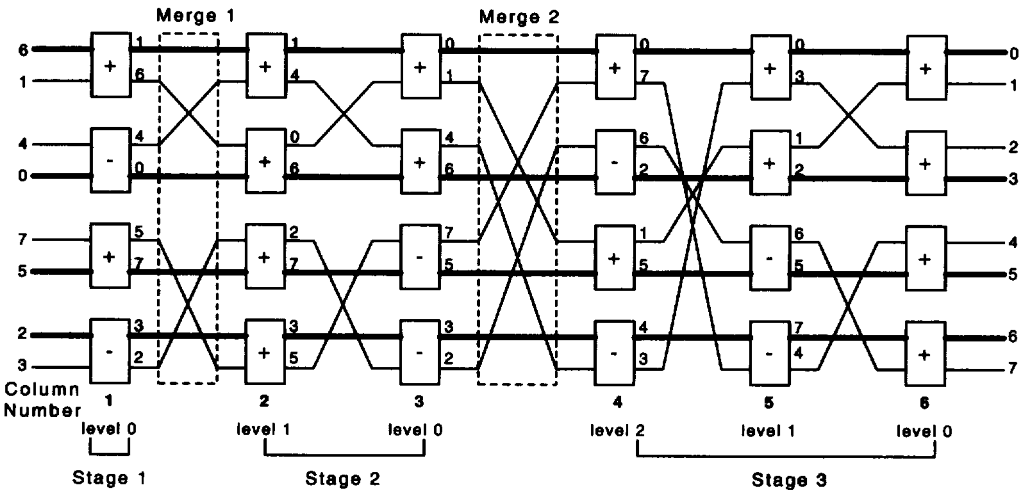
Proof : A BMS takes a bitonic sequence and transforms it into a sorted sequence. Sorting N keys in this sorter proceeds in log$N$ stages (i.e., iterations of the outer for-loop). At the first stage, the keys are divided into $N/2$ bitonic sequences of size 2 in which one key is an even-parity key and the other one is an odd-parity key. After a single compare-exchange step, the pairs are sorted iteratively ascendingly and descendingly (see stage 1 of Figure 7). Hence, all sequences of size 2^2 keys are bitonic as a result of the first stage. If a bitonic sequence of size 2^i is sorted into ascending order, while an adjacent sequence of size 2^i is sorted into decreasing order, then after i compare-exchange steps the resulting sequence of size 2^{i+1} is a bitonic sequence. Hence, the output of each level in the BSS is a concatenation of bitonic sequences that are twice as long as those at the input. By merging larger and larger bitonic sequences, a sorted sequence of size 2^k is obtained.

Given $N(=2^k)$ unsorted keys, the sorter is obtained by interconnecting log$N$ stage of levels, (stage$_1$, stage$_2$, ..., stage$_k$), through the interconnection scheme described above. The stage$_i$ consist of i comparator levels and each level contains 2^{k-1} comparators for a total of $2^{k-2}(k^2 + k)$ comparators. Each path goes through $\frac{k(k+1)}{2}$ levels. Note that $k(k+1)/2$ = $\log N(\log N+1)/2$. Therefore, the time complexity is $O(\log^2 N)$ since each level is performed in parallel with constant time. □

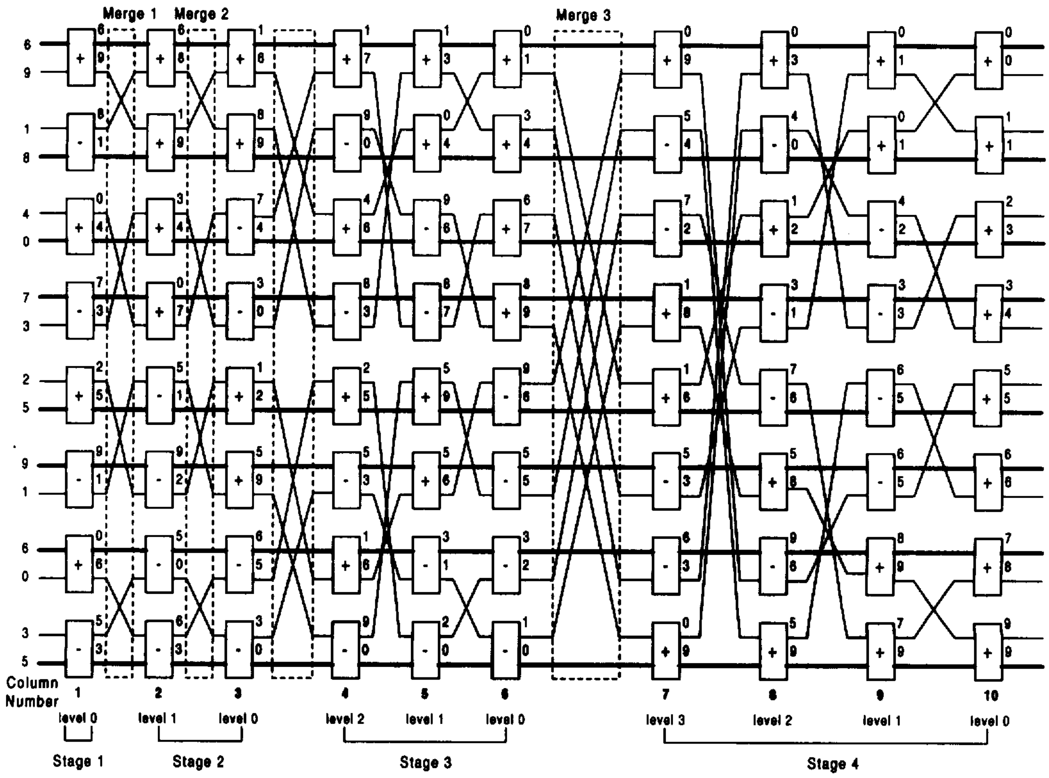
The BSSs for 8 and 16 keys are shown in Figures 7 and 8, respectively. Thick lines show that $N/2$ even-parity keys are wired straight through the sorter. Merge-link level (Merge i) shows the interconnection paths between two stages. Each comparator in each level performs a compare-exchange operation between an odd-parity key and an even-parity key.

4. Performance

Since Stone described an implementation of the bitonic sorting with a perfect-shuffle network (PSN), many researchers have improved the perfect-shuffle interconnection scheme and adapted it to a variety of applications[4,13,16]. Now, the proposed interconnection scheme is compared with Stone's. Two factors are usually used to measure the complexity of parallel sorters: (1) cost - the number of comparators the sorter contains and (2) time(or delay) - for any input



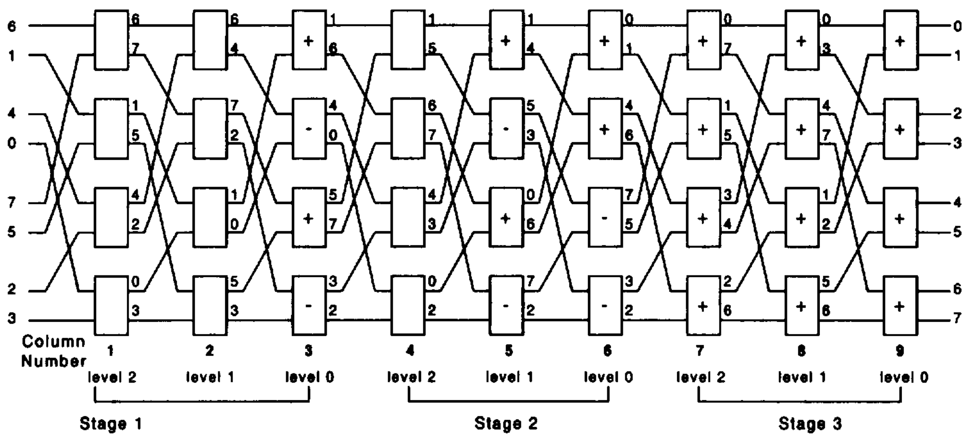
(Fig. 7) A BSS for 8 keys.



(Fig. 8) A BSS for 16 keys

x, the maximum number of comparators x has to visit before x exits the sorter.

Since every stage of the PSN consists of log$N$ levels and each level contains 2^{k-1} comparators, a list of $N(=2^k)$ unsorted keys can be sorted in $\log^2 N$ time with a sorter of $2^{k-1}*k^2$ comparators using the perfect-shuffle interconnections. Figure 9 shows the Batcher's sorter for 8 keys using the perfect-shuffle interconnection scheme(PSN). In this figure, there are three extra levels in stage$_1$ and stage$_2$, corresponding to the vertical tier of blank boxes. The blank boxes do not perform a compare-exchange operation: they output the values in the same order as they were input. Sample values are shown on the wires. As mentioned in theorem 2, the BSS requires $2^{k-2}(k^2+k)$ comparators in log$N$(log$N$+1)/2 time since the ith stage of the BSS needs i ($1 \le i \le$ log$N$) levels and each level contains 2^{k-1} comparators. Therefore, the interconnection scheme described here reduces the cost and time(or delay) by approximately one half compared with Stone's, and hence, improves performance. For easy understanding, Figure 10, Figure 11 and table 1 show the comparisons of the BSS and the PSN in terms of total cost (= total number of comparators) and total time(or delay), where the PSN denotes a sorter using Stone's perfect-shuffle interconnections.



(Fig. 9) A Batcher's sorter for 8 keys using perfect-shuffle interconnection.

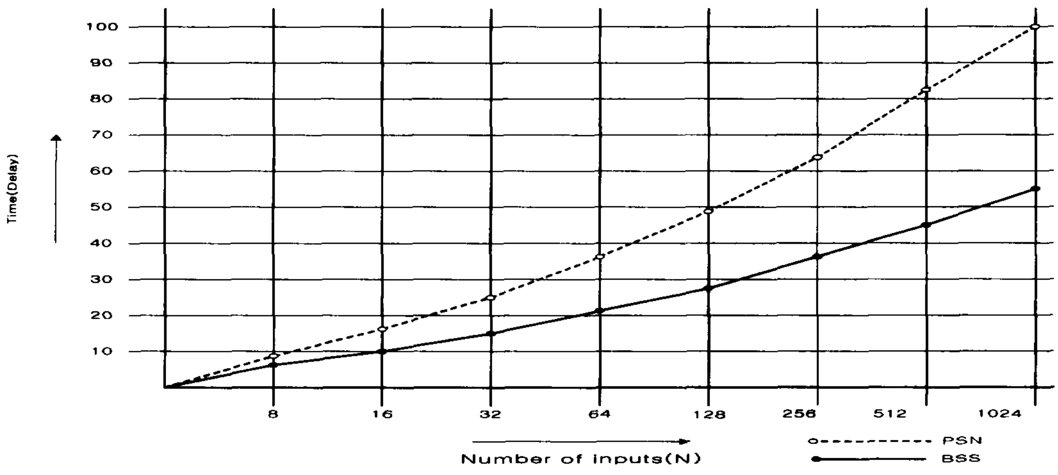
<Table 1> Total cost and total time versus number of inputs(N)

Type / #of input(N)	Cost		Time(or delay)	
	BSS	PSN	BSS	PSN
8	24	36	6	9
16	80	128	10	16
32	240	400	15	25
64	672	1,152	21	36
128	1,792	3,136	28	49
256	4,068	8,192	36	64
512	11,520	20,736	45	81
1,024	28,160	51,200	55	100

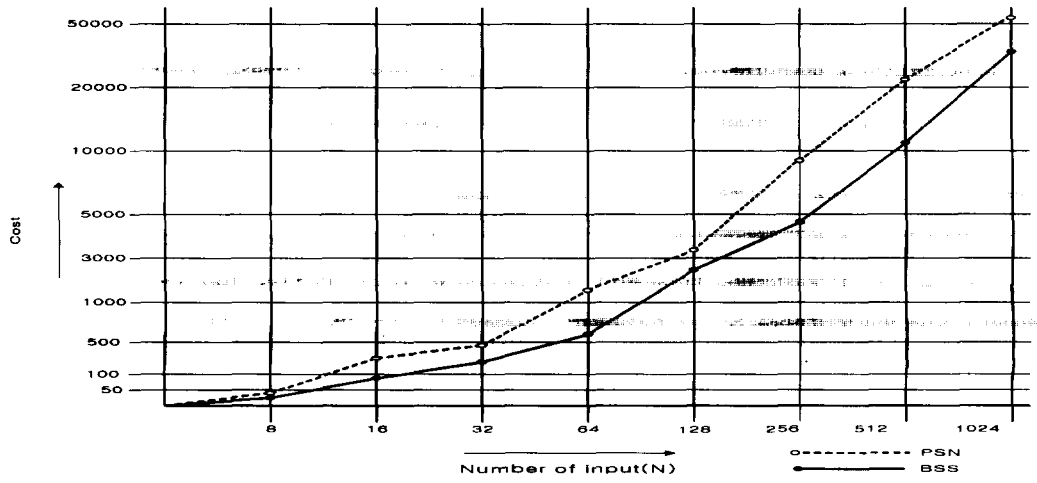
5. Applications

The applications of the proposed sorter(BSS) are presented in this section.

The BSS can be used to many ATM switching systems such as Moonshine, Sunshine, Starlite and SXmin in order to be applied to simplify the design of an arbitration circuit structure and a router. By the usage of the BSS in ATM switching system, the internal blocking which is an important factor of the design of the switching network is prevented. Furthermore, the BSS allows its use in solving some problems where large sets of cells(or data) must be manipulated. Some of these applications are switching networks with conflict resolution, multi-access memories and multi-processors [3]. There is no doubt that this sorter also can be used just for sorting and merging.



(Fig. 10) Total time of the BSS and the PSN



(Fig. 11) Total cost of the BSS and the PSN

6. Conclusions

This paper has focused on constructing the Batcher's sorter with simpler interconnections (BSS) and presented its applications for ATM switching systems. This proposed sorter has a time complexity of $O(\log^2 N)$ and a cost complexity of $O(N\log^2 N)$. In all comparators the compare-exchange operation is performed between an odd-parity key and an even-parity key. The interconnection scheme presented here is simplified between levels by employing a parity-strategy which leads to the algorithm CONSTRUCT-BSS. For simplifying the inter-level wiring, $N/2$ even-parity keys are wired straight through the sorter. This simplified interconnection scheme requires approximately half as many levels as the perfect-shuffle interconnection scheme: $\log N(\log N+1)/2$ levels instead of $\log^2 N$. Therefore, this proposed interconnection scheme outperforms the perfect-shuffle interconnection scheme both in terms of cost and delay. The layout of this proposed sorter on a printed circuit board or a VLSI chip may be simpler since half of the routes are straight lines. Furthermore, by the usage of the sorted cells, the cells in the ATM switches are routed without the internal blocking which is an important factor of the design of the switching networks.

References

[1] "Broadband aspects of ISDN", CCITT Recommendation, I.121, Nov. 1988.

[2] Ahmadi, H. and Denzel, W., "A survey of modern High-Performance Switching Techniques," IEEE J. on Selected Areas in Communications, Vol.7, No.7, pp.1091-1103, 1989.

[3] Batcher, K.E., "Sorting networks and their applications", Spring Joint Computer AFIPS Proc., Vol.32, pp.307-314, 1968.

[4] Batcher, K.E., "Low-cost Flexible Simulation with static shuffle network," Fourth Symposium on the Frontier of MPP (Frontiers 92), pp.434-441, IEEE, Oct. 1992.

[5] Chen, T. and Liu, S., "ATM Switching Systems," Artech House Inc., 1995.

[6] Giacopelli et al, J., "Sunshine: A High-performance self-routing Broadband packet switch Architecture", IEEE J. of selected Areas in Communications, Vol.9, No.8, pp. 1289-1298, Oct. 1991.

[7] Goke, L. and Lipsuski, G., "Banyan Networks for partitioning Multiprocessors," 1st Annual Symp. Computer Architecture, pp. 21-28, 1973.

[8] Hwang,W., Chen,W. and Deng,Y., "A High-Performance ATM switch with Completely and Fairly Shared Buffers", in Proceedings 1997 ICPADS, pp.203-208, December 1997.

[9] Imagawa, H., Urushidani, S. and Hagishima, K., "A new self-routing switch driven with input-output address difference", in Proc. GLOBECOM '88, pp.1607-1611, Dec. 1988.

[10] Ionescu, M. and Schauser, K., "Optimizing Parallel Bitonic Sort", in Proceedings of the 1997 IPPS, pp.303-309, April 1997.

[11] Kim, H.S. and Leon-Garcia, A., "A self-routing multistage switching network for broadband ISDN", IEEE Journal of selected Areas in Communications Vol.8, No.3, pp. 459-466, April 1990.

[12] Kulzer, J.J. and Montogomery, W.A., "Statistical switching architecture for future services", in Proc. ISS '84, pp.1-6, 1984.

[13] Kumar, M. and Jump, J., "Performance of Unbuffered Shuffle-exchange networks," IEEE Trans. Computers, Vol.C-35, No.6, pp.573-578, June 1986.

[14] Lee, Jae-dong and Batcher, K.E., "Minimizing Communication of a Recirculating

Bitonic Sorting Network", in Proceedings of the 1996 ICPP, Vol.I, pp.251-254, 1996.

[15] Pattavina, A., "Non-blocking Architectures for ATM Switching", IEEE Communications Magazine, pp.38-48, Feb. 1993.

[16] Stone, H.S., "Parallel processing with the perfect shuffle", IEEE Trans. on Computers, Vol.C-20, pp.153-161, Feb. 1971.

이 재 동

1985년 인하대학교 전자계산학과 (B.S)

1987년~1988년 대우중공업 정보관리센타

1991년 미국 Cleveland State University, Dept. of computer and Information science(M.S)

1996년 미국 Kent State University, Dept. of computer science(Ph. D)

1992년~1996년 Kent State University, 전산학과 T.A.

1997년~현재 단국대학교 전자계산학과 전임강사

관심분야 : 병렬처리, 알고리즘, Interconnection networks, 컴퓨터 네트워크, ATM network