論文 90-27-9-6

이중구조를 갖는 제어시스템의 구현과 신뢰도 분석에 관한 연구

(A Study on the Implementation of a Control System with Dual Structure and Its Reliability Analysis)

朴 世 華* 文 鳳 彩* 金 炳 國* 卞 增 男*

(Se Hwa Park, Bong Chae Moon, Byung Kook Kim, and Zeung Nam Bien)

要 約

이 논문에서, 높은 신뢰성을 갖는 내고장성 제어시스템을 구현하기 위해 CPU모듈과 I/O 모듈을 각각 이중화하였다. 이를 위해, 시스템 내에서의 고장을 조사하고, 고장탐지 기법으로써 모듈 자체 테스트, 비교 과정, 그리고 예외처리를 적용하였다. 모듈 자체 테스트는 어느 소자가 고장인가를 찾아내는데 사용되며, 비교과정은 두 CPU 모듈에 의해 계산된 제어 출력값을 비교하는 것으로, 잘못된 제어 출력에 의한 플랜트의 이상 동작을 미리 막을 수 있다. 마지막으로 예외처리는 I/O 모듈이 데이타 전달 인식 신호를 보내지 않았을 경우에 생기게 되는 버스 에러 같은 모듈 자체 진단이나 비교과정에 의해 즉시 발견될 수 없는 고장을 결정하는데 사용된다. 또한 이중구조의 이산 시간 Markov 모델로 신뢰도 분석을 하였다. 그리고 단일구조를 갖는 제어시스템보다 이중구조를 갖는 제어시스템이 신뢰도가 높음을 정량적으로 보였다.

Abstract

In this paper, a reliable control system structured with dual CPU modules and dual I/O modules is implemented as a means of achieving a highly reliable fault tolerant control system. For this, faults in the system modules are first examined, and a fault detection technique consisting of self diagnostic, comparison process, and exception processing is applied. Self diagnostic is used to locate which components in the modules have been failed, while comparison process is to compare control outputs computed by both CPU modules and protect the plant from malfunction by blocking failed control outputs in advance. Finally exception processing is used to determine the faults that are not detected immediately by the self diagnostic and comparison process, e.g. bus error processing when acknowledge signal for data transfer is not activated in the I/O modules. Also reliability analysis is conducted for the discrete time Markov model with dual structure. It is shown quantitatively that the reliability is improved in the control system with dual structure in comparison with a system with single module structure.

*正會員, 韓國科學技術院 電氣및 電子工學科
(Dept. of Electrical Eng., KAIST)

接受日字 : 1990年 2月 16日

I. 서 론

제어시스템은 대상 플랜트를 적절히 제어하여 외란에도 불구하고, 원하는 출력을 얻도록 하는 데에

그 목적이 있다. 현대 산업사회가 복잡화되고 규모가 커짐에 따라, 대규모 제어시스템의 신뢰도가 요구되고 있는 바, 지속적이고 안정되게 플랜트를 제어하기 위해 제어시스템 자체의 신뢰성을 높이려는 노력이 있어 왔다.[1-4]

발전소 역시 제어대상이 많은 대규모 플랜트의 일종으로써, 만일 발전소 제어시스템의 고장사고로 발전소가 가동 중지하게 되면 경제적, 사회적인 큰 손실을 가져오게 되므로 제어시스템의 신뢰도 문제가 중대한 문제로 대두되고 있다. 제어시스템의 낮은 신뢰도로 인해, 제어시스템의 고장사고가 생기기도 하는데, 예를들어 울산화학발전소의 경우 전체 고장사고의 35.1%가 제어시스템의 고장사고이며,[5] 최근의 국내 원자력 발전소의 가동 중지 46건중 7건이 제어시스템과 관련된 오동작에 의한 것으로 보고되고 있다.[6]

최근의 제어기는 반도체 기술과 마이크로 프로세서의 발달에 힘입어 디지탈 제어기가 보편화되어, 그림 1과 같이 연산기(CPU) 모듈(module)과 A/D 변환(anlog to digital conversion)기능과, D/A변환(digital to analog conversion)기능 등을 갖는 입출력 모듈로 구성된다.[7] 이 구조에서 모듈내의 단일소자만의 고장으로도 그 모듈이 오동작 할 수 있으며, 나아가서 전체시스템이 제기능을 다 못하는 경우가 생기기도 한다.

이런 고장에 의한 제어기의 오동작을 막는 방법으로, 고품질의 소자를 사용하여, 소자의 고장 자체를 되도록 줄게 하는 고장회피(fault avoidance) 방식과 소자에서 고장이 발생하더라도 시스템의 성능에는 영향을 주지 않도록 하는 내고장성제어(fault tolerant control)방식이 있다.[8][9] 그런데, 고장회피 방식만으로는 운용 환경의 변화, 경년 변화 등으로 인한 고장에 대처할 수 없기 때문에 좀 복잡하지만 적극적인 방법인 내고장성 제어방식을 취할 필요가 있다.[10] 내고장성 제어방식은 같은 기능을 하는 모듈을 여러 개 두는 중복구조(redundancy)를 갖도록 하는 것인데, 우주선 항공기의 제어나 원자력 발전소 같은 대규모 공정 제어 분야[9]등 높은 신뢰성을 요구하는 분야에 주로 이용되어 왔다.

Bailey 회사는 CPU 모듈을 이중화한 형태의 발전소 제어시스템을 선보이고 있으며,[11][12] Taylor 회사도 발전소용 제어시스템으로 한 CPU 모듈이 여러 개의 CPU 모듈 중 고장난 어느 모듈과도 스윗칭을 통한 백업 조치가 가능하도록 구성된 것을 볼 수 있는데,[13] 이들 모두 중복구조를 이용해 제어시스템의 신뢰도를 향상시키려고 하고 있다. 국내의 연구로는 아날로그 모듈로 이루어진 고장이 잦은 한 발전소 제어기의 신뢰성을 높이기 위해 디지탈 백업제어기를 구성한 경우도 있는데,[3] 이는 기존의 제어시스템에 추가로 덧붙여졌기 때문에, 하이브리드 형태의 중복구조를 취하고 있다. 높은 신뢰성을 요구하는 제어시스템은 중복구조를 가질 필요가 있으나, 지나치게 많은 중복구조를 갖도록 제어시스템을 다중화하면, 너무 많은 추가비용이 들어갈 수 있으며, 이중구조만으로도 충분히 경제적으로 시스템의 신뢰도를 높일 수 있다고 본다. 그러나, CPU 모듈만 이중화했을 경우에 I/O 모듈이 고장나면(예로 A/D 변환기 고장), 이중화된 CPU 모듈은 큰 의미가 없게 된다.

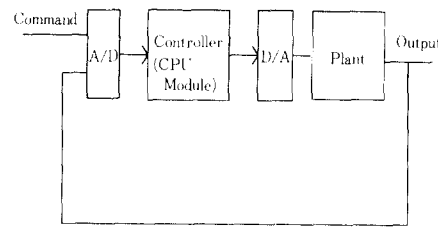

그림 1. 디지탈 제어기의 구성도
Fig. 1. The configuration of digital controller.

본 논문에서는, 동일한 형태와 기능을 갖는 CPU 모듈과 I/O 모듈 모두가 이중화된 제어시스템을 구현하였다. 그래서, 제어시스템 내의 어느 고장에 대해서도 보다 적극적으로 대처하려 한다. 그리고, 구현된 이중구조를 갖는 제어시스템과 단일구조를 갖는 경우와의 신뢰도를 정량적으로 비교 분석하는 작업도 수행하였다.

Ⅱ. 이중구조를 갖는 제어시스템

플랜트를 제어하기 위한 디지탈 제어기를 하드웨어적으로 구성함에 있어서 CPU 모듈, I/O 모듈, 신호 조절(signal conditioning)기능이 있는 모듈등이 필요하다. I/O 모듈은 신호 변환(D/A 변환, A/D 변환 등)등이 포함된 모듈을 말하며, 신호 조절 기능은 플랜트로 보내는 제어 신호를 전압 혹은 전류로 바꾸거나, 적절한 level로 신호를 바꾸는 기능을 말한다. 그림 2는 이중구조를 갖도록 구성한 시스템을 나타내고 있다. 점선 부분은 이중화시 추가로 포함된 부분이다. MLC(multi loop controller)는 MC68000 CPU를 탑재하고,[24] VME 버스규격[21]에 맞게 설계된 CPU 모듈이며, SIB(signal interface board)는 I/O 모듈로써 데이타 입출력을 위한 A/D 변환 또는

D/A 변환을 맡고 있으며, MLC와 SIB가 이중화되었다. STB(signal termination board)는 SIB와 플랜트 사이에서 신호 조절 기능을 담당함은 물론 이중구조를 갖는 SIB 출력 신호를 선택하기 위한 스위칭 기능을 포함하고 있다.

MLC-A는 주제어기 모듈이고, MLC-B는 MLC-A가 고장시 백업 제어를 하게 되는 부제어기 모듈을 의미한다. 마찬가지로 SIB-A가 우선하는 모듈이고, SIB-B는 SIB-A의 고장시에 그 기능을 대신하는 모듈이다. 그리고, S-RAM은 두 MLC 사이에서 글로발(global) 버스를 통한 통신과 데이타 저장 등을 위한 것이다.

본 제어시스템은 분산형 제어시스템[14]을 이루기 때문에 글로발(global) 버스를 통해 통신을 담당하는 다른 CPU 모듈의 중재를 거쳐 관리 제어시스템의 통제를 받기도 하는데, 이 부분은 본 논문의 고려대상에서 제외하기로 한다.

MLC는 로컬(local) 버스를 통해 SIB와 데이타를 주고 받는데, 로컬 버스 아비터를 구현함으로써, MLC-A와 SIB-B 또는 MLC-B와 SIB-A 등 4가지 조합의 어느 구성도 가능하여 크로스 백업(cross back-up)을 할 수 있다. MLC는 모듈 내부 혹은 외부의 어떤 디바이스(예로 메모리)로 부터든지 데이타를 입출력 받을 때 스트로브를 발생시키고 이에 대한 응답으로 데이타 전송 인식 신호(DTACK, data transfer acknowledge)[21]를 받는데, 로컬 버스를 통한 데이타 교환시에는 입출력 버퍼를 통하여 데이타를 전송한다. 그러므로, SIB로부터 데이타를 입출력하려 할때에도 스트로브를 보내고 DTACK을 받는데, 로컬 버스 아비터는 두 MLC가 보내는 두 스트로브 신호를 이용하여 state machine[16]으로 구현하여 순차적으로 각 MLC의 입출력 버퍼의 여닫음을 제어하는 신호를 만들어 주는 역할을 한다.

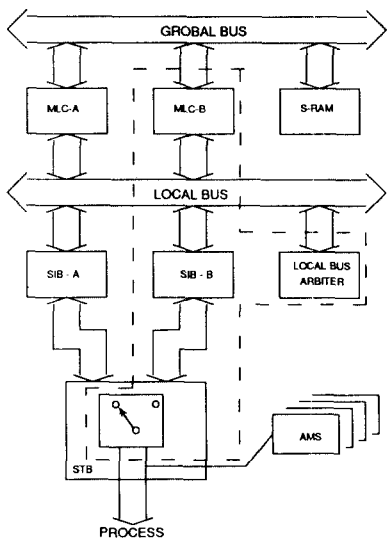

그림 2. 전체적인 하드웨어 구성도
Fig. 2. The hardware structure.

STB에 SIB의 스위칭을 위한 펄스 탐지 회로가 첨가되어 있는데, 0혹은 1의 레벨(level)로써 SIB의 스위칭 신호를 보내게 되면, 고장으로 항상 0(stuck at zero fault)이거나 1일 때 원하지 않는 스위칭이 일어날 수 있다. 그래서 스위칭 신호를 펄스 방식으로 함으로써 고장시에 잘못된 신호레벨에 의한 스위칭 가능성을 없앨 수 있다. 실제로 SIB-A의 고장시에 SIB-B를 통해 STB로 펄스를 보냄으로써 스위칭이 이루어진다.

AMS(auto/manual station)는 수동 운전과 자동운전의 절환을 용이하게 하기 위함으로 마련되었다. 만일 두 MLC나 두 SIB 모두 고장이 나서 더 이상 플랜트의 제어가 불가능할 때는 제어기의 출력을 차단하고 수동 운전 상태로 절환을 시켜 사람이 직접 제어 출력값을 조작할 수 있도록 하는 것이 잘못된 제어 출력으로 인해 플랜트에 미치는 악영향을 막을 수 있어서 시스템의 안전성을 높이게 된다. 그래서 두, MLC나 두 SIB의 고장을 탐지하여, 자동으로 수동절환을 시킬 수 있도록 하는 회로가 AMS에 첨가되어 있다. 즉, 정상시에 MLC는 매 샘플링 시간마다 펄스를 보내면, AMS에서주기적인 펄스를 탐지하게 되는데, 만일 이 펄스가 오지 않으면 AMS는 스스로 수동절환을 하게 된다.

Ⅲ. 제어시스템 내의 고장과 이중구조

1. 고장의 분류

제어시스템의 이중구조를 위해 부가된 local 버스 아비터와 SIB의 백업을 위한 STB내의 스위칭 부분은 간단하게 설계하여 신뢰성을 높임으로써, 타 모듈과 비교할 때 고장이 거의 일어나지 않음을 가정한다.

MLC의 경우 고장 부위 결정 단위는 기능별 구성요소로 설정하였으며, MC68000 CPU, Timer, 버스, 메모리 등으로 구분되고, SIB의 경우는 A/D 변환 회로, D/A 변환 회로, on-off의 디지탈 신호의 입출력 회로, MLC와 인터페이스(interface)하는 회로 등으로 구분된다.

2. 고장의 탐지 및 고장 부위 결정 방법

MLC 내의 고장의 탐지는 크게 모듈 자체 테스트에 의한 방법과 다른 MLC와의 계산된 데이타의 비교 과정, 그리고, 보완적인 방법으로 자체 진단과 비교 과정만으로 모든 MLC 내의 고장을 찾지 못하는 경우를 위해, MC68000 CPU가 처리해 주는 예외처리(exception processing)[24]에러에 의해 고장을 탐지하게 된다.

모듈 자체 테스트는 각 소자별로 그 기능을 제대로 수행하는 가를 확인하는 방법으로 MC68000 CPU, Timer, ROM, RAM, 데이타 버스, 어드레스 버스 등을 MLC가 플랜트를 제어하고 있지 않는 동안에 계속 반복 테스트하는 것이다. 예를 들어 RAM의 경우 쓴 데이타와 읽은 데이타의 비교시에 다를 경우 RAM이 고장이라고 보는 것인데, [4]에서 사용한 방법을 보완 적용하였다. 아날로그 제어기의 디지탈 백업제어기[3]에서의 고장탐지 방법은 주로 모듈 자체 테스트에 의해서만 했는데, 여기서는, MLC의 고장탐지를 모듈 자체 테스트뿐만 아니라 비교과정, 예외처리 과정을 둠으로써 보다 확장된 방법으로 고장에 대응한다.

비교 과정에 의한 고장 탐지는 계산된 출력값이 서로 차이가 날 때 한쪽을 고장이라고 판단하는 것이다. 일반적으로 그 비교값이 다를 경우 어느 모듈이 고장인지 그 결과만 보고 결정할 수 없기 때문에 다음과 같은 규칙에 의해 결정하였다.

$$\| y(hk)_{MLC\ A} | - | y(hk)_{MLC-B} \| > \epsilon, \quad k=0, 1, 2\cdots$$

일때

$$\max\left(\left| \frac{y(hk)_{MLC-A} - y(hk-h)}{y(hk-h) - y(hk-2h)} \right|, \left| \frac{y(hk)_{MLC-B} - y(hk-h)}{y(hk-h) - y(hk-2h)} \right| \right)$$

을 계산한 모듈을 고장난 MLC 모듈로 본다. 여기서, h는 샘플링 시간으로, hk는 총 동작 시간을 의미하며, y는 계산된 출력값이다. 즉 $y(hk)_{MLC-A}$는 hk 시간에 MLC-A에서 계산된 출력값이다. 또한 ϵ는 threshold를 의미하며 시스템에 맞게 정해준다. 위의 방법은 일반적으로 제어기의 경우 고장이 생기면, 그 출력신호 값이 급격히 변한다는 사실을 고려한 것이다.

예외처리[24]에 의한 고장 탐지는 MC68000 CPU가 처리해주는 고유한 기능을 이용한 것으로, 하드웨어나 소프트웨어에 이상이 생겼을 때, 정해진 예외벡터(exception vector)[24]에서 수행할 루틴의 시작 어드레스를 가져와 그 루틴을 수행하는 것이다. 예를 들어 잘못된 인스트럭션(instruction)이 있으면 예외처리 'illegal instruction' 에러가 발생한다. 또 다른 예로 하드웨어에 고장이 생겨 DTACK 신호를 발생시키지 못하면 예외처리 버스에러가 발생한다. 이와 같이 예외처리는 모듈 자체 테스트나 비교 과정에 의한 고장 탐지의 보완적인 방법으로, 예외처리 에러시 즉시 예외처리 루틴을 수행하게 되며, 예외처리 에러 루틴에 고장정보를 담은 플래그를 세워 고장의 상황을 나타낸다.

SIB에서 발생되는 고장은 MLC의 진단에 의해 탐지되며, 크게 다음의 3가지 방법을 적용하였다.

SIB에서 A/D 변환 시에 인터럽트(level 4) 방식을 사용하였기 때문에 변환 시작후 정해진 시간내에 인터럽트 신호가 오는가를 확인하는 방법을 적용하였다. 즉, 정해진 시간내에 인터럽트 신호가 오지 않을 경우, A/D 변환 회로의 고장임을 알 수 있다.

또, 범프 없는 전환(bumpless transfer)[7]를 위해 MLC가 SIB를 통해 STB로 내보내는 신호를 다시 SIB를 거쳐 MLC가 되받게 되어 있는데, 이 때, D/A 변환과 A/D 변환 과정을 거친다. 여기서 내보낸 값과 되받은 값이 다를 시에 이 부분 전체의 고장이라고 본다. 비슷한 방법이 on-off 신호에서도 적용된다.

MLC와의 인터페이스 부분에서 MLC가 SIB에 데이타를 읽거나 쓸 때, 데이타 전달 인식 신호를 MLC로 보내 주어야 한다. 만일, 이 신호가 오지 않으면 MLC에서 버스 에러가 발생하므로 SIB의 고장을 알 수 있다. 그러나 버스에러는 MLC 내부의 원인에 의해 생길 수도 있으므로, 이의 구별을 필요로 한다. 그 구별은 SIB 모듈을 읽거나 쓰기 전에 플래그를 세웠다가 후시 다시 플래그를 지우는 방법을 사용하여 만일 버스 에러가 생겼을 때 이 플래그가 세워져 있으면 SIB모듈에 의한 것이고 그렇지 않으면 MLC내부의 원인에 의한 것으로 구별을 할 수 있다. MLC 모듈 내부의 원인에 의한 예외처리 에러인 경우에 MLC 백업 등의 조처가 뒤따르고, MLC 모듈 외부의 원인 즉 SIB에 의한 경우이면 SIB 백업을 실행한다. MLC에서의 고장 부위의 결정은 세부적인 모듈 자체 테스트를 통하여 보완한다.

3. 이중구조 알고리듬

이중구조 알고리듬을 실행함에 있어서 두 MLC사이의 동작 상태를 나타내는 상태 플래그를 다음과 같이 설정하였다.

i) 각 MLC가 동작하고 있음을 나타내 주는 것
ii) SIB로부터 데이타를 입출력시에 고장이 있었는지를 나타내는 것
iii) SIB 고장시에 진단을 했음을 나타내 주는 것
iv) MLC-A가 제어 출력 값을 계산했음을 나타내

주는 것

v) MLC-B가 비교 과정을 끝냈음을 나타내 주는 것

vi) 비교 과정의 결과를 나타내 주는 것

vii) MLC-A나 MLC-B의 동작을 강제로 멈추게 하는 것

주기적으로 반복되는 샘플링 시간 동안의 수행되는 일의 타이밍 다이아그램(timing diagram)은 그림 3과 같다.

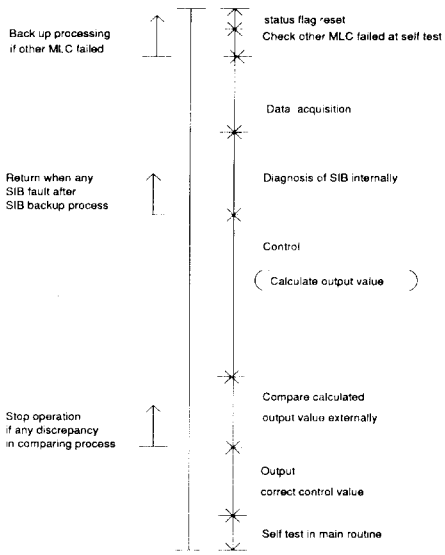

그림 3. 한 샘플링 시간 내의 일 수행도
Fig. 3. The timing diagram of jobs in one sampling time.

첫번째 단계로 상태 플래그 ii), iii), iv), v), vi)를 초기화한다. 두번째 단계로 다른 MLC에서 모듈 자체 테스트 중에 고장이 있었는가를 플래그를 통해 점검한다. MLC-A의 고장시에는 MLC-B가 백업 제어를 하고, MLC-B의 고장시에는 플래그 vii)을 세워서 MLC-B의 동작을 멈추게 된다.

세번째 단계로 SIB를 통해 데이타를 입력받고, SIB 내에 고장이 탐지되면 SIB백업을 한 후 ii), iii) 플래그를 세운다. SIB의 백업은 SIB-B를 통해 STB로 펄스를 보내어 SIB를 스윗칭하는 것이다. 네번째 단계에서 각 MLC는 제어 출력 값을 계산하고, MLC-A가 iv)플래그를 세우면, MLC-B가 비교한 후 v), vi)플래그를 세운다. 이 때 비교값에 차이가 있으면, 변화 폭이 큰 MLC를 고장으로 판단한다. 이 비교 과정에서 MLC-A가 고장일 경우 MLC-B가 백업 제어를 수행하며, 이 때 vii)플래그를 세운다. 마지막 단계는 자체 테스트를 수행하는 과정으로 고장이 탐지되면 고장정보를 담은 플래그를 세운다.

위의 작업 수행을 위한 소프트웨어는 크게 주요 루틴(main routine)과 매 샘플링 타임마다 MLC 내부의 타이머에 의해 MC68000 CPU에 보내지는 인터럽트(level 2)에 의한 타이머 인터럽트 처리 루틴, 그리고 A/D 변환이 끝난 후에 SIB가 발생시키는 인터럽트(level 4)에 의한 ADC 인터럽트 처리 루틴으로 이루어진다. 주요 루틴에서는 주로 MLC의 자체 테스트가 이루어지며, 타이머 인터럽트 루틴에서는 플랜트 제어와 관계되는 모든 작업이 수행되는데, SIB를 통한 데이터 입력, SIB 진단, MLC가 정상임을 나타내는 펄스 발생, 제어 출력값 계산, 제어 출력값의 비교, 그리고 그 값들의 출력 등이 수행된다. 그리고 ADC 인터럽트 처리 루틴에서는 A/D 변환이 끝났다는 플래그를 세워 변화된 데이타를 MLC가 읽어갈 수 있도록 해준다.

전체적인 소프트웨어 구조는 그림4, 그림5와 같다. 그림4는 주요 루틴 알고리듬이고, 그림5는 타이머 인터럽트 처리 루틴 알고리듬이다. 주요 루틴에서는 관리제어 시스템으로부터 시작 신호를 받고 있으며, MLC-A가 단독으로만 동작할 때도 고려하였다. 이 때는 계산값과 비교 과정 뿐만아니라 다른 MLC의

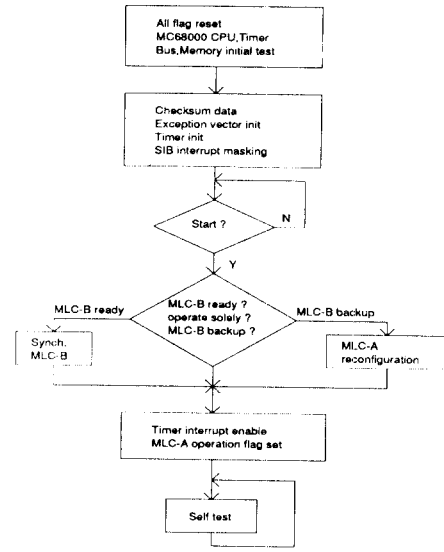

그림 4. 메인 루틴 알고리듬
Fig. 4. The algorithm of main routine.

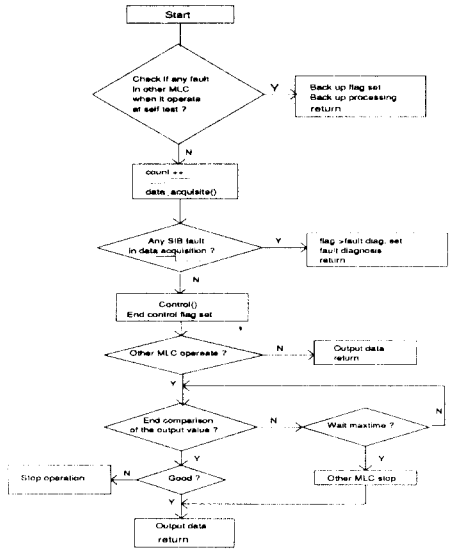

그림 5. 타이머 인터럽트 처리 루틴 알고리듬
Fig. 5. The algorithm of timer interrupt sevice routine.

상태를 점검하는 루틴 등이 생략된다. 그리고 오프라인(off line)으로 초기화하는 과정이 포함되어 있다. 타이머 인터럽트 처리 루틴에서는 전술한 바와 같은 작업이 수행된다.

두 MLC는 동기(synchronization)를 이루어 비슷한 작업을 수행하게 되는데, 정상적으로 동작 할 경우, 비교 과정은 MLC-B가, 데이타 출력과 SIB의 고장 진단은 MLC-A가 담당한다.

Ⅳ. 신뢰도 분석

1. 신뢰도 및 신뢰도 분석 방법

발전소 제어시스템은 높은 신뢰도를 가져야 하는데, 전기전자 공학자 협회의 전기전자 용어 사전에 의하면 자동제어에서의 신뢰도는 한 장치가 특정한 운용 조건하에서 주어진 기간동안에 목적한 바의 기능을 충분히 수행할 수학적 확률로 정의된다. 이에 대한 척도로써 평균수명(MTTF), 평균고장간격(MTBF), 신뢰도 함수(reliability fuction)등이 있는데, 이러한 규정은 약간씩 차이가 있으나 신뢰도를 높인다는 것은 운용 조건이 변하여도 제어시스템이 원하는 동작 성능을 충분기간 유지하는 것이라고 볼 수 있겠다.

본 논문에서는 정량적으로 신뢰도를 분석하기 위해서, 신뢰도 함수에 의한 방법을 사용하려고 하며, Markov 모델에 의하여 해석하는 것이 여러 요소를 고려하는데, 융통성을 주기 때문에,[16] 본 제어시스템을 Markov 모델로 표현하여 신뢰도 함수를 얻어내려고 한다. 신뢰도의 분석방법은 Siljak이 중복구조를 갖는 다중 제어기 구조에 대해 해석을 했는데,[17][18] Siljak의 접근 방법은 상당히 단순화되어 실제와는 거리가 멀기 때문에, Johnson의 접근 방법[16]과 유사하게 본 제어시스템에 적용해 보려 한다. MLC 자체는 hot stand-by 구조이기 때문에 크게 네 개의 상태로, SIB는 cold stand-by 구조이기 때문에 다섯개의 상태로 나누고, 몇가지 확률 변수를 정의하여, Markov 모델 식(equation)을 세웠으며, 그로부터 신뢰도 함수를 구하려고 한다.

2. Markov 모델

우선, 본 제어시스템에서 STB와 AMS 부분은 고장의 모델에서 제외를 하고, 또 버스 아비터도 고장이 없다고 가정한다. 그래서 MLC-A와 MLC-B, SIB-A,, 그리고, SIB-B에 국한시킨 이중구조의 제어시스템을 Markov 모델로 전개하려고 한다. 그러기 위해서, 표 1과 같이 여러 확률 변수 등을 정의했으며, MLC, SIB 모두 이중화했을 때의 가능한 상태를 나누면 표 2와 같이 20가지의 상태로 나눌 수 있다.

표 1. 확률 변수 정의
Table 1. The definitions of probability variable.

	변 수
$\mathbf{R}(t)$	시스템이 시간 t_0에서 동작하고 있을 때, $[t_0, t]$사이의 시간동안 계속 동작할 확률
$\mathbf{P}_{dual}$	MLC와 SIB 모두 이중구조를 갖는 본 제어시스템의 확률 상태 벡터(probability state vector)
$\mathbf{P}_{single}$	MLC와 SIB 모두 단일구조를 갖는 제어시스템의 확률 상태 벡터(probability state vector)
$\mathbf{T}_{dual}$	MLC와 SIB 모두 이중구조를 갖는 본 제어시스템의 상태 전이 매트릭스(state transition matrix)
$\mathbf{T}_{single}$	MLC와 SIB 모두 단일구조를 갖는 제어시스템의 상태 전이 매트릭스(state transition matrix)
C_c	MLC에서의 고장이 비교 과정에 의해서 탐지되어 성공적으로 구조재편성(reconfiguration)이 될 확률
C_d	MLC에서의 고장이 모듈 자체 테스트(self test)에 의해 탐지되고, 성공적으로 구조재편성(reconfiguration)이 될 확률
C	SIB에서의 고장이 MLC의 진단에 의해 발견되고, 성공적으로 구조재편성(reconfiguration)이 될 확률
λ_1	MLC의 고장률
λ_2	SIB의 고장률

표 2. 이중구조 시스템의 상태 정의
Table 2. The status definition of system with dual structure.

상 태	내 용
$P_{2_{1,1}}$	두 MLC와 두 SIB 모두 정상인 상태
$P_{2_{1,0}}$	두 MLC 모두 정상이고, SIB-A는 정상이고, SIB-B가 고장인 상태
$P_{2_{0,1}}$	두 MLC 모두 정상이고, SIB-A에 고장이 탐지되어 SIB-B로 성공적으로 스윗칭된 상태
$P_{2_{F,U}}$	두 MLC 모두 정상이고, SIB-A에 고장이 생겼지만, 탐지되지 않으며, SIB-B는 정상인 상태
$P_{2_{A,F}}$	두 MLC 모두 정상이고, 두 SIB 모두 고장인 상태
$P_{0.1_{1,1}}$	한 MLC는 정상이고, 다른 MLC의 고장이 비교 과정 혹은 자체 진단에 의해 탐지되어 조치를 취하고, 두 SIB 모두 정상
$P_{0.1_{1,0}}$	한 MLC는 정상이고, 다른 MLC의 고장이 비교 과정 혹은 자체 진단에 의해 탐지되어 조치를 취하고, SIB-A는 정상이고, SIB-B가 고장인 상태
$P_{0.1_{0,1}}$	한 MLC는 정상이고, 다른 MLC의 고장이 비교 과정 혹은 자체 진단에 의해 탐지되어 조치를 취하고, SIB-A에 고장이 탐지되어, SIB-B로 성공적으로 스윗칭된 상태
$P_{0.1_{F,U}}$	한 MLC는 정상이고, 다른 MLC의 고장이 비교 과정 혹은 자체 진단에 의해 탐지되어 조치를 취하고, SIB-A에 고장이 생겼지만, 탐지되지 않으며, SIB-B는 정상인 상태
$P_{0.1_{A,F}}$	한 MLC는 정상이고, 다른 MLC의 고장이 비교 과정 혹은 자체 진단에 의해 탐지되어 조치를 취하고, 두 SIB 모두 고장인 상태
$P_{1.0_{1,1}}$	MLC가 고장이지만, 탐지안 된 상태이고, 두 SIB 모두 정상인 상태
$P_{1.0_{1,0}}$	MLC가 고장이지만, 탐지안 된 상태이고, 두 SIB SIB-A는 정상이고, SIB-B가 고장인 상태
$P_{1.0_{0,1}}$	MLC가 고장이지만, 탐지안 된 상태이고, SIB-A에 고장이 탐지되어, SIB-B로 성공적으로 스윗칭된 상태
$P_{1.0_{F,U}}$	MLC가 고장이지만, 탐지안 된 상태이고, SIB-A에 고장이 생겼지만, 탐지되지 않으며, SIB-B는 정상인 상태

실제로 $\mathbf{P}_{dual}, \mathbf{P}_{single}, \mathbf{T}_{dual}, \mathbf{T}_{single}$ 등은 모두 시간에 대한 함수이다. 확률 상태 벡터 P_{dual}은

$$\mathbf{P}_{dual}(t+\triangle t)=\mathbf{T}_{dual}(t)\ \mathbf{P}_{dual}(t)$$

여고, 여기서,

$$\mathbf{P}_{dual}(t)=\begin{bmatrix}\mathbf{P}_2(t)\\ \hline \mathbf{P}_{0.1}(t)\\ \hline \mathbf{P}_{1.0}(t)\\ \hline \mathbf{P}_{A.F}(t)\end{bmatrix}$$

$$\mathbf{P}_2(t)=\begin{bmatrix}P_{2_{(1,1)}}(t)\\ P_{2_{(1,0)}}(t)\\ P_{2_{(0,1)}}(t)\\ P_{2_{(F,U)}}(t)\\ P_{2_{(A,F)}}(t)\end{bmatrix}\qquad \mathbf{P}_{0.1}(t)=\begin{bmatrix}P_{0.1_{(1,1)}}(t)\\ P_{0.1_{(1,0)}}(t)\\ P_{0.1_{(0,1)}}(t)\\ P_{0.1_{(F,U)}}(t)\\ P_{0.1_{(A,F)}}(t)\end{bmatrix}$$

$$\mathbf{P}_{1.0}(t)=\begin{bmatrix}P_{1.0_{(1,1)}}(t)\\ P_{1.0_{(1,0)}}(t)\\ P_{1.0_{(0,1)}}(t)\\ P_{1.0_{(F,U)}}(t)\\ P_{1.0_{(A,F)}}(t)\end{bmatrix}\qquad \mathbf{P}_{A.F}(t)=\begin{bmatrix}P_{A.F_{(1,1)}}(t)\\ P_{A.F_{(1,0)}}(t)\\ P_{A.F_{(0,1)}}(t)\\ P_{A.F_{(F,U)}}(t)\\ P_{A.F_{(A,F)}}(t)\end{bmatrix}$$

그리고

$$\mathbf{T}_{dual}(t)=\left[\begin{array}{c|c|c|c}\mathbf{T}_{11}(t) & \mathbf{T}_{12}(t) & \mathbf{T}_{13}(t) & \mathbf{T}_{14}(t)\\ \hline \mathbf{T}_{21}(t) & \mathbf{T}_{22}(t) & \mathbf{T}_{23}(t) & \mathbf{T}_{24}(t)\\ \hline \mathbf{T}_{31}(t) & \mathbf{T}_{32}(t) & \mathbf{T}_{33}(t) & \mathbf{T}_{34}(t)\\ \hline \mathbf{T}_{41}(t) & \mathbf{T}_{42}(t) & \mathbf{T}_{43}(t) & \mathbf{T}_{44}(t)\end{array}\right]$$

$$\mathbf{T}_{11}(t)=$$

$$\begin{bmatrix}1-2(\lambda_1+\lambda_2)\triangle t & 0 & 0 & 0 & 0\\ \lambda_2\triangle t & 1-(\lambda_2+2\lambda_1)\triangle t & 0 & 0 & 0\\ \lambda_1\triangle tC & 0 & 1-(\lambda_2+2\lambda_1)\triangle t & \lambda_2\triangle t\ C & 0\\ \lambda_2\triangle t(1-C) & 0 & 0 & 1-\lambda_2(1+C)\triangle t-2\lambda_1\triangle t & 0\\ 0 & \lambda_2\triangle t & \lambda_2\triangle t & \lambda_1\triangle t & 1\end{bmatrix}$$

$$\mathbf{T}_{21}(t)=[\ \mathbf{T}_{21-1}(t)\vdots\mathbf{T}_{21-2}(t)\ ]$$

$$\mathbf{T}_{21-1}(t)=\begin{bmatrix}2\lambda_1\triangle t(C_d+C_c-C_cC_d) & 0\\ 0 & 2\lambda_1\triangle t(C_d+C_c-C_dC_c)\\ 0 & 0\\ 0 & 0\\ 0 & 0\end{bmatrix}$$

$$\mathbf{T}_{21-2}(t)=\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 2\lambda_1\triangle t(C_d+C_c-C_dC_c) & 0 & 0\\ 0 & 2\lambda_1\triangle t(C_d+C_c-C_dC_c) & 0\\ 0 & 0 & 0\end{bmatrix}$$

$$\mathbf{T}_{31}(t)=[\ \mathbf{T}_{31-1}(t)\vdots\mathbf{T}_{31-2}(t)\ ]$$

$$\mathbf{T}_{31-1}(t)=\begin{bmatrix}-2\lambda_1\triangle t(C_d+C_c-C_cC_d-1) & 0\\ 0 & -2\lambda_1\triangle t(C_d+C_c-C_cC_d-1)\\ 0 & 0\\ 0 & 0\\ 0 & 0\end{bmatrix}$$

$\mathbf{T}_{31-2}(t)=$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -2\lambda_1\Delta t(C_d+C_c-C_cC_d-1) & 0 & 0 \\ 0 & -2\lambda_1\Delta t(C_d+C_c-C_cC_d-1) & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$\mathbf{T}_{41}(t)=5\times5$ null matrix

$\mathbf{T}_{12}(t)=5\times5$ null matrix

$\mathbf{T}_{22}(t)=$

$$\begin{bmatrix} 1-(2\lambda_2+\lambda_1)\Delta t & 0 & 0 & 0 & 0 \\ \lambda_2\Delta t & 1-(\lambda_2+\lambda_1)\Delta t & 0 & 0 & 0 \\ \lambda_2\Delta tC & 0 & 1-(\lambda_2+\lambda_1)\Delta t & \lambda_2\Delta tC & 0 \\ \lambda_2\Delta t(1-C) & 0 & 0 & \lambda_2(1+C)\Delta t-\lambda_1\Delta t-\lambda_1\Delta t & 0 \\ 0 & \lambda_2\Delta t & \lambda_2\Delta t & \lambda_2\Delta t & 1 \end{bmatrix}$$

$\mathbf{T}_{32}(t)=5\times5$ null matrix

$$\mathbf{T}_{42}(t)=\mathbf{T}_{43}(t)=\begin{bmatrix} \lambda_1\Delta t & 0 & 0 & 0 & 0 \\ 0 & \lambda_1\Delta t & 0 & 0 & 0 \\ 0 & 0 & \lambda_1\Delta t & 0 & 0 \\ 0 & 0 & 0 & \lambda_1\Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$\mathbf{T}_{13}(t)=5\times5$ null matrix

$\mathbf{T}_{23}(t)=$

$$\begin{bmatrix} 2\lambda_1\Delta t(C_d+C_c-C_cC_d) & 0 & 0 & 0 & 0 \\ 0 & 2\lambda_1\Delta t(C_d+C_c-C_dC_c) & 0 & 0 & 0 \\ 0 & 0 & 2\lambda_1\Delta t(C_d+C_c-C_dC_c) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$\mathbf{T}_{33}(t)=[\mathbf{T}_{33-1}(t)\vdots\mathbf{T}_{33-2}(t)]$

$\mathbf{T}_{33-1}(t)=$

$$\begin{bmatrix} 1-(2\lambda_2+\lambda_1)\Delta t-2\lambda_1\Delta t(C_c+C_d-C_cC_d) & 0 \\ \lambda_1\Delta t & 1-(\lambda_2+\lambda_1)\Delta t-2\lambda_1\Delta t(C_c+C_d-C_cC_d) \\ \lambda_2\Delta tC & 0 \\ \lambda_2\Delta t(1-C) & 0 \\ 0 & \lambda_2\Delta t \end{bmatrix}$$

$$\mathbf{T}_{33-2}(t)=\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1-(\lambda_2+\lambda_1)\Delta t-2\lambda_1\Delta t(C_c+C_d-C_cC_d) & \lambda_2\Delta tC & 0 \\ 0 & 1-\lambda_2(1+C)\Delta t-\lambda_1\Delta t & 0 \\ \lambda_1\Delta t & \lambda_1\Delta t & 1 \end{bmatrix}$$

$\mathbf{T}_{14}(t)=\mathbf{T}_{24}(t)=\mathbf{T}_{34}(t)=5\times5$ null matrix

$$\mathbf{T}_{44}(t)=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

그러면, 여기서

$$\mathbf{R}(t)_{dual}=P_{2_{(1,1)}}(t)+P_{2_{(0,1)}}(t)+P_{2_{(1,0)}}(t)+P_{0,1_{(1,1)}}(t)+P_{0,1_{(0,1)}}(t)+P_{0,1_{(1,0)}}(t)$$

이다.

또, MLC, SIB 모두 단일구조시의 상태는 표 3과 같이 4가지로 나눌 수 있다.

표 3. 단일구조 시스템의 상태 정의
Table 3. The status definitions of system with single structure.

	변 수
$P_{1,1}$	단일구조시에 MLC와 SIB 모두 정상인 상태
$P_{1,0}$	MLC가 정상이고, SIB가 고장인 상태
$P_{0,1}$	MLC가 고장이고, SIB가 정상인 상태
$P_{0,0}$	MLC, SIB 모두 고장인 상태

그러면,

$$\mathbf{P}_{single}(t+\Delta t)=\mathbf{T}_{single}(t)\,\mathbf{P}_{single}(t)$$

$$\mathbf{P}_{single}(t)=\begin{bmatrix} P_{1,1}(t) \\ P_{1,0}(t) \\ P_{0,1}(t) \\ P_{0,0}(t) \end{bmatrix}$$

$$\mathbf{T}_{single}(t)=\begin{bmatrix} 1 & (\lambda_1+\lambda_2)\Delta t & 0 & 0 \\ \lambda_1\Delta t & 1 & \lambda_2\Delta t & 0 \\ \lambda_2\Delta t & 0 & 1-\lambda_1\Delta t & 0 \\ 0 & \lambda_2\Delta t & \lambda_1\Delta t & 1 \end{bmatrix}$$

그리고,

$$\mathbf{R}(t)_{single}=P_{11}(t)$$

이다.

3. 신뢰도 비교

앞에서 구한 확률 상태 벡터를 이용하여 신뢰도 함수로 나타내어 보자.

우선 고장률 λ_1은 MLC의 고장률인데, 논문[16]에 따라 정했다. 일반적으로, 구조가 간단하고 사용되는 소자가 적을수록 좀 더 고장이 적게 일어난다. 그

래서, SIB는 MLC에 비해 소자의 수가 상대적으로 적기 때문에 고장률도 그만큼 적게 정했다.

그림 6(a)는 어떠한 고장이더라도 완전한 조치가 일어나도록 MLC와 SIB 모두 이중화했을 때와 MLC만 이중화했을 때, SIB만 이중화했을 때, 그리고 모두 단일화했을 때와의 신뢰도 비교인데, 단일구조의 경우는 어느 한 소자라도 고장이 생기면 신뢰성이 없는 것이므로, 완전하게 고장이 처리되는 경우와 비교하였을 때 그만큼 신뢰도가 떨어진다. 그래서 모두 이중화한 경우와 모두 단일화한 경우와는 신뢰도가 상당히 차이남을 볼 수 있다. 또한 MLC만 이중화 했을시와 SIB만 이중화 했을시의 신뢰도는 SIB가 MLC보다 낮은 고장률에도 불구하고 MLC를 이중화했을 경우에 더 신뢰도가 높게 나타났는데 그 이유는 MLC는 hot stand-by 구조로 SIB의 경우보다 많은 방법에 의해 고장을 탐지하여 조치를 취해주기 때문으로 SIB를 이중화한 cold stand-by 보다 MLC를 이중화한 hot stand-by가 신뢰도가 높음을 알 수 있다. Johnson의 분석[16]에서는 hot stand-by 형태가 cold stand-by 형태보다 신뢰도가 낮다고 하였으나, Johnson은 hot stand-by에서 비교 과정에 의해 고장이 발견됐을 경우에는 어느 모듈이 고장인지 모르기 때문에 cold stand-by가 더 신뢰도가 높다고 분석 되었는데, 본 논문에서는 비교 과정에 의해 고장이 탐지되었을 경우에도 정해준 규칙에 의해 어느 모듈이 고장인가를 결정할 수 있기 때문에 hot stand by가 cold stand-by 형태 보다 신뢰도가 높게 분석되었다.

여기서는 MLC의 고장을 MLC의 자체 모듈 테스트와 두 MLC 사이의 출력값의 비교에 의해서만 고장이 서로 탐지된다고 정했으며, 실제로는 완전하게 모든 고장이 탐지되어, 조치가 취해지지 않을 수도 있기 때문에 고장이 발견될 확률을 0.9로 잡고, 각 경우에 0.7, 0.5로 확률이 낮아질 경우에 대해 비교를 했다. 그리고 비교는 주로 MLC와 SIB 모두 단일구조인 경우와 두 모듈 모두 이중구조인 경우를 중심으로 비교하였다.

그림 6(b)는 C와 C_c가 각각 0.9이고 C_d가 0.9, 0.7 0.5일 때 단일 구조와의 비교이다. MLC의 고장 판단에 C_c, C_d가 똑같은 효과로 작용하기 때문에 C 와 C_d가 각각 0.9이고 C_c가 0.9, 0.7, 0.5일 때 단일 구조와의 비교시에도 그림 6(b)와 같은 결과를 가져온다. 즉, 일반적으로 비교과정에 의해 고장이 탐지되면, 고장이 생긴 정확한 위치를 알 수 없기 때문에 실제로 두 MLC 중 어느 MLC가 고장인지 두 MLC 사이의 판단 만으로는 결정을 내리기 곤란하나, 제3

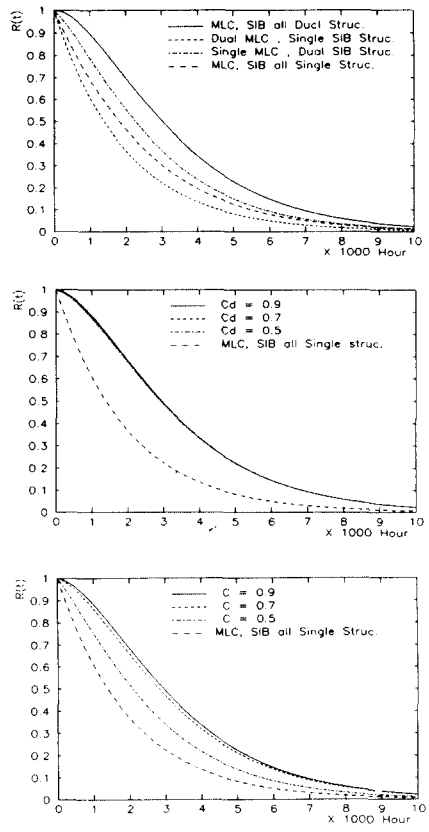

그림 6. (a) 완전 고장 탐지시 MLC, SIB 모두 이중구조, MLC만 이중 구조, SIB만 이중 구조, MLC, SIB 모두 단일 구조와의 비교 ($C=C_c=C_d=1.0$, $\lambda_1=0.00035$/hour, $\lambda_2=0.00015$/hour)
(b) MLC, SIB 모두 이중구조에서 C_d의 변화에 대한 것과 MLC, SIB 모두 단일 구조와의 신뢰도 비교 ($C=C_c=0.9$, $\lambda_1=0.00035$/hour, $\lambda_2=0.00015$/hour)
(c) MLC, SIB 모두 이중구조에서 C의 변화에 대한 것과 MLC, SIB 모두 단일구조와의 신뢰도 비교 ($C_c=C_d=0.9$, $\lambda_1=0.00035$/hour, $\lambda_2=0.00015$/hour)

Fig. 6. (a) Reliability comparison with both MLC and SIB dual, only MLC dual, only SIB dual, both MLC and SIB single at exact fault detection, ($C=C_c=C_d=1.0$, $\lambda_1=0.00035$ /hour, $\lambda_2=0.00015$/hour)
(b) Reliability comparison with both MLC and SIB dual w. r. t. C_d, both MLC and SIB single, ($C=C_c=0.9$, $\lambda_1=0.00035$/hour, $\lambda_2=0.00015$/hour)
(c) Reliability comparison with both MLC and SIB dual w. r. t. C, both MLC and SIB single. ($C_c=C_d=0.9$, $\lambda_1=0.00035$/hour, $\lambda_2=0.00015$/hour)

장에서 정해준 기준에 의하여 판단을 내리면 어느 한 모듈의 고장을 결정하고 이에 따라 조치를 취할 수 있다. 그 기준에 따라 고장이라고 판단을 내리면, 자체 진단에 의한 경우와 같은 확률일 때는 동일한 효과로 고장을 탐지하게 되므로 같은 신뢰도를 보인다.

그림 6(c)는 MLC의 진단에 의해 SIB의 고장이 탐지되어 성공적으로 조치가 취해질 확률의 변화에 따른 단일구조와의 신뢰도 비교인데, 그 확률이 낮아질수록, SIB의 상대적으로 낮은 고장률에도 불구하고, 신뢰도가 급격히 떨어지는 것을 볼 수 있다. 그 이유는 MLC는 여러가지에 의해 고장을 탐지하고 조치를 취하지만, SIB는 MLC같이 스스로 고장을 탐지하지를 못하고 MLC의 판단에만 의존하기 때문에 MLC가 고장을 제대로 탐지를 못해주면 그만큼 신뢰도가 급격히 떨어진다고 볼 수 있겠다. 즉 SIB의 고장시에 MLC가 조치를 취해주기 때문에 MLC 자체의 신뢰성이 시스템 전체의 신뢰도를 높이기 위해 보다 중요함을 알 수 있다.

V. 실 험

모듈 내의 소자에 고장이 났을때, 백업 기능을 확인하기 위해 임의로 고장을 발생시키지만, 고장 발생을 위해 고가인 하드웨어의 소자에 손상이 가게 하거나 제거할 수는 없기 때문에 간단히 고장과 같은 상황을 고의로 발생시켜 처리하였다.

SIB의 고장중 동작에 치명적인 영향을 끼치게 되는 인터페이스 부분의 고장은 고장시에 MLC로 데이타 전달 인식 신호를 보내주지 않는 것이다. 그래서, 데이타 전달 인식 신호를 보내주지 못하도록 모듈을 뽑았는데, 이때 처리되는 순서는 다음과 같다.

i) MLC와의 인터페이스부의 고장 발생
ii) MLC가 SIB로부터 데이타를 입출력하려할 때 예외처리 버스 에러발생(fault detection)
iii) 어느 SIB에서 발생되었나를 세워진 플래그로부터 찾음. (fault locatipon)
iv) SIB-B로 펄스를 보내주어 스윗칭. (fault isolation, reconfiguration)
v) MLC-B에 SIB 고장이 생겼음을 알리는 플래그를 세움으로써 그 sampling 시간에서의 제어출력 값의 비교는 하지않도록 함. (fault broadcasting)
vi) 메인 루틴으로 되돌아와 계속 자체 테스트. (회복)
vii) 다음 sampling 시간에는 스윗칭된 SIB로 데이타 입출력.

또 다른 SIB의 고장도 위와 비슷한 방법으로 처리가 되며, 주로 고장의 발생은 점퍼선을 사용하여 오픈 또는 단락 등에 의해 실험하였다.

MLC에서의 고장은 고장 시에 세워질 플래그를 임의로 바꾸어 고장 상황을 만들었다. MLC의 메모리 테스트 고장시에 MLC-B는 다음과 같은 과정으로 MLC-A의 고장을 인지하며 백업 조치를 취했다.

i) MLC-A가 자체 테스트 중 메모리 고장이 있었다는 플래그 세움.
ii) MLC-B가 확인하여 MLC-A를 멈추게 하는 플래그 세움
iii) 다음 샘플링 시간의 처음으로 간다.
iv) 기존에 갖고 있던 제어 알고리듬과 데이타로부터 다시 제어 시작.

위와 같이 몇가지의 고장 발생에 대해서 실험했을 시에 조치를 잘 해줌을 알 수 있었다.

실제로 local 버스에는 SIB 모듈이 8개까지 꽂힐 수 있으며, 모두 이중화하면 최대 16장이 꽂히게 되는데, 그림 7은 초기화 과정을 거친 후 동작 중에 SIB 5번에서 예외처리 버스 에러가 발생했을 시에 백업을 해 주었음을 모니터로 디스플레이(display)한 것이다. 그림 8은 MLC-A가 SIB 5번을 백업하고 제어하는 도중에 인위적으로 리셋(reset)을 통해 멈추게 하여 MLC-B가 백업 제어를 하도록 한 다음 이 때의 S-RAM 상의 플래그를 디스플레이한 것이다. F48020번지의 'F1'은 MLC 백업 중 임을 의미하고, '32', '63'은 각 MLC 내부에서 증가시키는 카운트(count)를 나타낸 것이며, F48035번지의 '05'는 SIB 5번을 백업하고 있음을 나타낸다. 그리고 F48050번지부터 세워진 '53'은 MLC-B가 계속 동작중임을 의미하고, '35', '37'은 MLC-A가 멈춰지기 전에 SIB 5번이 고장났을 때 MLC-A가 세운 것이며 '5D'는 MLC-A가 멈추어지기 전의 비교 과정에서 이상이 없었음을 의미하며, '3F'는 MLC-A가 멈추었을 때 MLC-B가 이를 확인하고 세운 플래그이다.

그림 9는 실제 발전소의 보일러를 대신하는 시뮬레이터(simulator)를 제어하는 도중에 고장이 발생하여 조치를 취한 것을 보인 것인데, 관리 제어시스템을 통해 공기 다이나믹스(air dynamics)를 모니터링한 것이다. 즉, SIB 0번 모듈을 뽑아서 고장을 발생시켰을 때 우상단의 0번에 불이 들어와 SIB 0 번의 백업을 나타내 줌을 볼 수 있으며, 모듈을 뽑는 순간 피이크(peak)가 잠시 생겼다가 곧 정상 상태로 가는 것을 볼 수 있다. 이때 생긴 피이크는 SIB 모듈의 고장으로 이상 신호가 받아들여졌기 때문이라고 생각된다. 또한, 우상단 A에 불이 들어온 것은

```
SIMU_2 > .A7 30000

SIMU_2 > .SR 2700

SIMU_2 > G 2000
PHYSICAL ADDRESS=00002000
TIMER Test Good
CPU Register Test Good
CPU ALU Test Good
CPU Exception Test Good
LOCAL ADDR BUS GOOD
GLOBAL ADDR BUS GOOD
DATA BUS TEST GOOD
Common Memory Good
Exception Vector init
Timer Setted
Now Operating ....

 SIB BUSERR
sib5 DTACK fault
sib13 Reconfigured


SIMU_2 >
```

그림 7. SIB 고장 실험 모니터링
Fig. 7. The monitoring of SIB fault experiment.

```
SIMU_2 > MD F48020
F48020    F1 00 00 00 00 00 00 32  00 00 00 63 00 00 00 00  q.......2...c....

SIMU_2 >
F48030    00 00 00 05 00 00 00 00  00 00 00 00 00 00 00 00  ................
F48040    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
F48050    00 53 35 00 37 00 00 00  50 3F 00 00 00 00 00 00  .S5.7...)?......
F48060    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
F48070    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
F48080    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
F48090    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
F480A0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
F480B0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
F480C0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
F480D0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
F480E0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
F480F0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
F48100    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
F48110    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
F48120    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................

SIMU_2 >
```

그림 8. 상태 플래그의 내용
Fig. 8. The contents of status flags.

이 상태에서 MLC-A가 시뮬레이터를 제어하는 것을 멈추게 했을 때 MLC-B로의 백업을 나타낸 것이다. 이 때는 SIB 모듈을 뽑을 때처럼 피이크가 생기지 않았는데, 그 이유는 I/O 모듈 스윗칭이 아니기 때문에 이상 출력 또는 이상 입력이 잠시라도 나타나지 않게 되며, MLC의 스윗칭이 끝날 때까지 시뮬레이터와의 데이타 입출력을 중단하고 기존의 출력값을 유지하고 있기 때문이다.

비교 과정에서의 threshold는 5%로 정하여 실험했

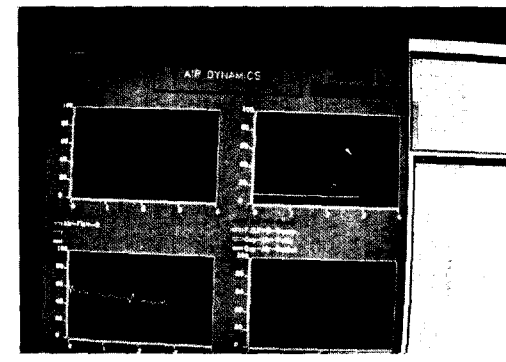

그림 9. 관리 제어시스템에서 본 고장 발생 상황
Fig. 9. The generated fault status monitored at supervisory control system.

으며, 두 MLC 모두 고장시에 수동절환이 되어 시뮬레이터를 보호함을 확인하였다.

Ⅵ. 결 론

본 논문에서는 제어시스템의 신뢰도를 높이기 위해 CPU 모듈과 I/O 모듈을 각각 이중화한 제어 구조를 제안했으며, 신뢰도 함수를 이용하여 이중화된 시스템의 신뢰도를 정량적으로 분석했다. 정량적인 신뢰도 분석을 통해 hot stand-by 구조의 CPU 모듈과 cold stand-by 구조의 I/O 모듈에서, 고장을 적절히 잘 탐지하여 스윗칭을 하면 단일한 구조보다 상당히 신뢰도가 높음을 보였다. CPU 모듈만 이중화했을 때와 I/O 모듈만 이중화했을 때가 I/O 모듈만 이중화했을 때보다 신뢰도가 높다는 결과를 얻었다.

또한, 고장의 탐지 방법을 세분화함으로써 고장시에 고장을 탐지하고 백업해 줄 확률을 높여 완전한 이중화 구조가 될 수 있도록 했다. 또 플랜트의 안정을 위해 두 CPU 모듈 또는 I/O 모듈이 모두 고장이 나더라도 자동으로 수동절환토록 하는 기능도 포함시켰다. 그리고, 임의로 몇가지 고장 상황에 대해 백업 조치를 하는 것을 실험을 통해 보였다.

그런데, 위에서 신뢰도 함수를 계산시에, 이중화 구조로 구성시 첨가된 스윗칭 로직과 로컬 버스 아비터는 되도록 간단하게 구성했기 때문에, 고장이 전혀 없다고 가정을 했지만, 실제로 약간의 고장 가능성이 있다. 그리고, 이 고장 가능성이 전체 시스템에 큰 영향을 줄 수도 있는데, 이를 좀 더 개선 보완하여 이 부분의 고장에 의해 전체 시스템이 오동작하지 않도록 해야할 것이다. 그리고, 스윗칭 로직과 로컬 버스 아비터의 고장 가능성을 포함한 Markov

모델을 구하고, 이로부터 좀 더 복잡하지만 구체적인 신뢰도 함수를 구해보는 것은 추후 과제로 남겨 놓는다.

본 이중화된 제어시스템의 CPU 모듈은 VME 버스 규격에 따라 제작된 CPU 모듈을 변형시켜 구성했기에 여러 제약점이 있었다. 우선, 같은 전원을 사용하므로, 전원 고장시에는 두 시스템이 동시에 멈춰질 수 있으며, 보다 신뢰도를 높이기 위한 중복 버스 구조를 취할 수 없는 제약점이 있다. 위와 같은 제약점을 극복하여 상호 의존도가 낮은 이중화된 시스템을 구성하는 경우에 대한 연구가 좀 더 필요하다.

參 考 文 獻

[1] Theordore J. Williams, "The development of reliability in industrial control systems," *IEEE MICRO,* pp. 66-80, 1984.

[2] 조영조, "가법적 중복 적응 제어기를 이용한 제어시스템의 신뢰도 향상에 관한 연구," KAIST 박사논문, 1989.

[3] "마이크로 컴퓨터를 이용한 전자제어 시스템의 고신뢰화에 관한 연구"(최종보고서), 한전 기술연구원, Feb. 1988.

[4] 신영달, "Bolier backup control을 위한 Multipr processor 방식에서의 신뢰도 개선에 관한 연구," KAIST 석사논문, 1987.

[5] 정광균, "화력 발전소용 보일러 제어기를 위한 백업콘트롤 시스템에 관한 연구," KAIST 석사논문, 1987.

[6] 허성광, "발전소 제어계통의 고장 진단," KAIST에서의 세미나 자료, May. 13. 1989.

[7] Astrom, Wittenmark, "Computer controlled systems," Prentice Hall, 1984.

[8] 이헌, "Fauit tolerant computing systems," 전자교환 기술 제 1 권 제 1 호, pp. 46-61, 1989.

[9] William F. McGill, Steven E. Smith, "Fauit tolerace in continuous process control," *IEEE MICRO, pp.* 22-33, 1984.

[10] David A. Rennels, "Fault tolerant computing concepts and examples," *IEEE Trans. on Computers* vol. C33, no. 12, pp. 1116-1129, Dec. 1984.

[11] "Bailey INFI 90", Catalog, U.S.A. 1988.

[12] "Bailey network 90 (Multi-Function Controller Module-NMFC05)", Catalog, E93-906-13, 1988.

[13] "Taylor MOD systems," Catalog, SDS-32 E001 Issue 4, Feb. 1987.

[14] "Bailey controls network 90 distributed control systems," Catalog.

[15] Chales H. Roth, "Fundamentals of logic design," West Publishing Company, 1985.

[16] Barry W. Johnson, "Reliability & safety analysis of a fault-tolerant controller," *IEEE Trans. on Reliability* vol. R-35, no. 3, pp. 15-524, 1983.

[17] D. D. Siljak, "Reliable control using Multiple Control Systems," *INT. J. Control,* vol. 31, no. 2, pp. 303-329, 1980.

[18] G.S. Ladde, D.D. Siljak, "Multiplex control systems: stochastic stability and dynamic reliability," *Int. J. Control,* vol. 38, pp. 515-524, 1983.

[19] Motorola Inc, "VME bus specification manul," Aug. 1982 .

[20] 이재혁, "자동 시스템을 위한 진단 기능을 갖는 구조적 관리제어에 관한 연구," KAIST 석사논문, 1988.

[21] Motorola, "MVME 110 VME module monoboard microcessor user's manual."

[22] "발전소 제어용 디지탈 계장제어 시스템 개발 (중간 보고서)," 한전기술연구원, Feb. 1989.

[23] Lance A. Leventhal, "68000 assembley language programming," McGraw-Hill International Editions, 1986.

[24] Motorola, "M68000 16/32-bit microprocessor programmer's reference manual,' 1987.

[25] 박세화, "이중구조를 갖는 제어시스템의 구현과 신뢰도 분석에 관한 연구," KAIST 석사논문, 1990.

著 者 紹 介

朴 世 華(準會員)
1965年 6月 15日生. 1988年 2月 서울대학교 공과대학 전자공학과 졸업. 1990年 2月 한국과학기술원 전기 및 전자공학과 공학석사 학위취득. 1990年 3月～현재 한국과학기술원 전기및 전자공학과 박사과정 재학중. 주관심분야는 내고장성 제어및 내고장성 제어시스템 설계 등임.

●

文 鳳 彩 (正會員) 第27卷 第1號 參照
현재 한국과학기술원 전기및 전자공학과 박사과정 재학중

卞 增 男 (正會員) 第27卷 第1號 參照
현재 한국과학기술원 전기및 전자공학과 교수

●

金 炳 國 (正會員) 第27卷 第1號 參照
현재 한국과학기술원 전기및 전자공학과 공학박사