論文2002-39SC-1-4

다단계 상호연결망의 고착고장에 대한 효율적인 고장진단 기법

(Efficient Fault Diagnosis of Stuck-at-Faults in Multistage Interconnection Networks)

金 永 宰 *, 曺 光 鉉 **

(Young-Jae Kim and Kwang-Hyun Cho)

요 약

본 논문은 다중 컴퓨터 시스템(multicomputer system)에서 사용되는 상호연결망(interconnection network)의 일종인 다단계 상호연결망(multistage interconnection network)에서의 고착고장(stuck-at fault)과 관련된 고장진단(fault diagnosis)에 대해 고찰한다. 지금까지 연구된 바에 의하면 다단계 상호연결망의 고장을 검출하고 위치를 찾기 위해서는 각 고장의 유형에 따라 서로 다른 다수의 과정을 거치거나 몇 번의 테스트를 통한 진단기법을 필요로 한다. 이에 본 논문에서는 단일과정을 통해 대표적인 고착고장의 유형을 검출하고 위치를 찾아내는 고장진단 알고리즘을 제안한다. 즉, 고착고장이 존재하는 임의의 다단계상호연결망에 대하여 고장 스위치 소자의 행과 열의 위치정보 및 고착고장의 유형진단 알고리즘을 제시한다. 마지막으로 다단계 상호연결망의 일종인 16×16 베이스라인 망(baseline network)에서 고착고장이 발생한 스위치의 위치와 유형을 찾는 과정을 통해 제안하는 알고리즘의 효용성을 검증한다.

Abstract

This paper is concerned with the fault diagnosis for stuck-at faults of a multistage interconnection network(MIN) which is a kind of interconnection networks in multicomputer systems. Up to the present, a fault diagnosis scheme has dealt with a fault model of all types, which results in complicated algorithms. In the literature, it is shown that a number of steps and computation are required for the fault detection and isolation algorithms for a class of MINs. In this paper, we propose a simple and easily implementable algorithm for the detection and isolation of the stuck-at fault in MIN. Specifically, we develope an algorithm for the isolation of the source fault in switching elements whenever the stuck-at fault is detected in MINs. After all, the proposed algorithm is illustrated by an example of 16×16 baseline networks of MINs.

Ⅰ. 서 론

최근 대규모 직접회로에 관련한 설계기술의 발전으로 고성능 컴퓨터가 등장하고 이들 시스템을 서로 연결하여 계산능력을 증대시킨 병렬 다중처리 컴퓨터 시스템(parallel multiprocessing computer system)은, 컴퓨터에 의한 고성능의 데이터 처리를 가능케 하였다. 컴퓨터의 높은 데이터 처리능력은 정보처리 형태에 질적 변화를 일으켜, 클라이언트-서버(client-server)방식에 의한 네트워크 컴퓨팅이 눈에 뛰게 보급되었다. 다중 컴퓨터 시스템에서 각 노드들 사이를 연결하거나 네트워크 상에서 교환기를 통한 고속의 정보 전송을 위해 스위치를 상호연결 시키는 상호연결망은 전체시

* 學生會員, ** 正會員, 蔚山大學校 電氣·電子·情報시스템工學部

(School of Electrical Engineering, University of Ulsan.)

※ 이 논문은 2001년도 한국학술진흥재단의 지원에 의하여 연구가 되었음. (KRF-2001-0410-E00279)

接受日字:2001年2月14日, 수정완료일:2001年11月15日

스템의 성능을 결정하는 주요 요인들 중의 하나이다. 특히, 다단계 상호연결망(MIN: Multistage Interconnection Network)은 알려져 있는 바와 같이 많은 종류의 네트워크들이 연구, 제안되고 있다[1]. 일반적으로 N×N MIN은 $N(=a^m)$개의 입력과 N개의 출력 사이의 연결을 제공한다. MIN은 $m(=\log_a N)$개의 단계로 구성되며, 각 단계는 N/a개의 $a\times a$ 크로스바(crossbar) 스위치들로 구성된다[2]. 정상적으로, 스위치 소자는 목적지 정보(destination information)의 논리 값에 따라 두 개의 가능한 상태를 갖는다. 그림 1은 2×2 스위치에서 "통과(through: T)"와 "교차(cross: X)"의 서로 다른 두 연결 상태를 보여주고 있다[3].

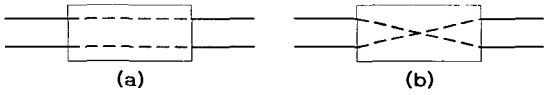

그림 1. 스위치의 연결상태와 고착고장: (a) T-상태(T-형 고착고장), (b) X-상태(X-형 고착고장).
Fig. 1. Connection state of a switch and stuck-at fault: (a) T-state(stuck-at T), (b) X-state (stuck-at X).

MIN시스템의 신뢰도를 높이기 위해서는 고장에 대한 대책을 마련해야 한다. 그리고 고장이 일어나는지 계속 감시하면서 고장이 발생했을 때 이를 확인하고 심각성을 파악해서 그때에 필요한 조치를 취해 주어야 한다. 고장을 조기 진단하여 건강한 시스템으로 대체하는 것이 MIN시스템의 내고장성을 향상시키기 위해 선행되어야할 조치이다. 이에 본 논문은 MIN을 구성하고 있는 스위치 소자에 발생하는 고장을 진단하는데 중점적으로 연구하였다. 스위치 소자에 발생하는 대부분의 고장은 고착고장(SaF: stuck-at fault)이다. 그림 1에서 스위치 소자의 연결선(link line) 즉, 입력선과 출력 선에 흐르는 논리값이 0또는 1에 관계없이 항상 0(1)로 고정된 논리값을 나타낼 때를 고착고장이라 한다. "T" 상태로 고정되었을 때를 T형 고착고장(SaT: stuck-at straight)이라 하며, "X"상태로 고정되었을 때를 X형 고착고장(SaX: stuck-at exchange)이라 정의한다[3,4]. 이러한 고장을 진단하여 정보를 제공하므로써 MIN의 내고장성을 향상시킬 수 있다. 현재까지 다양한 방법에 의한 고장진단 기법이 연구되었으나 주로 사용되어지는 방식은 테스트에 의한 방법과 각 고장의 경우에 따라 여러 단계의 알고리즘을 수행하는 진단 기법 등이 있다[4~6]. 테스트에 의한 방법은 과거부터 많이 연구되어온 방법으로 수 차례의 테스트를 통해 고장이 발생한 부분을 진단해 낸다. 이 방법은 테스트의 횟수를 줄이는 방향으로 연구가 진행되고 있으며, 제안한 테스트의 횟수 후에도 확실히 고장을 찾지 못하는 경우가 발생할 확률이 있다. 알고리즘에 의한 고장 진단 기법은 모든 유형의 고장을 다루다 보니 알고리즘이 길어지고 복잡해진다. 그래서 MIN의 네트워크방식이 달라짐에 따라 다른 알고리즘을 제시하여야만 한다. 대부분 MIN에서 발생하는 고장은 고착고장으로 취급될 수 있으며[3], 고착고장만을 고려할 경우 알고리즘의 길이를 효과적으로 줄일 수가 있다. 본 논문에서는 이러한 고착고장에 대해 라우팅 비트와 목적지 주소의 2진 코드를 비교하여 고장이 존재하는 단(stage)을 검출하고 고장의 유형을 진단하는 방법의 알고리즘을 제시하였다. II장에서는 고착고장, 단 과 단 사이의 링크함수, 스위치의 입출력포트와 라우팅 비트 등의 관계를 논리적 표현으로 정의하였다. 이를 통해 고장진단 알고리즘에서 논리적 연산을 수행하여 빠른 고장 검출을 가능하도록 하였다. 또한, 본 논문에서는 테스트의 횟수를 줄이기 위해서 라우팅 비트 r이 고착고장이 발생한 스위치를 통과했을 때 원하는 목적지 정보 d와 같다면 이는 고장으로 검출하지 않는다. 즉 실제 라우팅에 영향을 주는 고장 스위치만을 검출하여 테스트 횟수를 줄이는 것이다. 이렇게 함으로서 실제 라우팅에 영향을 미치는 스위치만을 선별적으로 선택하여 대체 스위치로의 빠른 교체가 실시간으로 가능해진다.

II. 다단계 상호연결망과 고착고장 모델

1. 다단계 상호연결망

본 논문에서는 고장진단 문제를 고려함에 있어서 다단계 상호연결망 중의 한 모델인 베이스라인 망을 이용하였다. 그림 2에서 베이스라인 망은 N개의 시작지 주소(source)와 N개의 목적지 주소(destination)들로 구성된다. 시작지 주소는 단계 0에 있는 스위칭 소자에 연결되며 목적지 주소는 단계 $m-1$에 연결된다. $N=2^m$, 이때 m은 망을 구성하는 단계 수이며 각 단계는 $N/2$ 개의 2×2 스위칭 소자로 구성된다. 그리고 스위칭 소자의 동작상태는 4개의 입·출력 단자들의 결합에 따라 16종류가 있을 수 있으나 그림 1에서 보

는 바와 같이 통과, 교차에 의한 동작 상태를 망을 구성하는 스위칭 소자들의 정상상태로 간주한다[4].

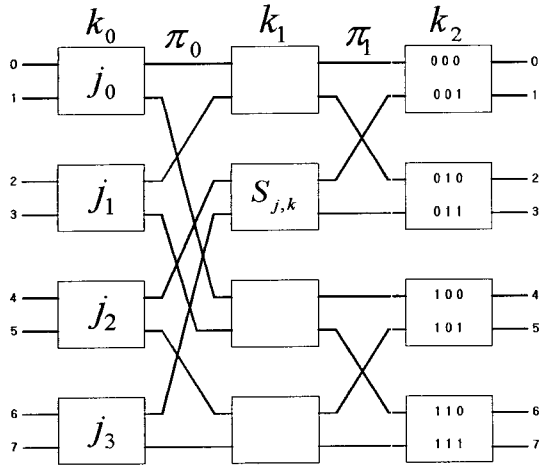

그림 2. N=8인 베이스라인 망.
Fig. 2. A baseline network with N=8.

그림 3에서 2×2 스위치는 라우팅비트(routing bit)의 1비트를 참조하여 라우팅 비트 r이 0이면 위쪽 포트(port)로, 1이면 아래쪽 포트로 각각 출력한다. 라우팅 비트 $r=r_{m-1}r_{m-2}\cdots r_1r_0$일 때, r는 각 단에서의 라우팅 정보를 가지게 된다. 라우팅 비트는 목적지 정보와 같으며, 위쪽 포트는 짝수 주소 아래쪽 포트는 홀수 주소를 갖는다. 스위치의 입·출력포트 주소($A_i=A_o=b_{m-1}b_{m-2}\cdots b_1b_0$) 및 라우팅 비트는 이진 표현을 사용한다. 그림 3에서 A_i는 라우팅 비트가 입력되는 입력 측 포트 주소이며, A_o는 라우팅 비트에 의해 결정된 출력 측 포트 주소이다.

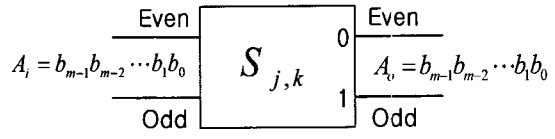

그림 3. 2×2 스위치
Fig. 3. 2×2 Switch.

각 스위치 소자는 그림 2에서 스위칭 함수 $S_{j,k}$로 위치 정보를 표현하며, MIN에서 j는 스위치의 행의 위치를, k는 열의 위치 정보를 나타낸다. 이때 $j=\{0,1,2,\cdots,N/a\}$, $k=\{0,1,2,\cdots,m\}$ 이다. 그림 2와 같은 MIN에서 가지는 구조적 특성을 아래와 같은 정의 및 정리로 나타낼 수 있다.

정의 1 : 스위치 $S_{j,k}$의 A_i를 k단의 라우팅 비트 r_k의해 A_o를 정의한 식은

$$A_i \xrightarrow{S_{j,k}(r_k)} \left\{ \begin{array}{l} r_k=0 \text{ 이면, } A_o=even \\ r_k=1 \text{ 이면, } A_o=odd \end{array} \right\} \quad (1)$$

이다.

정의 2 : MIN의 k단계(stage)에 해당되는 k번째 라우팅 비트를 r_k라 하면 라우팅 비트에 따른 스위치 형태는 $r_k=A_i(b_0)=A_o(b_0)$이면 "통과 상태: T"라고 하며, $r_k \neq A_i(b_0)=A_o(\overline{b_0})$이면 "교차 상태 : X"라고 정의한다.

네트워크에서 각 단계들 사이의 상호 연결은 그림 2에서 보는 바와 같이 링크 함수(link function) π_i에 의해 결정되며, $i=\{0,1,2,\cdots,m-1\}$이다. 연결 함수 π_i의 대표적인 연결 방식은 셔플 교환(shuffle exchange)과 나비교환(butterfly exchange)이 있다. 셔플 교환은 네트워크의 형태에 따라 두 가지로 나뉘어 지며, MIN의 형태가 베이스라인 망이면 1-비트 순환 우천이(1-bit circular right shift)방식을, 오메가 망(omega network)이면 1-비트 순환 좌천이(1-bit circular left shift)방식을 사용한다[2,7]. k번째 단계에 위치한 임의의 스위치 출력포트 A_o^k는 k번째에 위치한 링크함수 π_k에 의해 다음 단 $k+1$에 위치한 임의의 입력포트 A_i^{k+1}로 연결된다.

정의 3 : k단에 위치한 임의의 스위치 출력포트 A_o^k의 $m-$비트 이진수를 $b_{m-1}b_{m-2}\cdots b_1b_0$로 표현 할때, $A_o^k \xrightarrow{\pi_k} A_i^{k+1}$ 위한 링크함수 π_k를 정의하면

$$\pi_k(A_o^k)=b_{m-1}b_{m-2}\cdots b_{m-k}b_0b_{m-k-1}\cdots b_2b_1 \quad (2)$$

이다[7]. 이와 같은 형태의 π_k를 1-비트 순환 우천이 방식이라 부른다.

정의 4 : k스테이지 링크함수 π_k 의해 얻은 정보(A_i^k)를 가지고 각 입력포트의 이진코드워드 표현의 십진수표현을 n이라고 하면 현재 전송 동작을 수행하고 있는 스위치의 행 위치 함수 $j(A_i^k)$는

$$j(A_i^k)=\left\{ \begin{array}{l} A_i^k=even \text{ 이면, } j=\frac{n}{2} \\ A_i^k=odd \text{ 이면, } j=\frac{n-1}{2} \end{array} \right\} \quad (3)$$

이다.

정리 1 : 베이스라인 네트워크에서 주어진 라우팅 비트에 의해 시작지 주소로부터 목적지 주소 사이에는 단 하나의 경로만이 존재한다. 그리고 현재 전송 동작을 수행하고 있는 스위치의 위치 정보를 알 수 있다.

(증명) 단계 k에 있는 스위칭 소자는 각 단계의 라우팅 비트 r_k에 따라 소자의 출력이 결정되며 이것은 정의 1을 따른다. 이후에 정의 3의 상호 연결 함수를 적용시키므로써 다음 단계에 있는 스위칭소자의 입력으로 연결된다. 스위치의 연결 형태는 정의 2에 의해 알 수 있고, 현재 전송 동작을 하고 있는 스위치 소자의 정확한 위치는 정의 4에 의해 얻을 수 있다. 여기서, 하나의 시작지 주소로부터 주어진 라우팅 비트 r에따라 스위칭 함수 및 상호 연결 함수가 결정되면 전송 중인 r의 위치를 알 수 있으며, 두 함수에 의해 결정되는 목적지 주소는 유일하다. 그러므로 하나의 스위치에는 하나의 라우팅 비트 태그(tag)만이 존재하고, MIN에서는 언제나 하나의 전송경로 만을 갖는다.

2. 고착고장

라우팅 비트 r에 상관없이 스위치 형태가 그림 1의 어느 한가지 형태로 고정되어 항상 하나의 논리 값만을 가지게 되면 고착고장이라고 정의한다.

정의 5 : 고착고장이 발생했을 때 r의 값에 상관없이 스위치의 입·출력포트 주소가 항상 같으면 SaT, 출력 주소의 b_0값이 입력 주소의 b_0값에 항상 보수를 가지면 SaX이다. 이는 다음의 식으로 정의한다.

$$A_i \xrightarrow{S_{j,k}(r_k)} A_o \tag{4}$$

$$b_{k-1}\cdots b_0 \xrightarrow{S_{j,k}(r_k)} \left\{ \begin{array}{l} r_k=0 \text{ 이면, } b_{k-1}\cdots \overline{b_0} \\ r_k=1 \text{ 이면, } b_{k-1}\cdots b_0 \end{array} \right\}$$

단, A_i의 $b_0=0$이다 (5)

$$b_{k-1}\cdots b_0 \xrightarrow{S_{j,k}(r_k)} \left\{ \begin{array}{l} r_k=0 \text{ 이면, } b_{k-1}\cdots b_0 \\ r_k=1 \text{ 이면, } b_{k-1}\cdots \overline{b_0} \end{array} \right\}$$

단, A_i의 $b_0=1$이다. (6)

위의 두 식을 정리하면

$$b_{k-1}\cdots b_0 \xrightarrow{S_{j,k}(r_k)} b_{k-1}\cdots b_0 \text{ : SaT} \tag{7}$$

$$b_{k-1}\cdots b_0 \xrightarrow{S_{j,k}(r_k)} b_{k-1}\cdots \overline{b_0} \text{ : SaX} \tag{8}$$

정리 2 : 고장이 발생한 스위치는 정의 1과는 무관하게 라우팅이 이루어지며, 언제나 스위치는 고정된 논리값만을 가진다.

(증명) 임의의 스위치에 라우팅 태그 r_k가 입력되면 r_k에 의해 스위치의 출력이 결정되어진다. 그러나 고착고장이 발생하면 스위치는 이미 정해진 출력값을 가지고 있으므로 정의 5와 같이 r_k에는 무관하게 언제나 동일한 값을 출력한다. 정의 1에서 정상 적인 스위칭은 r에 관한 함수지만, 고착 고장이 발생한 스위치는 정의 5에서 보는 바와 같이 r에 관계없이 이미 정해진 고정된 논리값을 가지게 된다.

Ⅲ. 고장진단 알고리즘

본 절에서는 고장이 발생한 MIN에서 고장의 위치를 찾아내고, 고장의 유형을 진단하는 알고리즘을 단일과정으로 제시한다. 알고리즘의 처음은 각 단계에서의 라우팅 비트의 정보를 관찰하고 수집하여 최종 적으로 도달한 목적지 정보와 비교하여 서로 다른 비트의 위치를 찾아낸다. 이때 라우팅 비트 r과 목적지 정보 d의 서로 다른 두 비트가 x번째 단계에 존재할 때 이를 나타내는 식은 (9)와 같다. 이 식의 첫 번째 행 $F^1_m=\{0,1,2,\cdots x_i \cdots m\}$ 은 r_i와 d_i에 서로다른 비트가 존재할 때 해당 단 번호를 통해 고장이 발생한 스위치의 열(column)위치 정보를 나타낸다.

$$F_m=\begin{bmatrix} \text{단계 번호} \\ \text{라우팅 비트 } r \\ \text{목적지 정보 } d \end{bmatrix} = \begin{bmatrix} 0 & 1 & \cdots & x_i & \cdots & m-2 & m-1 \\ r_{m-1} & r_{m-2} & \cdots & r_i & \cdots & r_1 & r_0 \\ d_{m-1} & d_{m-2} & \cdots & d_i & \cdots & d_1 & d_0 \end{bmatrix} \tag{9}$$

다음에는 F_m를 근거로 고장이 존재하는 입력포트를 정의 1과 정의 3에 의해 다음 단으로 전송이 진행되어 가면서 찾는다. 그리고는 이 입력포트 주소의 정보를 가지고 고장이 존재하는 행(row)값을 찾는다. 이상의 과정으로 고장이 존재하는 스위치의 위치를 찾은 후 고착고장의 형태를 정의 5에 의해 진단한다. 이러한 일련의 과정을 수도코드(pseudo code)로 나타내면 다음과 같으며, 사용되는 기호는 표 1을 따른다.

```
Begin
  for k=0 to m
    d= d_{m-1}d_{m-2}···d_1d_0
```

$r = r_{m-1} r_{m-2} \cdots r_1 r_0$

if ($r_k == d_k$)

$k \notin F_m$

else ($r_k \neq d_k$)

$k \in F_m$

end

return ($k = k+1$)

end

for $k=0$ to m

$A_i^k = b_{m-1} b_{m-2} \cdots b_1 b_0$

$r = r_0 r_1 \cdots r_{m-2} r_{m-1}$

$F_m^1 = \{0, 1, 2, \cdot \cdot x_i \cdot \cdot m\}$

$j = \varepsilon$

$A_i^k \xrightarrow{S_{j,k}(r_k)} A_o^k$

if ($r_k == 0$)

$A_o^k == even$

$\alpha = 2j$

$(\alpha)_{decimal} \Rightarrow (A_o^k = b_{m-1} \cdots b_0)_{binary}$

else ($r_k == 1$)

$A_o^k == odd$

$\alpha = 2j+1$

$(\alpha)_{decimal} \Rightarrow (A_o^k = b_{m-1} \cdots b_0)_{binary}$

end

return ($k = k+1$)

$k = k+1$

$A_o^k \xrightarrow{\pi_k} A_i^\beta$

$(A_i^\beta = b_{m-1} \cdots b_0)_{binary} \Rightarrow (n)_{decimal}$

if ($\beta \in F_m^1$)

if ($A_i == even$)

$A_i^k \xrightarrow{S_{j,k}(r_k)} A_o^k$

if ($r_k == 0$)

$A_o == odd$

Stuck-at X Fault

else ($r_k == 1$)

$A_o == even$

Stuck-at T Fault

end

else ($A_i == odd$)

$A_i^k \xrightarrow{S_{j,k}(r_k)} A_o^k$

if ($r_k == 1$)

$A_o == even$

Stuck-at X Fault

else ($r_k == 0$)

$A_o == odd$

Stuck-at T Fault

end

end

if ($n == even$)

$j = \frac{n}{2}$

else ($n == odd$)

$j = \frac{n-1}{2}$

return (h)

end

else ($\beta \notin F_m^1$)

if ($n == even$)

$j = \frac{n}{2}$

else ($n == odd$)

$j = \frac{n-1}{2}$

return (j)

end

end

end

end

표 1에서 d, r, r_k, d_k, A_i^k, A_o^k는 2진 코드워드를 사용하며 k, m, F_m^1, j, n, α, β, ε는 10진수를 사용한다.

표 1. 수도코드와 그림 4의 기호에 관한 정의 및 설명

Table 1. Definition and explanation of the symbol in pseudo code in Figure 4.

기호	정의	설명
π_k	$A_o^k \xrightarrow{\pi_k} A_i^\beta$	k번째에 위치한 연결 함수이다.
j	$\{0, 1, \cdots k \cdots N/a\}$	스위치의 행 위치를 나타내는 상수이다.

기호	정의	설명
$S_{j,k}$	$A_i \xrightarrow{S_{j,k}(r_k)} A_o$	j행 k열에 위치한 스위치로서 r_k에 의해 전송이 수행된다.
n	$(A_i)_2 \Leftrightarrow (n)_{10}$	A_i의 2진 코드워드를 십진수로 변환시킨 값이다.
α	$(A_o)_2 \Leftrightarrow (\alpha)_{10}$	A_o의 2진 코드워드를 십진수로 변환시킨 값이다.
β	$k+1$	k단에서 모든 라우팅이 이루어진 후의 그 다음 단을 나타낸다.
ε	행 위치 초기 값	1열에 위치한 스위치에서 처음 라우팅비트가 입력이 되는 스위치의 행 위치 값이다.
d	$d_{m-1}d_{m-2}\cdots$ $\cdots d_1d_0$	목적지 주소로서 m-bit로 구성된다.
r	$r_{m-1}r_{m-2}\cdots$ $\cdots r_1r_0$	m 개의 라우팅 비트로서 스위치의 라우팅을 제어한다.
k	$\{0,1,\cdots k\cdots m\}$	r, d에서 k번째 위치한 비트 또는 임의의 특정 단을 나타낸다. 스위치의 열의 위치를 지칭한다.
r_k	$r_{m-1}r_{m-2}\cdots$ $r_k\cdots r_1r_0$	r에서 k번째 위치한 라우팅 비트로서 k번째 단에 위치 한 스위치의 라우팅 정보를 가지고 있다.
d_k	$d_{m-1}d_{m-2}\cdots$ $d_k\cdots d_1d_0$	d에서 k번째 위치한 비트이다.
m	$\log_a N$	$N\times N$ MIN의 총 단의 수이다.
F_m	[단계 번호, 라우팅 비트 r, 목적지 정보 d]	r과 d의 2진 코드워드를 비교하여 서로 다른 비트의 위치(스테이지 번호)를 찾는다.
F^1_m	$\{0,1,\cdots x\cdots, m\}$	$F(x)$에 의해 나온 결과로서 고장이 발생한 단의 집합이다. 이것은 고장이 발생한 스위치의 열 정보이다.
A_i^k	$b_{m-1}b_{m-2}\cdots$ $b_k\cdots b_1b_0$	k번째(열)단에 위치한 임의의 스위치에 있는 입력포트 주소이다.
A_o^k	$b_{m-1}b_{m-2}\cdots$ $b_k\cdots b_1b_0$	스위치에서 라우팅이 이루어진 후의 k번째(열)단에 위치한 스위치의 출력포트 주소이다.

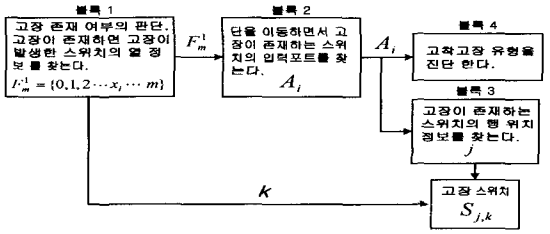

그림 4. 고장진단 블록도.
Fig. 4. Block diagram of the fault diagnosis.

그림 5는 수도코드와 블록도로 나타낸 이상의 고장 진단 과정을 흐름도로 나타낸 것이다.

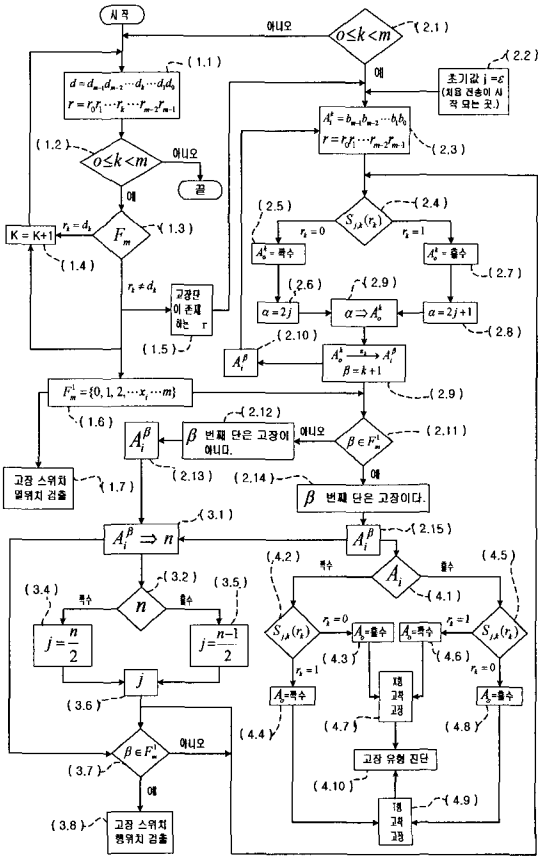

그림 5. 고장진단을 위한 흐름도
Fig. 5. Flow chart of the fault diagnosis procedure.

수도 코드로 보인 고장진단 알고리즘의 수행과정을 개략적인 블록도로 나타내면 크게 4개의 블록으로 나눌 수가 있다. 블록 1에서는 스위치의 고장 발생여부를 판단하여 고장이 발생한 스위치의 열 정보 F^1_m를 찾는다. 이때 F^1_m는 블록 2에 제공이 되어 라우팅 비트에 의해 단을 이동하면서 그 단이 F^1_m에 포함되는지

여부를 찾아내고 포함이 되면 고장 스위치의 입력포트 A_i를 얻는다. 이때 얻은 A_i는 블록 3과 블록 4에 제공되어 고장의 유형과 고장스위치의 위치를 찾게 된다. 각 블록이 수행하는 역할은 그림 4와 같이 나타낼 수가 있다.

그림 5의 (a.b)표기에서 a는 각 블록의 번호이고, b는 알고리즘 흐름의 순서이다. 그리고 그림 5의 고장진단 과정을 상세히 설명하면 다음과 같이 기술 할 수 있으며, 모든 기호는 표 1을 따른다.

1. 고장위치검출

단계(1.1)에서는 d와 r의 모든 정보를 수집하여 단계(1.2)에 보인 바와 같이 m번 루프를 반복하면서 단계(1.3)에 의해 고장이 존재하는 단을 찾은 후 단계(1.6)에 저장한다. 이때 루프를 한번 반복 할 때마다 단계(1.4)를 수행한다. 고장 단의 위치는 단에서 고장이 발생한 스위치의 열 위치 정보임을 단계(1.7)에 알려준다. 단계(1.5)에서 고장이 발생한 단을 포함하는 비트가 하나라도 존재하는 라우팅 비트(r)라면 단계(2.3)에 알려준다. 단계(2.3)에서는 전송이 처음 이루어진 스위치의 위치와 입력포트 주소를 통보 받는다. 그리고는 이 루프는 단계(2.1)에 보인 바와 같이 m번 루프를 반복한다. 단계(2.3)에서 받은 정보인 입력 스위치 주소와 라우팅비트를 가지고 단계(2.4)에 의해 스위치의 출력포트가 결정된다. 만약 $r_k=0$이면 단계(2.5)의 결과를 가지게 되어 단계(2.6)에 있는 식에 의해 출력포트의 십진수 값을 알 수가 있다. $r_k=1$인 경우도 앞과 동일한 과정인 단계(2.7), 단계(2.8)을 거치게 된다. 이렇게 얻은 출력포트의 십진수 값을 단계(2.9)에서 2진수 값으로 변화시킨다. 그러면 이 출력포트 주소는 단계(2.9)의 연결 함수에 의해 다음 단의 입력포트 주소로 이동이 된다. 이렇게 얻은 입력포트 주소는 단계(2.10)을 거친 후 단계(2.3)에 보내어지게 된다. 위의 과정을 계속 반복하면서 다음 단으로 계속 이동이 되어진다. k단에서 이동이 된 다음 단 $k+1$을 β라 하고, 단계(2.11)에 의해 적합하면 단계(2.14)를, 부적합하면 단계(2.12)를 선택하게 된다. 만약 단계(2.12)를 선택한 경우는 β번째 단이 고장이 아니므로 단계(2.13)을 거친 후 단계(3.1)에서 2진 입력포트 주소를 십진수 n으로 바꾼다. n이 짝수이면 단계(3.4)를, 홀수이면 단계(3.5)를 거쳐서 단계(3.6)의 j를 구한다. 단계(3.7)에서 고장스위치 여부를 판단하여 고장이 아니면 j를 과정(2.4)에 알려준다. 단계(2.14)를 선택한 경우에는 과정(2.15)를 거친 후 과정(3.1)에서 과정(3.7)까지는 단계(2.12)를 선택한 경우와 동일한 과정을 수행하게 된다. 이때 과정(3.7)에서 β가 F_m^1에 포함되는 경우이면 j는 고장스위치의 행 정보임을 통보 받는다.

2. 고장유형검출

1) X형 고장검출

단계(2.15)에서 β번째 단이 고장 단이므로 고장 유형을 진단하기 위해 단계(4.1)을 수행한다. 단계(4.1)에서 짝수이면 단계(4.2)를 홀수이면 단계(4.5)를 수행한다. 만일 단계(4.2)에서 $r_k=0$이거나 단계(4.5)에서 $r_k=1$이면 X형 고착고장임을 진단한다.

2) T형 고장검출

단계(4.2)에서 $r_k=1$이 되어 단계(4.4)를 수행 하여 T형 고착고장임을 진단한다. 또한 단계(4.5)에서 $r_k=0$일 때 위와 동일한 과정인 단계(4.8)등을 통해 고장의 유형이 T형임을 진단할 수가 있다.

Ⅳ. 고착고장 진단의 예

다음은 앞 절에서 제안한 알고리즘을 예로써 기술한 것이다. 고착고장이 발생한 $N=16$, $m=4$이며 8개의 2×2크로스바 스위치로 구성된 16×16 베이스라인 망에서 고장진단 과정을 예로써 제시한다. 그림 6에서 시작지 주소 0110에 목적지 주소 0011을 가진 라우팅 비트 r를 전송 시켰다. 그러나, 이 라우팅 비트가 도달한 목적지 주소는 0101로서 원하는 목적지에 도달하지 못했음을 알았다. 그래서 차후에 데이터정보는 전송되지 못했다. 이상의 정보를 가지고 $S_{1,1}, S_{2,2}$에 각각 SaT와 SaX고장이 존재함을 밝히겠다. 알고리즘에 의한 진단 과정은 다음과 같은 순서로 진행된다. 첫째, 라우팅 비트r 과 목적지 주소 d를 비교하여 올바르지 못한 목적지 주소를 가지게 되면 고장 단을 검출해낸다. 둘째, 모든 고장 스위치의 정확한 위치 정보를 찾는다. 셋째, 어떤 유형의 고착고장인지를 진단한다.

이 과정을 각각 상세히 기술하면 다음과 같다.

첫째, 고장이 발생한 단의 열 정보 F_m^1를 얻을 수 있다.

$$F_m = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \tag{10}$$

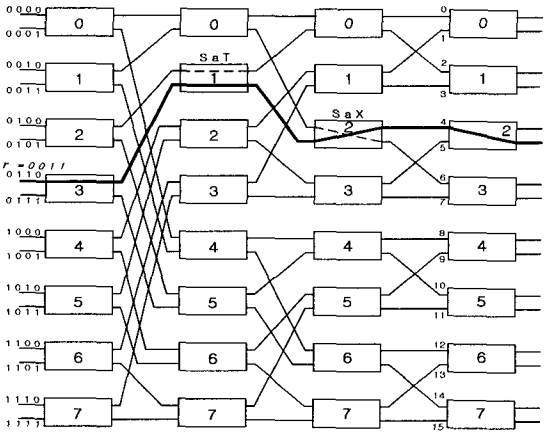

그림 6. 16×16 베이스라인 망
Fig. 6. 16×16 baseline network.

다음으로는 고장 스위치 행 위치 정보 j를 검출하고 고장 유형을 진단한다. 이때, 시작지 주소가 0110이므로 정의 4에 의해 3행 0열에 위치한 스위치 $S_{3,0}$에서 라우팅이 시작된다. 그리고 시작지의 라우팅비트 $r_0=0$이므로 정의 1에 의해서 0단에 위치한 스위치의 출력포트 주소는 $S_{3,0}(0)=A_o^0=even$이다. 이때, A_o^k의 십진수 표현을 α라 하면 정의 4로부터 다음 (11)을 얻을 수 있다.:

$$\alpha(A_i^k)=\left\{\begin{array}{l} A_o^k=even \text{ 이면, } \alpha=2j \\ A_i^k=odd \text{ 이면, } \alpha=2j+1 \end{array}\right\}. \qquad (11)$$

(11)에서 $j=3$이고 $A_o^0=even$이므로 $\alpha=2j=2\times3=6$이다. 6의 이진수 값은 0110이므로 A_o^0 = 0110이다. 정의 3의 링크함수 π_0에 의해 0단에서 1단으로 이동되면 A_i^1의 값을 얻을 수 있다. 첫번째 단, $1\in F_m^1$ 이므로 고장이 존재하는 단임을 알 수가 있다. 그러므로 정의 3에 의해 $A_o^0 \xrightarrow{\pi_0} A_i^1$(0단에서 1단으로)는 $b_3b_2b_1b_0 \xrightarrow{\pi_0} b_0b_3b_2b_1$이므로 $0110 \xrightarrow{\pi_0} 0011$이다. 이때, 정의 4에 의해서 0011의 십진수 값 n=3, $A_i^1=odd$ 이므로

$$j=\frac{3-1}{2}=1 \qquad (12)$$

이다.

(10), (12)에서 나온 두 값으로 처음고장이 발생한 스위치는 1행 1열에 위치한 $S_{1,1}$임을 알았다. 그리고 고장이 존재하는 스위치 $S_{1,1}$에 라우팅 비트 $r_1=0$을 입력하면 정의 5에서의 어느 한 가지 형태를 가지게 된다. 여기서 $A_i^1 \xrightarrow{S_{1,1}} A_o^1$이면 $0011 \xrightarrow{S_{1,1}} 0011$이므로 $b_{m-1}\cdots b_0 \xrightarrow{S_{1,1}} b_{m-1}\cdots b_0$형태가 되어서 T형 고착고장에 해당이 된다. 즉, 스위치 $S_{1,1}$에 발생한 고장의 형태는 T형 고착 고장임을 알 수가 있다. 앞에서 얻은 A_o^1의 값으로 스위치 $S_{2,2}$에서 발생한 고장도 동일한 과정을 반복하여 진단 가능하다. 이와같이 제안한 알고리즘을 통해 반복시행착오(trial-and-error)를 거치지 않고 체계적으로 고착고장 발생스위치의 위치 및 고장 형태를 진단 할 수 있다.

V. 결 론

시스템에서 고장의 조기 진단은 시스템의 성능 저하를 줄여서 신뢰성을 높여주므로 매우 중요하다. 본 논문은 이러한 의미에서 MIN에서의 고장진단 기법에 대해서 고찰하였으며 특히 MIN에서 발생할 수 있는 고착고장에 대해 중점 연구하였다. 제시한 알고리즘은 $O(\log_{10} N)$의 수행시간을 보이며 이는 기존 방식가운데 정량적으로 알고리즘의 수행시간을 분석한 Feng 알고리즘[4]의 $O(\log_2 \log_2 N)$보다 적은 수행시간을 통해 고장진단이 가능함을 나타낸다. 또한 네트워크의 형태가 바뀌더라도 링크함수만 변화시켜 적용 할 수 있어서 기존의 알고리즘보다 효율성을 높였다. 향후 연구과제로는 상호연결망에서 고장진단 문제를 해결하고 라우팅 시에 발생될 수 있는 충돌(conflict)에서의 고장에 대한 해결 방안 모색 등이 있다.

참 고 문 헌

[1] S-H. Byun and D-K. Sung, "The UniMIN switch architecture for large-scale ATM switches", *IEEE/ACM Trans. Networking*, Vol. 8, No. 1, pp. 109-120, 2000.

[2] Y. Yang and J. Wang, "Optimal all-to-all personalized exchange in self-routable multistage network", *IEEE Trans. Parallel and Distributed Systems*, Vol. 11, No. 3, pp. 261-274, 2000.

[3] U. Maulik, S. Bandyopadhyay, and S. Bhatta-

charyya, "Fault tolerant permutation mapping in multistage interconnection network", *Journal of System Architecture*, Vol. 46, No 3, pp. 297-300, 2000.

[4] T-Y. Feng and Y-G. Kim, "Fault-diagnosis for a class of distributed control multistage interconnection network", *IEEE Proc. Distributed Computing Systems*, pp. 1085-1095, 1995.

[5] S. Das and A. Chaudhuri, "An algorithm for identification of multiple faults in a non-redundant multistage interconnection network", *Proc. of IEE on Conf. Intelligent Systems Engineering*, No. 395, pp. 434-441, 1994.

[6] S. Das and A. Chaudhuri, "Fault diagnosis in a benes interconnection network", *IEEE Trans. Parallel and Distributed Systems*, Vol. 9, No. 7, 1998.

[7] J. P. Hayes, *Computer Architecture and Organization*, McGraw-Hill, 1988.

저 자 소 개

金 永 宰(學生會員)

2001년 울산대학교 전기공학과 졸업. 2001년~현재 울산대학교 전기전자정보시스템공학부 석사과정. 관심분야는 이산사건시스템의 관리제어, 광통신망 시스템(스위칭 및 교환기)

曺 光 鉉(正會員)

1993년 한국과학기술원 전기및전자공학과 졸업. 동대학원 석사(1995), 동대학원 박사(1998). 1998.~1999. 동대학원 위촉연구원, 1999. 3.~현재 울산대학교 전기전자정보시스템공학부 조교수. 관심분야는 이산사건시스템의 해석 및 관리제어, 통신망 분석 및 제어, 광통신망 시스템(스위칭 및 라우팅), 시스템이론의 바이오정보공학 응용 등