결함 포용 정적 Shuffle-Exchange 네트워크

(Fault Tolerant Static Shuffle-Exchange Network)

최 홍 인 †

(Choi Hong In)

요 약 정적 shuffle-exchange 네트워크는 여러 응용 알고리듬에 적용되고 현재 많이 사용되는 다중 단계 네트워크에 비해 적은 하드웨어를 사용하는 등 많은 장점이 있으나 아직까지 어떤 병렬처리 컴퓨터에도 채택된 없었다. 그 이유 중에 하나는 결함 내성 기능이 없었기 때문이다.본 논문에서는 다중 결함 포용 정적 shuffle-exchange network를 소개한다. 본 논문에서 제시되는 결함 포용 정적 shuffle-exchange 네트워크는 k 결함을 제어하기 위해서 최소 $2k$의 추가 처리 요소들과 각 처리 요소들은 최대 $4k$의 추가 shuffle 링크를 필요로 한다. k 결함 내성을 가진 정적 shuffle-exchange 네트워크를 m개의 동일한 모듈로 분리하여 네트워크의 신뢰성을 증가시키는 것을 보였다.

키워드 : 결함 내성, 병렬처리, 네트워크

Abstract A static shuffle-exchange network is not only useful for several parallel applications but also use less hardware than the popular multi-stage network or hypercube. Even though it has a lot of advantages, it has never been used in any implemented parallel machine. One of the reasons is there has not been any techniques to make the network fault-tolerant.

In this paper multiple fault-tolerant static shuffle-exchange networks are presented. In order to recover from k faulty processing elements, a network needs at least $2k$ additional processing elements and at most $4k$ additional shuffle ports for each processing elements. By decomposing the k fault-tolerant static shuffle-exchange network into m identical modules, this paper shows that the reliability of the network can be increased.

Key words : Fault-Tolerant, Parallel Processing, Network

1. Introduction

Perfect shuffle network, described by Stone[1,2] in 1971, can be used as a base architecture in a massively parallel machine. Two important characteristics of perfect shuffle networks are

1. They are not only useful for particular algorithms but have a wide variety of application: one can simulate the most widely used networks like multi-stage networks and hypercube networks.
2. They use less hardware than the popular multi-stage networks (rather than pass 2^n items through n stages of switches one time, a perfect shuffle network passes the 2^n items through one stage of switched n times), and hypercube networks. (rather than n links per processing elements in an n-cube, a perfect shuffle network has a constant number of links per processing element)

Despite these advantages a perfect shuffle network has never been used in any implemented machine whereas multi-stage networks and hypercubes are very popular. One reason a perfect shuffle network has never been implemented in any machine is that there has not been any technique to make the network fault tolerant. The inter-node message

† 정 회 원 : 단국대학교 전자계산학과 교수
kchoi@cs.dankook.ac.kr
논문접수 : 2002년 10월 22일
심사완료 : 2003년 1월 16일

passing technique can not be used since the perfect shuffle network can be disconnected even in the presence of a single fault. Graceful degradation of performance such as in a hypercube network is not possible either. To tolerate faults redundancy must be added to the perfect shuffle network.

Batcher[3] introduced a static shuffle-exchange network which is alternate description of perfect shuffle network :

- A static shuffle-exchange(SSE) network uses $N=2^n$ processing elements(PEs).
- Each PE has a shuffle-in, a shuffle-out, and an exchange port.
- The PEs are numbered from 0 through $N-1$.
- For $0 \le i < N/2$, the shuffle-out port of PE P_i is connected to the shuffle-in port of P_{2i}.
- the shuffle-out port of $P_{i+N/2}$ is connected to the shuffle-in port of P_{2i+1}; and
- the exchange port of P_{2i} and P_{2i+1} are connected to each other.
- All links are bi-directional.

It is useful to describe a *SSE* network with graphs. The following graph G represents a distributed multiprocessor with a static interconnection network.

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

1. $|V|=2^n=N$ *is the number of vertices.* $V=(v_0, v_1, \ldots, v_{N-2}, v_{N-1})$
2. *For* $0 \le i \le \frac{N}{2}-1$, $(v_{2i}, v_{2i+1}) \in E$ *and* $(v_i, v_{2i}) \in E$ *and*
3. *For* $\frac{N}{2} \le i \le N-1$, $(v_i, v_{2i+1-N} \in E)$

By associating vertices with PEs and edges with links it is clear from Definition 1 that a graph G is *SSE*-graph if and only if the associated network is an *SSE* network. This paper develops a fault-tolerant static shuffle-exchange network by adding redundant PEs and links. The fault model is considered here permanent fault in links or PEs.

A single fault-tolerant *SSE* network is introduced in Section 2 and a multiple fault-tolerant *SSE* network is developed in Section 3. Decomposition of the multiple fault-tolerant *SSE* network into modules is discussed in Section 4, and conclusion of this paper is given in Section 5.

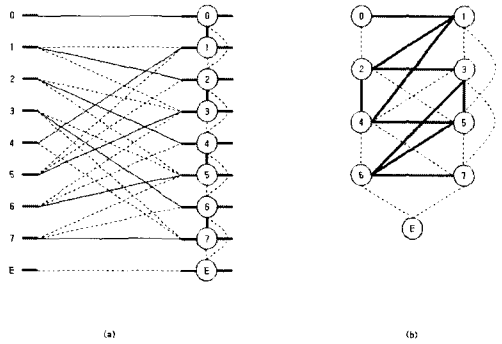


Figure 1 Single fault-tolerant SSE network using an extra PE.

2. Single Fault-Tolerant Static Shuffle-Exchange Network

To tolerate any single fault in an *SSE* network with $N=2^n$ PEs, it is necessary and sufficient that a spare PE be connected to each of the PEs in the *SSE* network. However, in that case, the number of extra links for the spare PE will be $N-1$, which is much too many if N is large. In VLSI design it is best if all PE's and modules are identical. Therefore, it is important to find the optimal number of the maximum node degree in a single fault-tolerant *SSE* network.

There are several ways to build a fault-tolerant *SSE* network. Here we introduce the simplest fault-tolerant(*SFT*1) technique in a *SSE* network. The basic idea of *SFT*1 is the use of redundancy; one spare PE and a certain number of extra links per PE. If a fault occurs in P_i or a link attached to P_i, use P_{i+1} as P_i, P_{i+2} as P_{i+1}, and so on. *SFT*1 can achieve the fastest reconfiguration time. Equations 1 and 2 define the exchange and shuffle-out connection links of P_i $(0 \le i \le N)$ in *SFT*1, respectively. Here, P_N represents a spare PE.

$$EX_i = \begin{cases} i+1, i+2 & \text{if } i=0 \text{ or } even(i) \\ i-1, i+1 & \text{if } odd(i) \end{cases} \quad (1)$$

$$SH1_i=\begin{cases} i & \text{if } i=0 \text{ or } i=N \\ 2i-1, 2i+1-N & \text{if } i=N/2 \\ 2i-1, 2i, 2i+1 & \text{if } 0<i<N/2 \\ 2i-1-N, 2i-N, 2i+1-N & \text{if } N/2<i<N \end{cases} \quad (2)$$

As shown in Equations 1 and 2 the maximum number of the additional ports for each PE in *SFT1* is six: all even numbered PEs except P_0 and P_N need three additional exchange ports, two additional shuffle-out ports (one additional shuffle-out for $P_{N/2}$), and one additional shuffle-in port whereas all odd numbered PEs needs one additional exchange port, two additional shuffle-out ports, and three additional shuffle-in ports (two additional shuffle-in ports for P_1 and P_{N-1}). No additional shuffle ports are needed for P_0 and P_N. Equation 2 shows only shuffle-out ports of P_i. The logical structure of a STF1 with eight PEs is shown in Figure 1. There is a spare PE and there are redundant link as indicated by dotted lines. A thick line represents an exchange link or a bundle of 1 or 3 links in Figure 1(a). With this arrangement any single failure in SFT1 can be tolerated. Figure 1(b) is a similar diagram with duplicated links eliminated.

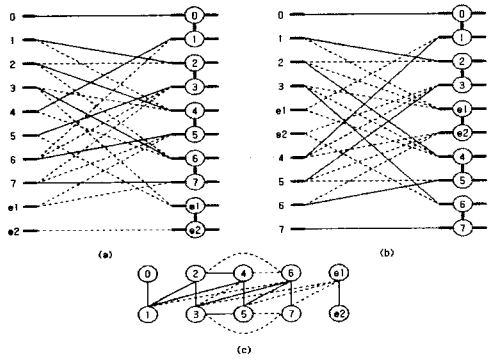
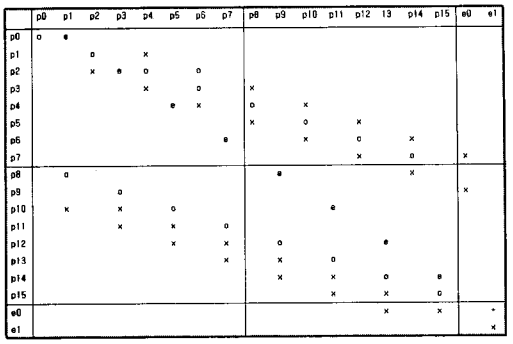


Figure 2 Single fault-tolerant static shuffle-exchange network using two extra PEs.

By adding one more extra PE the number of additional ports for each PE can be reduced, call it *SFT2*. Using two extra PEs connected by an exchange link reduces the maximum number of additional ports to four: two extra shuffle-out ports and two extra shuffle-in ports. Since any single faulty

Table 1 Connection table for *SFT2*



PE (P_{2i} or P_{2i+1} $(0 \le i < N/2)$) and PE connected by a exchange link can be replaced by two extra PEs, there is no need for the additional exchange ports. If a fault occurs in P_{2i}, P_{2i+1}, or a link attached to P_{2i} or P_{2i+1}, isolate P_{2i}, P_{2i+1}, and links attached to P_{2i} and P_{2i+1}, and use P_{2i+2} as P_{2i}, P_{2i+3} as P_{2i+1}, and so on. Equation 3 shows the shuffle-out connection links for P_i $(0 \le i \le N+1)$ in *SFT2*. Here, P_N and P_{N+1} represent two spare PEs:

$$SH2_i=\begin{cases} i & \text{if } i=0 \text{ or } i=N+1 \\ 2i, 2i+2 & \text{if } i=1 \\ 2i-3-N, 2i-i-N & \text{if } i=N \\ 2i-2, 2i+1-N & \text{if } i=N/2 \text{ or } i=N/2+1 \\ 2i-2, 2i, 2i+2 & \text{if } 1<i<N/2 \\ 2i-3-N, 2i-1-N, 2i+1-N & \text{if } N/2<i<N \end{cases} \quad (3)$$

As shown in Figure 2 and Equation 3, no additional shuffle ports are needed for P_0 and $P_{N+1}(P_{e2})$. Additional shuffle-in and shuffle-out ports are necessary for P_1 and $P_N(P_{e1})$. P_2 and P_{N-1} require two additional shuffle-out ports and one additional shuffle-in port. $P_{N/2}$ and $P_{N/2+1}$ require one additional shuffle-out port and two additional shuffle-in ports. The other PEs need two additional shuffle-in and shuffle-out ports. Equation 3 shows only shuffle-out ports of P_i. Figure 2 (a), (b), and (c) are isomorphic. Figure 2(c) does not show the duplicated links. As illustrated in Figure 2 (a) and (b), the configuration of the network is not affected by the location of the extra PEs: the only difference between network (a) and (b) is the logical number of each PE.

Table 1 shows the connection links of *SFT2*. The 'o', 'e', and 'x' represent the original shuffle link,

original exchange link, and extra shuffle link, respectively. The row connection represents the shuffle-out links and the column connections represent the shuffle-in links. The table clearly show that the maximum node degree of *SFT2* is seven.

Figure 3 (a) and (b) show single fault-tolerant *SSE* graphs represented by closed loop. Figure 3 (a) shows optimal *SSE* graph and (b) has better maximum node degree than *SFT2*. The problem, however, is that increasing the number of nodes increases the number of spare links and the worset-case node degree.

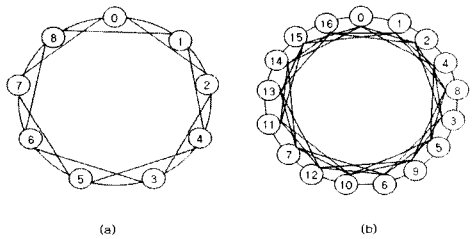


Figure 3 A single fault-tolerant SSE graphs represented by closed loop. (a) N=8 (b) N=16

3. Multiple Fault-Tolerant Static Shuffle-Exchange Network

The *SFT2* technique can be used to construct k fault-tolerant *SSE*(k-*SFT2*) networks. In order to tolerate k faulty elements, the network need at least $2k$ extra PEs and at most $4k$ extra shuffle ports for each PEs. In order to define the configuration of the shuffle connection links for each PEs, it is easy to define it in terms of the inverse shuffle connection.

For all even numbered PEs, P_{2i} ($0 \le i < N/2+k$),

- A shuffle-in port of P_0 is connected to a shuffle-out port of itself.
- The shuffle-in ports of P_{2j} ($k < j < N/2+k$) are connected to the shuffle-out port of P_{j-k}, P_{j-k+1}, ... , P_j, ... , P_{j+k-1}, P_{j+k}.
- The shuffle-in ports of P_{2j} ($0 < j \le k$) are connected to the shuffle-out port of $P_{j-(j-1)}$, $P_{j-(j-2)}$, ... , P_{j-1}, P_j, ... , P_{j+j}.
- The shuffle-in ports of P_{2j} ($j = N/2+k$) are connected to the shuffle-out port of P_{j-k}, P_{j-k+2}, ... , P_{j+k-2}, P_{j+k}.

For all odd numbered PEs, P_{2i+1},

- A shuffle-in port of P_{N+2k-1} is connected to a shuffle-out port of itself.
- The shuffle-in ports of P_{2j+1} ($0 < j < N/2-1$) are connected to the shuffle-out port of $P_{j+N/2}$, $P_{j+N/2+1}$, ... , $P_{j+N/2+2k-1}$, $P_{j+N/2+2k}$.
- The shuffle-in ports of P_{2j+1} ($N/2-1 < +j < N/2+k-1$)are connected to the shuffle-out port of P_{2j+1}, P_{2j+2}, ... , $P_{N+2(k-1)}$.

Table 2 Shuffle connection table of a 4 fault-tolerant static shuffle-exchange network. (*SFT2*, N=16, 8 extra PEs)

	p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15	e0	e1	e2	e3	e4	e5	e6	e7
p0	×																							
p1			×		×		×		×		×													
p2			×		×		×		×		×		×											
p3					×		×		×		×		×		×									
p4					×		×		×		×		×		×		×							
p5							×		×		×		×		×		×		×					
p6							×		×		×		×		×		×		×		×			
p7									×		×		×		×		×		×		×		×	
p8		×							×		×		×		×		×		×		×			
p9				×							×		×		×		×		×		×		×	
p10		×		×		×							×		×		×		×		×			
p11				×		×		×							×		×		×		×		×	
p12		×		×		×		×		×							×		×		×			
p13				×		×		×		×		×							×		×		×	
p14		×		×		×		×		×		×		×							×			
p15				×		×		×		×		×		×		×							×	
e0		×		×		×		×		×		×		×		×								
e1				×		×		×		×		×		×		×		×						
e2						×		×		×		×		×		×		×						
e3								×		×		×		×		×		×		×				
e4										×		×		×		×		×		×				
e5												×		×		×		×		×		×		
e6														×		×		×		×		×		
e7																								×

• The shuffle-in ports of P_{2j+1} $(j=0)$ are connected to the shuffle-out port of $P_{j+N/2}$, $P_{j+N/2+2}$, ... , $P_{j+N/2+2k}$.

Table 2 shows the shuffle connection of 4-*SFT2* with $N=16$ and eight extra PEs.

When there are no faults, P_0 to P_{N-1} are used as the functioning PEs and their associated links are emulate the circulated perfect shuffle. If P_{2j} or P_{2j+1} $(0\le j<N/2+k)$, or a link attached to P_{2j} or P_{2j+1} is found to be faulty, the faulty PE and its neighboring PE are isolated and the system reconfigured. There are two ways to reconfigure the system. The first method uses the logical number of the functioning PEs. Initially the logical numbers of PEs, P_L, are the same as the physical number. Once a faulty PE is detected and isolated, the logical numbers of the PEs are reset by the following equation;

$$L=\begin{cases} L' & \text{if } L'<2j \\ L'-2 & \text{if } L'>2j+1 \end{cases} \quad (4)$$

Then logically numbered P_0 to P_{N-1} are used as the functioning PEs. In order to select the proper shuffle-in and shuffle-out ports, all functioning PEs send their logical numbers to all neighbors. Once each PE obtains the logical numbers of its neighbors the selection of proper port is the same as with the original perfect shuffle connection scheme.

The other method uses the physical numbers of all faulty PEs. For simplicity, assume that all PEs have the same number of ports. The following algorithm reconfigures the *k-SFT2* network:

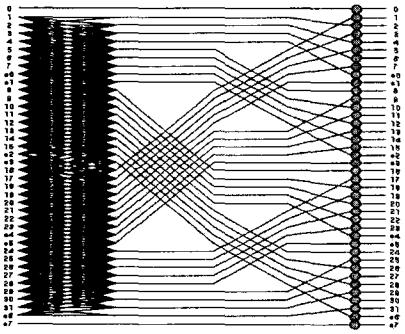


Figure 4 4 fault-tolerant SSE network decomposed into 4 modules

Algorithm RECONFIGURE-k-SFT2

• Input : A list of faulty PEs, $F=\{x_0, x_1, \ldots, x_{k-1}\}$

• Comments : Initially if there is no fault, for all currently active P_i, set the logical number of P_i to be the physical number of P_i, set shuffle-out port to $2i$ if $0\le i<N/2$ and set shuffle-out port to $2i+1-N$ if $N/2\le i<N$. Assume there are k fault in the system, then let the value of x_k be the physical number of the faulty PE such as $(x_0<x_1<x_2 \ \ldots x_{k-2}<x_{k-1})$. There are no exchange-pairs in F. First a faulty node or link is diagnosed by its immediate neighboring nodes. These local diagnosis results are forwarded by message passing ($O(\log_2 N)$ steps) to the central observer PE. Central observer broadcasts the physical numbers of the faulty PEs.

for all currently active P_i

(a) if its exchange-pair is in F then deactivate itself

/* Count the number of faulty PEs with physical number less than i */

(b) for $j=0$ to $k-1$

if $i\ge x_j$ then $n++$

else break

set the logical number (l) of P_i to $i-2n$

/* Compute the proper shuffle-out connection link */

(c.1) if $0\le l<N/2$ then

let q be $2l+2n$

for $j=0$ to $k-1$

if $q\ge x_j$ then $m++$

else break

let q be $q+2(m-n)$

(c.2) if $N/2\le l<N$ then

let q be $2l+1-N$

for $j=0$ to $k-1$

if $q\ge x_j$ then $m++$

else break

let q be $q+2m$

(c.3) while q is on F or exchange-pair of F

$q=q+2$

Set the shuffle-out port to q for all active P_i

The first part of the algorithm deactivates the currently fault free exchange pair in F. The second part of the algorithm sets the logical number of each PE by simply counting the number of PEs less than its own physical number. For example P_8 in Table 2 can obtain five possible logical numbers, (0,2,4,6,8). If there are two faulty PEs between P_0 and P_7, the logical number of P_8 will be 4. The third part of the algorithm selects a proper shuffle-out link from the maximum $2k+1$ shuffle-out links. For example, assume $F=(2,7,10,15)$ in Table 2, the logical number of P_8 is set to 4 by (b). The shuffle-out connection (q) is computed to 12 at first, but m is set to 3 by for-loop statement in (c.1), the value of q is set to 14 by (c.1). Since P_{14} is an exchange-pair of P_{15}, eventually the shuffle-out link of P_8 is connected to $P_{16}(E_0)$ by (c.3).

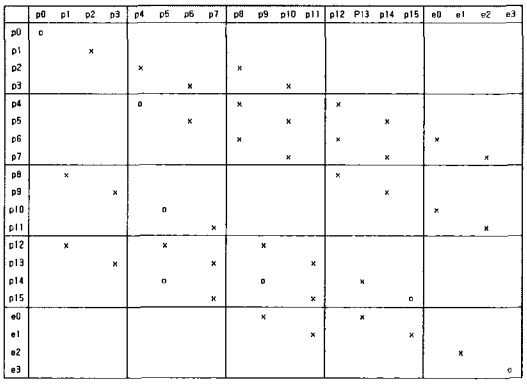
The advantage of the algorithm ***RECONFIGURE-k-SFT2*** is that there is no need for broadcasting the logical number of each PE during the recovery process. The complexity of the algorithm is $O(k)$ so that it does not depend on the number of processors in the system.

4. Decomposition of k Fault-Tolerant Static Shuffle-Exchange Network into Modules

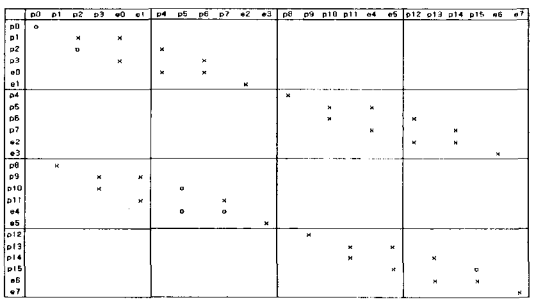
In 1991 Batcher[4] described the decomposition of a *SSE* network into modules of various sizes so it can be efficiently spread across the chips, boards, and racks. As indicated in Section 3, the configuration of a fault-tolerant network is not affected by the position of the extra PEs. When $M=2^m$, an M fault-tolerant *SSE* network (*M-SFT2*) can be decomposed into M identical modules. A 2^n+2M PE *SSE* network can be decomposed into 2^m ($2\le m\le n-2$) identical modules with $2^{n-m}+2$ PEs in each module. However, it is impractical to build an M fault-tolerant network if the number of modules (M) is large. Since there are only two PEs in each module, PEs from neighboring modules must be borrowed in case of more than one fault in a module. The system configuration is the same as a k fault-tolerant *SSE* network (Figure 4).

There are three ways to build a k fault-tolerant *SSE* network with N PEs and M identical modules ($N=2^n$, $M=2^m$, and $m<n$). The first method (*EXM*) is to build an k fault-tolerant *SSE* network by adding k extra modules. The second method (*EXP*1) is to build an k fault-tolerant *SSE* network with $2k$ extra PEs in each module: the maximum number of faults tolerated is k. The third method (*EXP*2) is the same as *EXP*1 in the number of extra PEs per module, but each module can independently tolerate k faulty elements.

Table 3 Shuffle connection table of 1 fault-tolerant *SSE* network built by *EXM* scheme.



Tabl 4 Shuffle connection table of 1 fault-tolerant *SSE* network built by *EXP*1 scheme.



EXM is the best when each module is small and the number of the modules is large. The network built by *EXM* uses $N+2^{n-m}k$ PEs, k extra modules, the maximum of $4k$ extra shuffle ports per PE, and can tolerate up to k faults.

*EXP*1 is the best when the size of each module is large and the number of the modules is small. The network built by *EXP*1 uses $N+2kM$ PEs, $2k$ extra PEs per module, up to $4k$ extra shuffle ports for each PE, and can tolerate up to k faults.

*EXP*2 uses a large number of shuffle ports. The network built by *EXP*2 uses $N+2kM$ PEs, $2k$ extra PEs per module, up to $7k-1$ extra shuffle ports per PE, and can tolerate up to k faults per module. Tables 3, 4, and 5 show the shuffle connection table for *EXM*, *EXP*1, and *EXP*2, where $N=16$, $M=4$, and $k=1$. An 'x' represents a shuffle connection and an 'o' represents a duplicated shuffle connection or self connection.

Table 5 Shuffle connection table of 1 fault-tolerant *SSE* network built by *EXP*2 scheme.

	p0	p1	p2	p3	e0	e1	p4	p5	p6	p7	e2	e3	p8	p9	p10	p11	e4	e5	p12	p13	p14	p15	e6	e7
p0	o																							
p1			x		x																			
p2			o				x		x															
p3					x				x		x													
e0							x		x															
e1									x		x													
p4													x		x									
p5															x		x							
p6													x		x				x		x			
p7															x		x				x		x	
e2																			x		x			
e3																					x		x	
p8		x		x																				
p9				x		x																		
p10		x		x				o		o														
p11				x		x				x														
e4								o		o														
e5										x		x												
p12														x		x								
p13																x		x						
p14														x		x			x					
p15																x		x				o		
e6																			x		x			
e7																								x

A comparison of the *EXM* and *EXP*1 methods with *EXP*2 reveals an interesting point. If the number of the extra PE are same, the *EXM* and *EXP*1 approaches are the same in terms of hardware overhead: the only difference is the system topology.

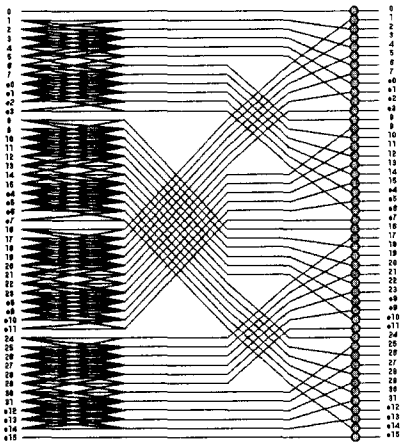


Figure 5 2 fault-tolerant SSE network decomposed into 4 modules. Each individual modules tolerate single fault, and the overall network tolerate additional single fault.

The *EXP*2 scheme can tolerate k faults per module, i.e. up to kM faults, with only two extra ports per PE. The network built by *EXP*2 will be very reliable if k is small and the number of modules is large.

By combining a k fault-tolerant *SSE* network built by the *EXM* scheme with an l fault-tolerant *SSE* network built by the *EXP*2 scheme one can build an $(k+1)(l+1)-1$ fault tolerant *SSE* network (Table 6). This network consists of $N+2(M+k)l+2^{n-m}l$ PEs, up tp $3k(7l+1)$ shuffle ports per PE, and can tolerate up to $lM+(l+1)k$ faults. Unfortunately, this scheme requires a large number of extra shuffle ports.

Table 6 Comparison of additional hardware between Rennels and a fault-tolerant SSE network built by SFT2 scheme

Node Size	Hypercube		*EXP*2 (8)		*EXP*2 (16)	
	nodes	edges	nodes	edges	nodes	edges
16	20	52	20	56	18	48
32	40	124	40	129	36	119
64	80	288	80	278	72	264
128	160	656	160	579	144	557
256	320	1472	320	1184	288	1146
512	640	3264	640	2397	576	2327
1024	1280	7168	1280	4826	1152	4692

Another method is to build a network combining the *EXP*1 and *EXP*2 schemes. Each module has $2k$ extra PEs, and each PE has the maximum $5k$ extra port if k is even, $4+10(k\ DIV\ 2)$ extra port if k is odd. Each module of the network not only tolerates $\left|\frac{k}{2}\right|$-faults but the overall network tolerates an additional $\left|\frac{k}{2}\right|$-faults (Figure 5).

5. Conclusion

This paper shows fault-tolerant static shuffle-exchange networks. The number of extra ports per PE does not depend on the size of the network, only on the number of faults tolerated. A k fault-tolerant *SSE* network (k-*SFT*2) needs at least $2k$ PEs and at most $4k$ extra shuffle ports. A k fault-tolerant *SSE* network built by the *EXP*1 and *EXP*2 schemes can improve the reliability of the system using modulization technique. The reconfiguration time for the multiple tolerant *SSE* network defined in Section 3 is minimal.

Table 6 compares the additional hardware of the fault-tolerant *SSE* network built by *SFT*2 scheme to the fault-tolerant binary hypercube proposed by Rennels[5]. The number in parentheses represents the number of PEs in each module. This table clearly shows that *SFT*2 can achieve the near-optimality in the number of spare edges. Further research is required to see if these single fault-tolerant SSE networks (*SFT*1 and *SFT*2) are optimal.

Bibliography

[1] H.S. Stone High-Performance Computer Architecture, Second Edition, Addison Wesley, Reading MA, 1990.

[2] H.S. Stone, "Parallel processing with the perfect shuffle", IEEE Transactions on Computers vol C-20, 1971, pp 153-161

[3] K.E. Batcher, "Low-cost Flexible Simulation with the Static Perfect Shuffle Network", Frontiers, Oct. 1992.

[4] K.E. Batcher, "Decomposition of Perfect Shuffle Networks", Proceeding of the 1991 International Conference on Parallel Processing, CRC Press, Boca Raton FL, vol. I, 1991, pp 255-262.

[5] D.A. Rennels, "On implementing fault tolerance in binary hypercubes", Proceedings of 16th International Symposium on Fault-Tolerant Computing, July 1986, pp 344-349.

최 홍 인
1982년 한성대학교 영어영문학과 학사 1991년 Kent State University 컴퓨터과학 석사. 1997년 Kent State University 컴퓨터과학 박사. 1999년~2001년: 아코텍(주) 연구팀장. 2001년~현재 단국대학교 강의전임강사. 관심분야는 병열처리 컴퓨터 구조, 병열처리 알고리듬, 분산네트워크