분할 가능한 단단계(Single-Stage) Shuffle-Exchange 네트워크의 설계

(Design of a Partitionable Single-Stage Shuffle-Exchange Network)

이 재 동 †

(Jae-Dong Lee)

요 약 본 논문에서는 단단계(Single-Stage) Shuffle-Exchange 네트워크의 분할성에 대하여 연구하였다. SSEN_to_PSEN 알고리즘은 단단계 Shuffle-Exchange 네트워크를 분할 가능한 Shuffle-Exchange 네트워크로 변환하는 방법을 제안한다. 제안된 알고리즘은 네트워크의 크기가 $N \leq 8$일 경우에는 추가적인 링크없이 네트워크가 분할성을 갖는 것을 보이며, 네트워크의 크기가 $N \geq 16$일 경우에 단단계 Shuffle-Exchange 네트워크를 분할하기 위해서는 추가적인 링크가 필요하다. SSEN_to_PSEN 알고리즘의 시간 복잡도는 $\theta(N\log N)$이며, 하이퍼큐브 네트워크와 비교하여 분할 가능한 Shuffle-Exchange 네트워크는 적은 링크 수를 사용한다. 분할이 가능해짐에 따라서 대용량의 병렬컴퓨터에서 분할 가능한 Shuffle-Exchange 네트워크는 여러 사용자들을 위한 다양한 문제의 처리가 동시에 가능하기 때문에 컴퓨터의 처리 효율이 향상됨을 알 수 있다.

키워드 : 분할성, 상호연결네트워크, 병렬컴퓨터, shuffle-exchange 네트워크

Abstract This paper presents the problem of partitioning the Single-Stage Shuffle-Exchange Network(SSEN). An algorithm, named SSEN_to_PSEN, is devised to transform an SSEN into a Partitionable Shuffle-Exchange Network (PSEN). The proposed algorithm presents that the SSEN can be partitioned into independent sub-networks without additional links for $N \leq 8$. Additional links are needed in order to partition an SSEN, but only when $N \geq 16$. The running time of the algorithm SSEN_to_PSEN is $\theta(N\log N)$. By comparing with a hypercube network, the PSEN is less expensive than a hypercube network even when some additional links are added. By partitioning, a large PSEN in a massively parallel machine can compute various problems for multiple users simultaneously, thereby the processing efficiency of the machine is improved.

Key words : Shuffle-exchange network, Partitionability, Single-Stage network, Interconnection network, Parallel computer

1. Introduction

In recent years, the design of multiprocessor(or multi-computer) systems in which a large number of processors(or PCs) can be applied to a single problem has grown. Many applications, such as weather prediction, pollution monitoring, radar tracking, and image processing, are modeled by imposing a grid over the domain being modeled[1,2,3]. Any multi-ocessor(or multi-computer) system must be designed to allow efficient communication between processors. As the number of processors grows, the interconnection design becomes more critical as centralized bus connections become impractical. One of the major problems in designing multiprocessor systems is to design a cost-effective interconnection network.

The partitionability of an interconnection network is the ability to divide the network into independent sub-networks of different sizes[3,4,5]. A partitionable network provides the following advantages: (a) it

· The research was conducted by the research fund of Dankook University in 2000.

† 정 회 원 : 단국대학교 컴퓨터과학전공 교수
letsdoit@dankook.ac.kr

논문접수 : 2002년 7월 19일
심사완료 : 2002년 9월 11일

can be used in parallel for multiple users in multi-ocessor systems such as SIMD, multiple-IMD, MIMD and partitionable SIMD/MIMD environments and (b) a smaller-size SIMD machine will utilize its processors(or PEs) more efficiently than a larger-size SIMD machine when both are performing the same task for some cases, so information about partitioning can be used to improve processing efficiency in a multiprocessor system[6,7,8].

The shuffle-exchange network has been shown to be a very good interconnection network in some applications such as polynomial evaluation, sorting, matrix transposition, and fast Fourier transform [1,3,7,9,10]. Figure 1.(a) shows a Perfect Shuffle Network for 8 processors. Here the two-by-two switches are removed and replaced by exchange links between pairs of processors to make a Single-stage Shuffle-Exchange Network which is shown in Figure 1.(b). An SSEN has the following advantages:

• An SSEN uses less hardware than the popular multi-stage network like the Butterfly network or the static hypercube network [3,10].

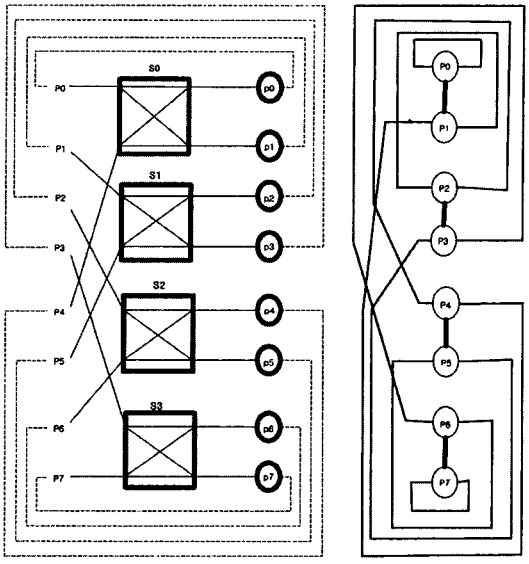
• An SSEN can be used as a low-cost flexible simulator for other networks like the hypercube network [10,11].

• An SSEN is very efficient for the implementation of particular algorithms such as the fast Fourier transform, polynomial evaluation, sorting and matrix transposition [13,14].

• An SSEN has the capability of realizing every other permutation [5,10].

Despite these advantages, an SSEN has not been popularly implemented in some machine. One of the reasons is that an SSEN cannot be partitioned into independent sub-networks, as shown by Siegel in 1980[6,8].

Here, the partitioning method for the Single-Stage Shuffle Exchange Network is presented. In order to transform an SSEN into a PESN, a few additional links between processors are needed. An algorithm SSEN_to_PSEN which shows the method for adding the additional links is discussed.



(a) (b)

Fig 1 (a) Perfect-Shuffle Network of size 8 with four 2×2 switches. (b) Single-stage Shuffle-Exchange Network of size 8 with four exchange links.

This paper is organized in the following way: Section 2 presents definitions for an SSEN, section 3 presents a Partitionable Shuffle-Exchange Network(or PSEN for short) and discusses an algorithm SSEN_to_PSEN which transforms an SSEN into an efficient PSEN. Section 4 discusses the comparison to other networks and section 5 states some conclusions.

2. Definitions

According to [1,5,9,11,12], the following definitions are used for a Single-stage Shuffle-Exchange Network.

① For some integer n, an SSEN has $N(=2^n)$ processors.

② An SSEN uses three ports in each processor: A shuffle-in port, a shuffle-out port, and an exchange port.

③ The processors are indexed 0 through $N-1$ for physical processor address.

④ For $0 \le i < \frac{N}{2}$, the shuffle-out port of processor P_i is connected to the shuffle-in port of P_{2i}

⑤ For $\frac{N}{2} \le i < N$, the shuffle-out port of processor P_i is connected to shuffle-in port of P_{2i+1-N}

⑥ For $0 \le i < \frac{N}{2}$, the exchange ports of P_{2i} and P_{2i+1} are connected to each other.

⑦ All links are bi-directional.

It is useful for our purpose to describe an SSEN using graphs. Therefore, the following definition is introduced.

(Definition 1) Let G=(V,E) be an ordered graph. Then G is an SSEN graph denoted by SSEN(G), if and only if all of the following conditions are satisfied:

① $|V| = 2^n = N$ be the number of vertices.

② For $0 \le i \le \frac{N}{2}-1$, $(v_i, v_{2i}) \in E$ and

③ For $\frac{N}{2} \le i \le N-1$, $(v_i, v_{2i+1-N}) \in E$ and

④ For $0 \le i \le \frac{N}{2}-1$, $(v_{2i}, v_{2i+1}) \in E$

By associating vertices with processors and edges with links, it is clear from the definitions that a graph G is an SSEN-graph if and only if the associated network is an SSEN.

3. A Partitionable Shuffle-Exchange Network

This section introduces the Partitionable Shuffle-Exchange Network and describes an algorithm SSEN_to_PSEN which lead to a PSEN by adding additional links into an SSEN.

(Definition 2) Let G=(V,E) be an ordered graph, and let $|V| = 2^n = N$ be the number of vertices. Let us denote with $V^1 = \{v_0, v_1, \cdots, v_{\frac{N}{2}-1}\}$ and with $V^2 = \{v_{\frac{N}{2}}, v_{\frac{N}{2}+1}, \cdots, v_{n-1}\}$. Then G is a PSEN-graph denoted by PSEN(G) if and only if

① SSEN(G) and $|V| = 2$ or

② SSEN(G) and PSEN(G^1) and PSEN(G^2), where $G^1 = (V^1, E^1)$ and $G^2 = (V^2, E^2)$ and $E^1, E^2 \subseteq E$.

Clearly, if a graph G is a PSEN-graph, then the associated network is a Partitionable Shuffle-Exchange Network.

(Theorem 1) Let G=(V,E) be an SSEN-graph (i.e., SSEN(G) holds). If $|V| \le 8$, then SSEN(G) => PSEN(G) (i.e., G can be partitioned into independent sub-networks without additional links).

(Proof) Let SSEN(N) be an SSEN-graph of N PEs. From our assumption, it follows that we have to show the theorem for SSEN(2), SSEN(4) and SSEN(8). If $N = 2$, then trivially SSEN(2) => PSEN(2) by definition 2. When $N = 4$, figure 2.(a)-(b) shows that two copies of SSEN(2) can be mapped onto SSEN(4). This graph shows that SSEN(4) => PSEN(4) without additional links. When $N = 8$, figure 2.(b)-(c) shows that two copies of SSEN(4) can be mapped onto SSEN(8). This graph shows that SSEN(8) => PSEN(8) without additional links. Therefore, if $|V| \le 8$, then SSEN(G) => PSEN(G). □

3.1 An algorithm SSEN_to_PSEN

Since an SSEN cannot be partitioned into independent sub-networks, the SSEN needs additional link(s) to be a PSEN for $N \ge 16$[6,8]. In the following, the algorithm SSEN_to_PSEN which shows how to add the needed additional links onto an SSEN in order to construct the PSEN is presented.

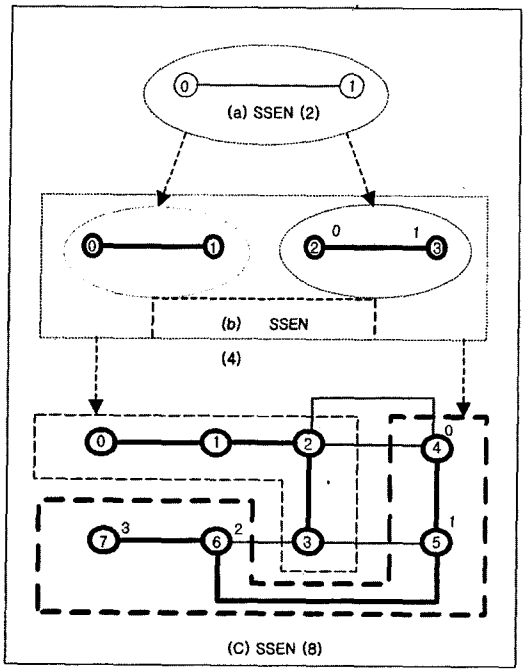


Fig 2 Partitioning graphs of the SSENs without additional links when $|V| \le 8$

Algorithm SSEN_to_PSEN : This algorithm transforms an SSEN into a PSEN by establishing additional links.

- Input: G with SSEN(G)
- Output: G′ with PSEN(G′) such that G ⊆ G′

Comment: For an additional connection in step S2, processor(P_i) for the shuffle-in port (b_{k-1}, b_{k-2}, ⋯, b_1, b_0) is connected to the processor(P_j) for the shuffle-out port (b_{k-1}, b_0, b_{k-2}, ⋯, b_1), where i and j represent the binary vector of each processor. This connection is a right end-around shift of the rightmost (k-1) bits of shuffle-in vector.

S0 [Check termination]
$2^k \leftarrow |V|$, where $|V|$ = size (G);
If $k \leq 3$ (i.e., $|V| \leq 8$), then return(G);
S1 [Renumbering the logical index i of each processor]
$i \leftarrow i \mod 2^k$;
S2 [Establish additional links]
S2.1 For ($0 \leq i \leq 2^{k-1}-1$ and i = odd)
$i \leftarrow$ binary vector($b_{k-1}, b_{k-2}, \cdots\cdots, b_1, b_0$);
$j \leftarrow$ binary vector($b_{k-1}, b_0, b_{k-2}, \cdots\cdots, b_1$);
if $((v_i, v_j) \notin E)$
then $E \leftarrow E \cup \{(v_i, v_j)\}$
S2.2 For ($2^{k-1}+1 \leq i < 2^k-1$ and i = even)
$i \leftarrow$ binary vector($b_{k-1}, b_{k-2}, \cdots\cdots, b_1, b_0$);
$j \leftarrow$ binary vector($b_{k-1}, b_0, b_{k-2}, \cdots\cdots, b_1$);
if $((v_i, v_j) \notin E)$
then $E \leftarrow E \cup \{(v_i, v_j)\}$
S3 [Main Recursion]
Divide G into G^1 and G^2 such that size(G^1)=size(G^2)= 2^{k-1};
$G^{(1)}$ = SSEN_to_PSEN(G^1);
$G^{(2)}$ = SSEN_to_PSEN(G^2);
Return ($G^{(1)} \cup G^{(2)}$);

(Lemma 1) Let G={V,E} be a set of ordered SSEN(G), where $|V| = 2^n = N$. Then, after the step S2 of the algorithm SSEN_to_PSEN, SSEN(G) => SSEN(G^1) and SSEN(G^2), where G^1=(V^1, E^1) and G^2=(V^2, E^2).

(Proof) The condition of ① of definition 1 is clearly satisfied. (i) Since G is a set of ordered SSEN(G), when we bisect G into G^1 and G^2, G^1 still suffices the conditions ② and ④ of definition 1 and step S2.1 of the algorithm SSEN_to_PSEN establishes additional edges for condition ③ of definition 1. And then, after the step S2.1, SSEN(G) => SSEN(G^1) with size (see Figure 3:L group). (ii) Similarly, G^2 suffices the conditions ③ and ④ of definition 1 and step S2.2 establishes additional edges for condition ② of definition 1 (see Figure 3:H group). Then, after step S2.2, SSEN(G) => SSEN(G^2) with size . Therefore, by the results of (i) and (ii), SSEN(G) => SSEN(G^1) and SSEN(G^2) (i.e., G can be partitioned into independent sub-networks). □

(Theorem 2) Let G be an SSEN(G). Then the output of the algorithm SSEN_to_PSEN is a PSEN-graph (i.e., PSEN(G′) is true).

(Proof) We prove it by induction on |V|. Let G=(V,E) be an ordered SSEN(G) and $|V|=2^n$ be the number of vertices. When $n \leq 3$, the claim is proved by theorem 1. Now consider for $n \geq 4$. Let G^1 be the result of step S2.1 of an algorithm to G and let G^2 be the result of step S2.2 to G. Then by lemma 1, there exist SSEN(G^1) and SSEN(G^2), where G^1=(V^1, E^1) with $|V^1| = 2^n-1$ and G^2=(V^2, E^2) with $|V^2|=2^n-1$. By induction hypothesis, SSEN_to_PSEN(G^1) and SSEN_to_PSEN(G^2) return PSEN-graph, respectively. Therefore, by definition 2, we conclude that the algorithm SSEN_to_PSEN returns PSEN-graph (i.e., PSEN(G′) is true). □

(Theorem 3) The running time of the algorithm SSEN_to_PSEN is $\theta(N \log N)$

(Proof) Since the partitioning produces two groups of size $\frac{N}{2}$ and step S1 requires $\theta(N)$, the recurrence is $T(N)=2T(\frac{N}{2})+\theta(N)$, which by the Master Theorem[14] has solution $T(N) = \theta(N \log N)$. Therefore, the running time of the algorithm SSEN_to_PSEN is $\theta(N \log N)$. □

3.2 An example for 16-processor PSEN

As an example, consider a 16-processor SSEN. The algorithm SSEN_to_PSEN is used in the following way in order to establish additional links.

- Step S1 renumbers the logical index of each processor. Here the logical index of each processor is identical with physical index (straightforward mapping).
- In step S2.1, for $0 \leq i < 7$ and i = odd (for a low group), vector(0001) of shuffle-in port produces vector(0100) of shuffle-out port, vector(0011) produces

vector(0101) and vector(0101) produces vector(0110), for additional links, respectively. Thus the additional links for the low group are connected between the following pairs:

P_1 - P_4 P_3 - P_5 P_5 - P_6

Using these additional links, the low group can be a complete 8-processor SSEN (see Figure 3:L).

• In step S2.2, for $9 \leq i < 15$ and i = even (for a high group), vector(1010) of shuffle-in port produces vector(1001) for shuffle-out port for an additional connection. Vector(1100) produces vector (1010) and vector(1110) produces vector(1011), respectively. Thus the additional links for the high group are connected between the following pairs:

P_{10} - P_9 P_{12} - P_{10} P_{14} - P_{11}

The high group can be another 8-processor SSEN using these additional links (see Figure 3: H).

• Step S3 returns a PSEN-graph with six additional links in addition to an SSEN-graph.

Therefore, this 16-processor PSEN can be partitioned into independent sub-SSENs of various size, powers of two(2,4 and 8). Figure 3 shows the 16-processor PSEN for a partitioning of various size. In Figure 3, 16-processors are divided into two groups, based on high and low physical numbers. Each group also can be divided into independent subgroups of various sizes by recursion (see smaller boxes). Each number indicates the physical number: L- and H- indicate low and high group logical numbers respectively. Thick lines show the additional links and dashed lines show the duplicated link or connection itself. Here, low indexed group (see box L) shows a complete 8-processor SSEN with three additional links after a partition. For an easy understanding, Table 1 shows the connections of a 16-processor PSEN with six additional links which are represented by letter e.

Figure 4 illustrates a PSEN-graph with |V| = 16, where thick lines indicate the additional links. Here, L: and H: indicate the low and high logical group numbers, respectively. This figure shows that two copies of SSEN(8) can be mapped onto PSEN(16) with six additional links.

Table 1 Connection table of a 16-processor PSEN

PE	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}
P_0																
P_1	x		x													
P_2					x											
P_3			x				x									
P_4		e						x								
P_5				e	x						x					
P_6						e							x			
P_7							x								x	
P_8		x														
P_9				x					x		e					
P_{10}													e			
P_{11}								x			x				e	
P_{12}										x						
P_{13}													x	x		
P_{14}															x	
P_{15}																x

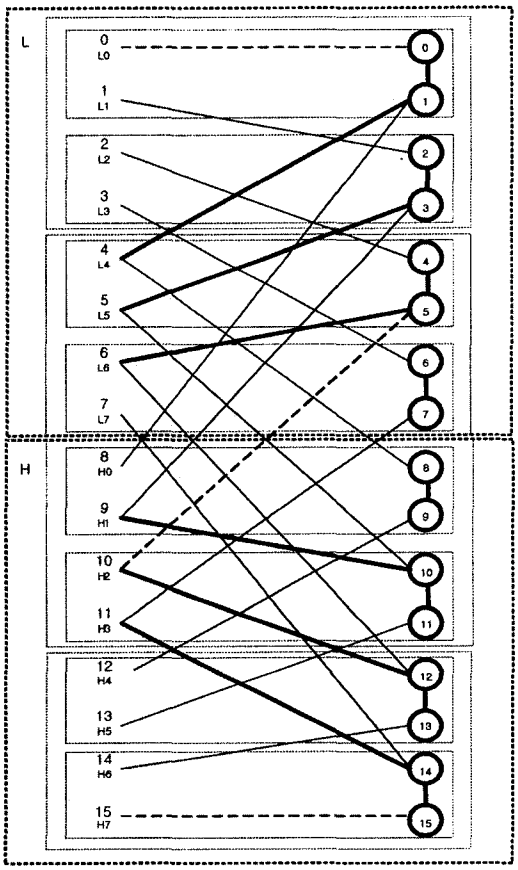
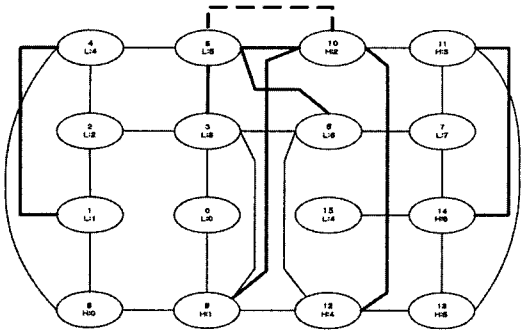


Fig 3 Partitioning the 16-processor PSEN with six additional links. Thick lines show the additional links



PSEN(16)

Fig 4 16-processor PSEN-graph

4. Comparison to other networks

This section discusses the criteria that characterize the cost and performance of static networks.

4.1 Comparison of the cost

Many criteria can be used to evaluate the cost of a network. One way of defining the cost of a network is in terms of the number of communication links required by the network[1,4,9]. Here, the PSEN is compared with hypercube network since it is the most popular network for multicomputers.

Table 2 shows that the average number of links per processor (Links/Processor) and the total cost of the PSEN (Total Cost) which result from our simulation for $n \leq 11$. In Table 2, saved the number of links indicates the difference of the total number of links between the PSEN and the Hypercube network. Here, the total number of links of the PSEN is always smaller than that of the hypercube network. Theorem 4 demonstrates that the cost of the PSEN is always smaller than that of a hypercube network. Thus according to this cost criteria, the PSEN is less expensive than a hypercube network even when some additional links are added to allow for partitioning. Even though the PSEN is less expensive than the hypercube, it is not always superior to the hypercube network, because the hypercube network is better than other criteria such as regularity(or modularity) and the fault-tolerance. We must tradeoffs and select a network on the basis of both the cost and its intended applications.

(Theorem 4) The total number of links of the PSEN is always smaller than that of the hypercube network.

(Proof) $H(n) = n2^{n-1}$ is the cost of the hypercube [5]. Let $P(n)$ denote an upper bound for the cost of PSEN, where $P(n)$ = exchange_links + shuffle_links + additional_links (or extra_links). Now we try to show that $P(n) < H(n)$. We clearly know that exchange_links is 2^{n-1} and shuffle_links is $2^n - 2$ by our definition (see Section 2). Consider the sum of additional_links by applying the algorithm SSEN_to_PSEN. Each step in S2 needs at most $(2^{i-1}-2)$ additional links. By applying our recurrence algorithm, the total number of additional_links is

$$\begin{aligned}\sum extra\_links &= \sum_{i=n}^{4} 2^{n-i}(2^{i-1}-2)\\ &= (n-3)2^{n-1} - \sum_{i=1}^{n-3} 2^i\\ &= (n-3)2^{n-1} - (2^{n-2}-2)\\ &= n2^{n-1} - 3\cdot 2^{n-1} - 2^{n-2} + 2.\end{aligned}$$

$$\begin{aligned}\text{Thus } P(n) &= 2^{n-1} + 2^n - 2 + n2^{n-1} - 3\cdot 2^{n-1} - 2^{n-2} + 2\\ &= n2^{n-1} - 2^{n-2}.\end{aligned}$$

Clearly $P(n) < H(n)$.

□

Table 2 Comparisons between PSEN and Hypercube

N(n)	PSEN		Hypercube		Saved number of links
	Avg. # of Links per Processor	Total Cost	Avg. # of Links per Processor	Total Cost	
8(3)	2.50	10	3	12	2
16(4)	3.38	27	4	32	5
32(5)	4.38	70	5	80	10
64(6)	5.34	171	6	192	21
128(7)	6.34	406	7	448	42
256(8)	7.34	939	8	1024	85
512(9)	8.34	2134	9	2304	170
1024(10)	9.33	4779	10	5120	341
2048(11)	10.33	10582	11	11264	682

4.2 Comparison of the other characteristics

This section discusses criteria which are the number of nodes, bisection width of the network and the number of edges per node.

The bisection width of a network is the defined

as the minimum number of communication links that have to be removed to partition the network into two equal halves. It is a good measure of the overall bandwidth of the network. High bisection width is better. The bisection width of a network should scale close to linearly with the number of processors for a scalable network. If the bisection width does not scale well, the interconnection network will become a bottleneck as the number of processors is increased. Here, Butterfly network clearly provides the best bisection width.

It is the best if the number of edges per node is a constant independent of the network size, because then the network scales more easily to systems with large numbers of nodes. The hypercube is noteworthy as the only network in which the number of edges per node is an increasing function of the network size.

Table 3 Comparison of size, bisection width, and constant number of edges

Network	SIZE(N)	Bisection Width	Constant # of links
Bufferfly	$(n+1)\cdot 2^n$	2^n	YES
HyperCube	2^n	2^{n-1}	NO
3-D mesh	n^3	n^2	YES
PSEN	2^n	2^{n-1}	YES

Various characteristics of the compared networks are shown in Table 3. Figures 5 and 6 show a plot of the bisection width of various networks with respect to the number of processors and the network size, respectively.

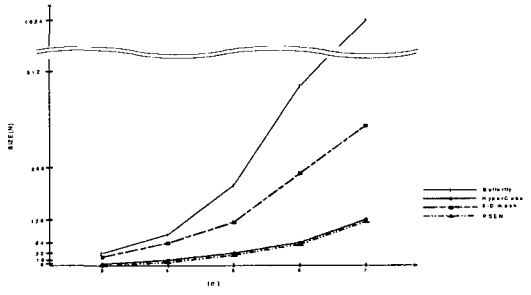


Fig 5 Comparison of network size with respect to n

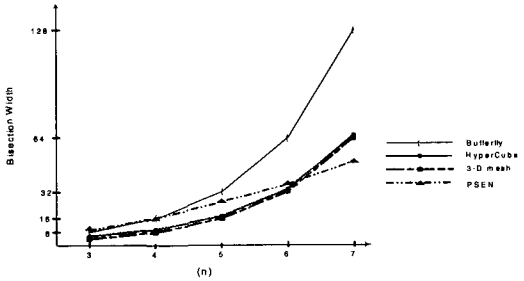


Fig 6 Comparison of the bisection width with respect to n for various network

5. Conclusions

This paper has focused on constructing the Partitionable Shuffle-Exchange Network, which includes the additional links for the partitioning of a Single-Stage Shuffle-Exchange Network. The algorithm SSEN_to_PSEN transforms an SSEN into a PSEN by adding a few additional links, but only when $N \geq 16$. This PSEN uses less hardware than a hypercube network even when a few additional links are added. Since the partitionable multiprocessor machine is a parallel processing system which can be dynamically reconfigured to operate as one or more independent virtual multiprocessor machines of various sizes, the PSEN is obviously applicable to a partitionable multiprocessor system. Therefore, a system using the PSEN can treat the various problems of multiple users simultaneously such that more efficient processing is possible in a multiprocessor system.

References

[1] Sara Baase and Allen Van Gelder, *'Computer Algorithms:3rd Edition'*, Addison Wesley, 2000.

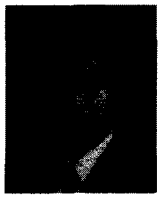
[2] D.E. Culler, J.P. Singh and A. Gupta, *'Parallel Computer Architecture: A Hardware/software Approach'*, M.K. Publishers, CA, 1998

[3] V. Kumar, A. Grama, A Gupta, and G. Karypis, *'Introduction to Parallel Computing : design and analysis of parallel algorithms'*, The Benjamin/ Cummings Publishing Company Inc., 1994.

[4] P. Chen, D.H. Lawrie, and D.A. Padua, "Interconnection Networks using shuffles," *IEEE Computer*, Vol. 14, pp.55-64, Dec. 1981.

[5] H.J. Siegel, "Partitionable SIMD computer system

interconnection network universality," *in Proc. 16th annual Allerton Conference on Communication, Control and Computers,* pp.586-595, Oct. 1978.
[6] T. Lang and H. Stone, "A Shuffle-Exchange Network with Simplified Control," *IEEE Transactions on Computers,* Vol. C-25, NO.1, Jan. 1976.
[7] J-D. Lee and K. Batcher, "Minimizing Communication in the Bitonic Sort," *IEEE Transactions on Parallel and Distributed Systems,* Vol. 11, No. 5, May 2000
[8] H.J. Siegel, "The Theory Underlying the Partitioning of Permutation Networks," *IEEE Transactions on Computers,* Vol. C-29, No.9, September 1980.
[9] H-I. Choi, "Fault tolerant Bitonic Sorting Networks and Static Shuffle-Exchange Networks," Ph.D Dissertation, Kent State University, 1997.
[10] H.J. Siegel, *'Interconnection networks for Large-Scale Parallel Processing : Theory and Case Studies'*, Second Edition, McGraw-Hill, 1990.
[11] K.E. Batcher, "Low-cost Flexible Simulation with the Static Perfect Shuffle Network," Fourth Symposium on the Frontiers of Massively Parallel Computation (Frontiers '92), pp.434-441, *IEEE Computer Society Press*, Oct. 1992.
[12] K.E. Batcher, "Decomposition of Perfect Shuffle Networks," *Proceedings of the 1991 International Conference on Parallel Processing,* CRC Press, Boca Raton FL, Vol.I, pp,255-262, 1991.
[13] H.S. Stone, *'High-Performance Computer Architecture'*, Second Edition, Addison-Wesley, Reading MA, 1990.
[14] H.S. Stone, "Parallel processing with the perfect shuffle," *IEEE Transactions on Computers,* Vol C-20, pp. 153-161, Feb.1971.



이 재 동

1991년 9월~1996년 5월 Kent State University 전자계산학(Computer Science) 박사(Ph.D). 1989년 1월~1991년 3월 Cleveland State University 전산정보학(Computer&Information Science) 석사(M.S). 1981년 3월~1985년 2월 인하대학교전자계산학(Computer Science) 학사(B.S). 1997년 3월~현재 단국대학교 정보컴퓨터학부 교수. 1996년~1997년 (주) 두루넷 기술기획팀. 1992년~1996년 Kent State University R.A & T.A. 1987년~1988년 대우중공업 정보관리실 시스템 분석. 관심분야는 (mobile) Internet Technologies/Applications, High Performance Networks/Network design, GIS Technologies and Applications, Many aspects of parallel/distributed processing