핫 스탠바이 스페어링 기법을 이용한 고장 감내 이중화 시스템 설계

정회원 신 진 욱*, 박 동 선*

The Implementation of Fault-Tolerant Dual System Using the Hot-Standby Sparing Technique

Jin-wook Shin*, Dong-sun Park* ***Regular Members***

요 약

분산 컴퓨팅 기술 발달과 인터넷 이용의 확산에 따라 고속의 멀티미디어 서비스에 대한 사용자의 욕구가 날로 증가하고 있다. 이에 따라 영상, 음성 등이 포함된 대용량 정보매체를 다루는 서비스가 주로 이루어지고 있으며 망 사업자들은 이러한 대용량 정보매체의 고속 전송이 가능하도록 초고속 네트워킹 설비에 끊임없이 투자하고 있다. 이와 같은 빠른 속도의 서비스뿐만 아니라 이와 동시에 만족되어야 하는 서비스의 요건은 안정성이다.

시스템 고장으로 인하여 기반 시설이 마비될 수 있는 전자 정보 시스템은 매우 높은 가용성 및 신뢰성을 가져야 한다. 이러한 고가용성과 고신뢰성을 얻기 위하여 본 논문에서는 핫 스탠바이 스페어링 기법을 이용한 고장 감내 다중화 시스템을 제안하고 구현한다. 제안된 시스템은 일반적인 단일 모듈 시스템을 다중화 하여 고장이 발생하면 유연하게 대처하도록 하고 고장 검출 버스를 적용하여 비교를 통한 고장 검출 기능이 가능하도록 하였다. 또한 제안된 구조는 단일 모듈 시스템에 버스 변환 장치를 도입하여 보다 쉽게 고장 감내 다중화 시스템을 구현할 수 있도록 하였다. 그리고 본 논문에서 제안한 하드웨어 시스템의 성능 평가를 위하여 마코프 프로세스를 이용한 모델링을 적용하여 고가용성 및 고신뢰성을 검증하였다.

ABSTRACT

This paper is basically to achieve the high-availability and high-reliability of the control system from the implementation of the fault-tolerant system using the hot-standby sparing technique. To meet the objective, we design and implement a board with fault tolerance I/O bus to detect the fault.

Warm-standby sparing technique is the fault tolerance technique usually used for switching control system in present. This technique can be easily implemented, but can not detect the fault quickly and can malfunction because of the hardware fault. The hot-standby sparing fault tolerant technique implemented in this paper is consists of dual processor modules and a I/O processor using fault tolerant I/O bus. The proposed method can find the faults as soon as possible, so it can prevent from wrong operation. Also it is possible to normal re-service due to the short recovering time.

To implement the fault-tolerant dual system with fault detection bus, two daughter, called FTMA and FTIA, boards designed and implemented are applied to the system. And we also simulated the proposed method to verify the high-availability and high-reliability of the control system using Markov process.

* 전북대학교 전자정보공학부 멀티미디어연구실(jwshin@chonbuk.ac.kr)
논문번호 : 040084-0219, 접수일자 : 2004년 2월 24일

Ⅰ. 서 론

현대사회에서의 전자정보 시스템들은 사회 기반의 중요한 부분을 차지하게 되면서 시스템의 신뢰성 및 가용성이 중요한 문제로 부각되고 있다. 예를 들어 교환기, 인공위성, 은행 결재 시스템 등의 고장 및 오류는 생활의 불편함을 초래할 뿐만 아니라 사회적 혼란을 야기할 수도 있으며 의료장비의 경우 생명을 위협할 수도 있기 때문이다. 따라서 이러한 전자정보 시스템들의 고장을 최대한 빨리 검출하고 복구하여 신뢰성 및 가용성을 높일 수 있도록 연구가 활발히 진행되고 있다.

본 논문에서는 시스템에서 발생할 수 있는 고장을 검출하고 복구하는 방식으로 하드웨어를 다중화하는 방식을 이용하여 적용한다. 하드웨어 다중화 방식은 서비스를 수행하는 모듈, 서비스 수행 모듈과 동일한 형태의 여분 모듈들로 시스템을 구성하여 서비스 수행중인 모듈에 고장이 발생하면 여분의 모듈이 서비스를 계속 수행할 수 있도록 한 방식이다. 이러한 구성은 고장이 발생하더라도 정상적인 서비스를 계속 수행하는 특징을 가지고 있다. 이와 같은 하드웨어 다중화 방식에는 콜드 스탠바이 스페어링 , 웜 스탠바이 스페어링, 핫 스탠바이 스페어링 기법들이 있다[1].

콜드 스탠바이 스페어링 기법은 서비스 수행모듈이 정상적으로 서비스중인 경우에는 여분의 모듈들은 전원 무공급 상태로 유지하며 현재 서비스중인 모듈에 고장이 발생하면 여분의 모듈에 전원을 공급하여 서비스를 대신 수행하는 방식이다. 이 기법은 전력 소모를 줄일 수 있는 장점은 있으나 서비스를 정상적으로 수행할 때까지의 시간이 오래 걸리며 하드웨어 고장을 검출하지 못하는 경우가 발생할 수 있다.

웜 스탠바이 스페어링 기법은 서비스 수행 모듈이 정상적으로 동작중인 경우에는 콜드 스탠바이 스페어링 기법과는 달리 여분의 모듈들을 전원 공급 상태로 유지시키면서 정상적으로 서비스중인 수행모듈의 정보를 부분적으로 동일하게 유지시켜 주는 방식이다. 이 기법은 정상적인 서비스가 수행되기까지의 시간은 단축할 수 있으나 이 방식도 하드웨어 고장을 검출하지 못하는 경우가 발생할 수 있다.

핫 스탠바이 스페어링 기법은 서비스 수행 모듈과 여분의 모듈들을 동시에 전원 공급 상태로 유지시키면서 동일시간 동일동작을 수행하도록 밀결합 동기 상태를 유지하는 방식이다. 이 기법은 서비스 수행중인 모듈 또는 여분의 대기 수행중인 모듈에서 발생할 수 있는 하드웨어의 고장 검출이 가능하므로 시스템의 고가용성 및 고신뢰성을 높일 수 있다[2]. 그러나 이 방식은 다중화 모듈간의 밀결합 동기와 동일한 정보 자원을 유지해야 하는 어려운 점이 있다.

본 논문에서는 전자정보 시스템의 고가용성 및 고신뢰성을 위하여 핫 스탠바이 스페어링 기법을 이용한 고장 감내 다중화 시스템을 설계하고 성능을 평가한다.

본 논문의 구성은 2장에서 기존의 고장 감내 시스템에 대하여 살펴본 후 3장에서는 본 논문에서 제안한 비교 기법을 이용한 핫 스탠바이 스페어링 방법에 대하여 기술한다. 마코프 프로세서를 이용한 고장 감내 이중화 시스템의 성능평가는 4장에서 이루어지며 5장에서 결론을 맺는다.

Ⅱ. 기존 고장 감내 시스템의 구조

본 장에서는 기존의 고장 감내형 시스템 구조들인 프로세서 단계에서의 비교 구조, Stratus XA 구조, Tandem Integrity S2 구조 그리고 FT-SPARC 구조 등에 대하여 살펴본다[3][4].

2.1 프로세서 레벨 비교 구조

프로세서 레벨 비교 기법은 가장 낮은 레벨에서의 비교 작업을 통한 고장 감내 시스템의 구조로 그림 1처럼 프로세서의 입력과 출력들이 프로세서 버스 사이클마다 비교되어진다. 이는 프로세서의 오류발생을 검출하는 가장 간단한 방법으로 프로세서에서 발생할 수 있는 거의 모든 경우의 오류를 정확한 시점에서 검출할 수 있는 장점을 가지고 있다. 그러나 이 방식은 비교 시간이 많아져서 프로세서의 성능저하를 초래하며 또한 과거 프로세서들의 속도가 느릴 때 가능한 방법이었으나 현재의 프로

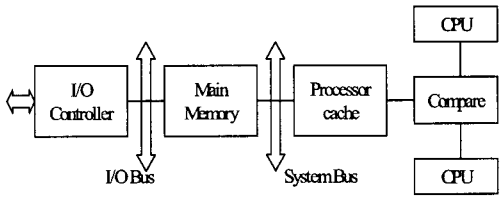

그림 1. 프로세서 레벨 비교 구조

세서들처럼 빠른 작업 처리 속도를 고려할 때는 불가능한 방법이라고 할 수 있다. 그림 1은 CPU 만이 이중화 된 구조로써 CPU가 프로세서 캐쉬나 메인 메모리, 입출력 콘트롤러에 접근할 경우에 비교 작업을 수행한다.

2.2 Stratus XA 구조

그림 2와 같은 구조로 구성되는 Stratus XA 시스템은 두개의 프로세서 모듈로 구성되며 각 프로세서 모듈에는 서로 비교할 두 개의 프로세서가 존재한다. 이 구조의 특징은 프로세서 모듈이 동작하는 액티브 프로세서 모듈과 대기중인 백업 프로세서 모듈로 나뉘어져 오류 발생 시에 백업 모듈로 대체가 가능하도록 하였다. 백업 모듈을 이용하여 이중화를 구현한 것으로써 CPU와 프로세서 캐쉬가 하나의 프로세서 모듈에 포함되어 있으며 시스템 버스와 입출력, 버스 입출력 컨트롤러들을 이중화하였다. 그러나 프로세서 레벨 비교 기법과 마찬가지로 비교 횟수의 빈번함으로 인한 성능저하의 단점을 가지고 있다.

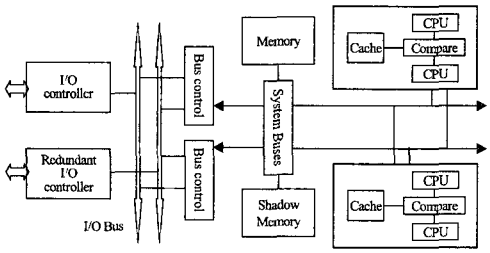

그림 2. Stratus XA 구조

2.3 Tandem Integrity S2 구조

그림 3과 같은 Tandem Integrity S2 구조에서 단일 프로세서 모듈은 프로세서와 세컨더리 캐쉬, 로컬 메모리로 구성된다. 이 구조에서는 CPU가 캐쉬나 로컬 메모리에 접근할 때는 프로세서들끼리 비교 작업 없이 동작하도록 한다. 그러나 메인 메모리나 외부 입출력 장치에 접근할 경우에만 비교 작업을 수행하도록 하여 프로세서 레벨 비교 보다 비교 횟수를 현저히 줄임으로써 오버헤드를 줄일 수 있는 장점을 가지고 있다. 그러나 이 구조에서는 각 프로세서는 자신의 클럭으로 동작하므로 모듈간 동기가 불일치 되는 문제가 발생한다.

2.4 FT-SPARC 구조

FT-SPARC 구조는 핫 스탠바이 스페어링 기법을

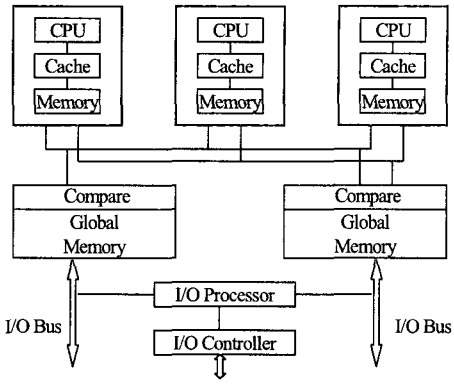

그림 3. Tandem Integrity S2 구조

적용한 예로 그림 4처럼 캐쉬와 메인 메모리를 포함하는 FT-SPARC CPU 집합들로 구성되어 있으며, CPU 집합간에는 공통의 기준 클럭으로 동작하도록 한 밀결합 동기 방식을 사용한다. 이 구조는 비교작업이 입출력 버스를 액세스할 경우에만 수행되므로 비교 오버헤드를 줄일 수 있는 장점을 제공한다.

그림 4의 구조에서는 삼중화된 구조를 채택하여 다수결 원칙에 의거하여 고장 모듈을 검출하고 고장으로 판명된 모듈은 고장 진단 및 수리 모드로 전환되게 된다. 이 구조는 모든 시스템 모듈이 정상 동작 할 경우에는 액티브 상태에 있으며 외부 시스템과의 데이터를 주고받을 때는 한 모듈만이 마스터로써 담당하며 다른 모든 시스템 모듈은 마스터

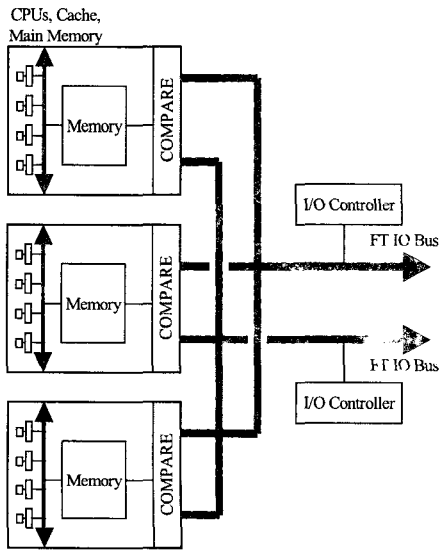

그림 4. FT-SPARC 구조

모듈의 상태와 내용이 동기과정을 통해서 동일하게 유지되도록 하는 것이 특징이다.

2.4.1 이중화 고장 감내 프로세서 모듈 구조

그림 5는 FT-SPARC 구조를 자세히 나타낸 것으로 두 프로세서 모듈 중 한 모듈이 마스터 프로세서 모듈이 되며 나머지 한 모듈은 슬레이브 모듈이 되어 비교를 수행한다. 마스터 모듈은 클럭 라인을 이용하여 두 프로세서 모듈에 동일 클럭을 제공하여 동기를 맞춘다.

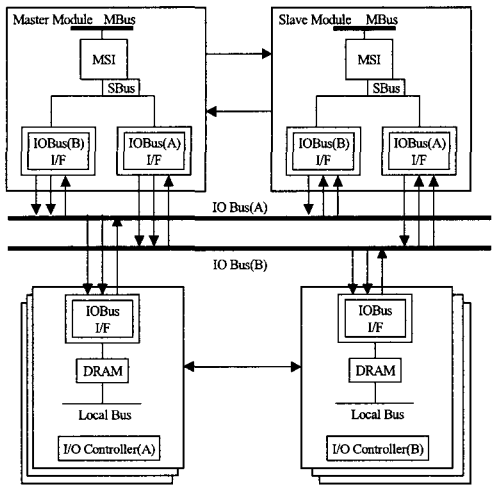

그림 5. 이중화 고장감내 프로세서 모듈 구조

또한 외부 입출력 컨트롤러로의 실제적인 데이터 전송은 마스터 프로세서 모듈이 담당하며 I/O Controller를 이중화 모듈로 구성하여 고장이 발생할 경우에 대체하도록 하였다.

III. 고장 감내 시스템의 설계

본 논문에서는 FT-SPARC 구조와 유사한 핫 스탠바이 스페어링 기법의 고장 감내 이중화 시스템을 구현한다. 제안된 설계에서는 공통의 입출력 장치로 구성되며 입출력시 서로의 결과를 비교하여 오류를 검출한다. 또한 밀결합이 가능하기 위하여 동일 클럭으로 이중화 시스템이 동작하도록 클럭부를 설계하였다.

3.1 고장 감내 다중화 시스템 모델

그림 6은 이중화 시스템이 아닌 일반적인 단일 모듈 형태의 시스템 하드웨어 모델을 간단하게 나타낸 것이다. 이 시스템은 그림 6에서 보는 바와 같이 프로세서 모듈이 CPU, 메모리, 시스템 버스, 시스템 버스와 입출력 버스간의 브릿지(S2I), 입출력 버스, 입출력 장치들로 구성된다. 이 구조는 고장을 검출하거나 고장이 발생하였을 경우 대처할 수 있는 능력이 부족한 시스템이다.

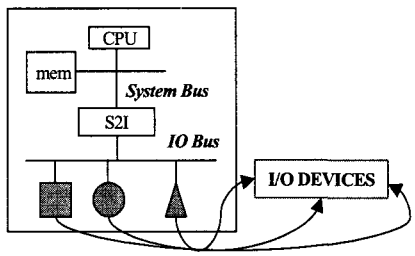

그림 6. 단일모듈 시스템

그림 7은 단일 모듈시스템을 이중화한 핫 스탠바이 스페어링 기법의 고장 감내형 이중화 시스템 모델이다. 프로세서 모듈은 마스터 모듈과 슬레이브 프로세서 모듈, 고장검출버스와 연결되어있고 실제 입출력 장치들을 갖는 입출력 프로세서 모듈이 이중화로 구성된 총 4개의 단일 모듈시스템이 필요하게 된다. 모든 시스템 하드웨어 자원들을 이중화함으로써 하드웨어 비용은 단일 시스템 모듈에 비하여 4 배가 소요되지만 고장 발생을 하드웨어적인 비교 작업 모듈을 통하여 검출할 수 있으며 고장 발생시 유연하게 대처하여 시스템의 가용성 및 신뢰성을 높일 수 있다.

본 논문에서 제안한 시스템의 구조적인 특징은 프로세서 모듈을 마스터와 슬레이브로 이중화한 것과 마스터와 슬레이브 모듈의 입출력 버스에 연결할 SCSI, 이더넷 등 자체 입출력 장치들을 별도의 모듈에 배치하여 입출력 작업을 수행하도록 한 것이다.

본 시스템을 구성하는 각 블록의 기능은 다음과 같다.

- S2I : System Bus 와 IO BUS 간의 인터페이스를 담당
- MXI : 메모리 쓰기를 감지하여 동시 쓰기를 제공
- Xbus : 메모리 동시 쓰기를 위한 채널

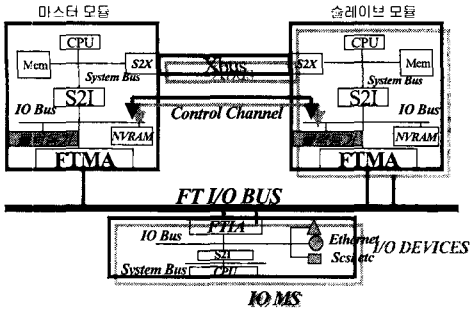

그림 7. 핫 스탠바이 스페어링 기법의 고장 감내형 삼중화 시스템 모델

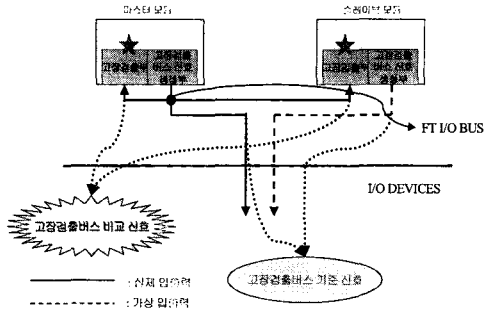

그림 9. 고장검출 버스 흐름도

· Control Channel : 이중화 프로세서 모듈간 정보를 주고 받을 때 사용
· Interrupt: 입출력 장치 프로세서 모듈에서 인터럽트를 요청
· FT I/O BUS : 핫 스탠바이 스페어링 기법의 고장 감내형 시스템에 적합하도록 제안된 버스
· FTMA : IO BUS 신호를 FT I/O BUS 신호로 변환시켜서 고장 검출할 수 있도록 하는 블록
· FTIA : DPRAM 이 존재하여 마스터 프로세서 모듈과 입출력 장치 프로세서 모듈간 정보를 주고 받기 위한 블록

3.2 고장 감내 다중화 시스템 설계

본 논문에서 제안한 핫 스탠바이 스페어링 기법을 적용한 고장 감내 이중화 시스템은 그림 8과 같으며 FT I/O 버스를 구현하여 이중화 모델을 설계하도록 하였다. 기존 상용 시스템을 이용하여 CPU는 SuperSparc, System Bus 는 Mbus, IO Bus 는 SBus 를 각각 사용하였다[5]. 또한 Xbus를 이용하여 동시 쓰기가 가능하도록 Xbus 인터페이스부와 (MXI), 클럭동기를 맞추기 위한 클럭동기부, *FT I/O BUS*를 이용한 하드웨어 비교 검출을 하기 위한 FTMA , FTIA DAUGHTER 보드를 구현한다. Xbus 는 이중화 모듈간 메모리 정보를 동일하게 유지하기 위하여 필요하며 클럭동기부는 두 프로세서 모듈이 동일 연산을 동일 클럭으로 수행하기 위하여 필요하다.

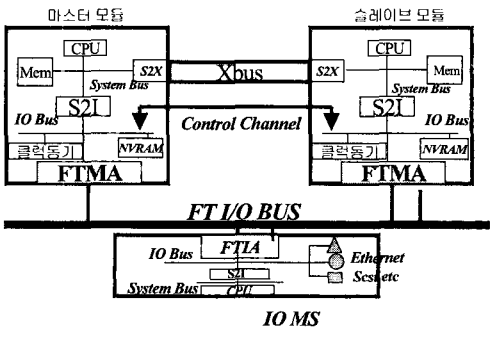

그림 8. 고장 감내 이중화 시스템

3.2.1 고장 검출 버스(FT I/O BUS)

그림 9는 동일한 모듈로 구성되는 마스터 모듈, 슬레이브 모듈이 입출력 장치들과 인터페이스 하는 방법을 보여주는 고장검출 버스 흐름도이다.

마스터 모듈과 슬레이브 모듈은 각각 고장 검출부와 고장 검출 버스 신호 생성부를 가지고 있으며 입출력 장치들과 데이터를 주고받는다. 그러나 실제로는 마스터 모듈만 고장 검출 버스를 통하여 데이터를 송수신하며 슬레이브 모듈은 마스터 모듈과 동일 작업을 수행하고 있지만 실질적인 데이터를

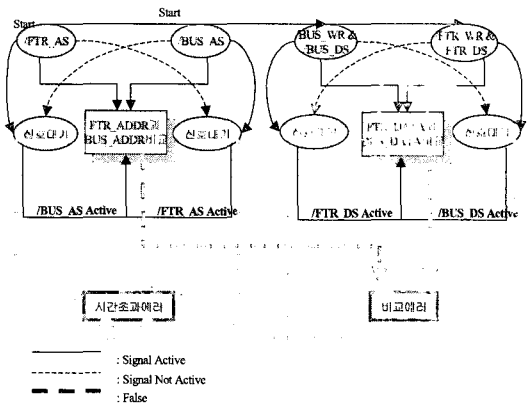

그림 10. 상태 비교 및 시간초과 오류 발생 천이도

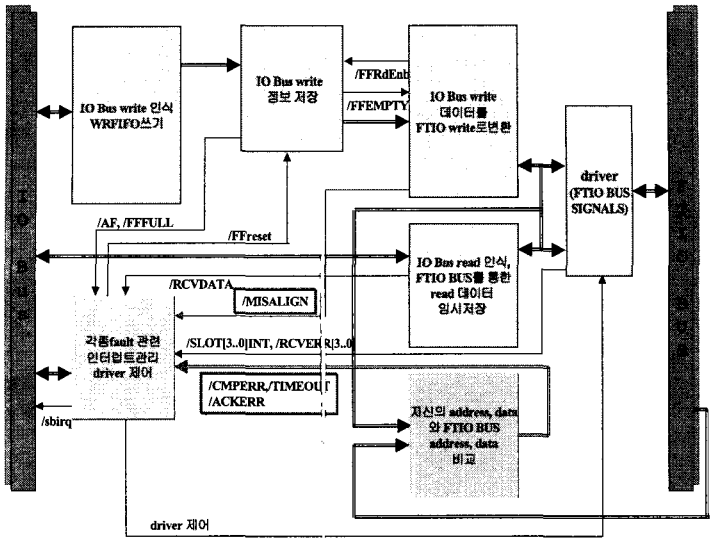

그림 11. FTMA 블록도

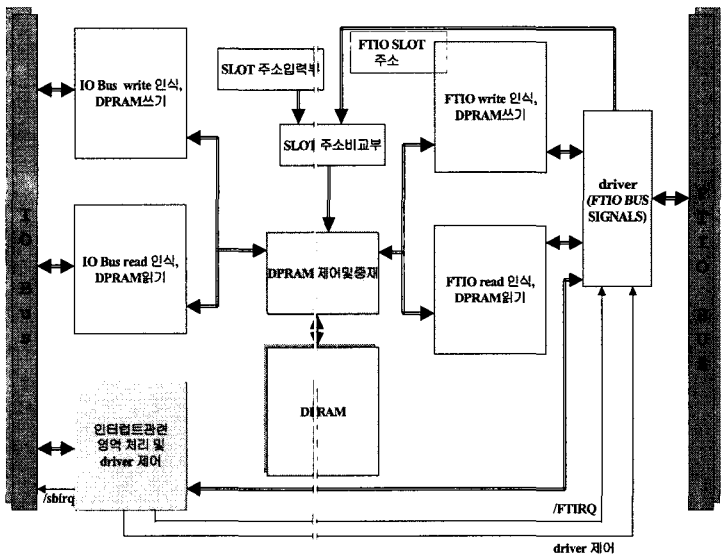

그림 12. FTIA 블럭도

주고받지는 않는다.

고장 검출부는 고장 검출 버스의 신호를 입력받아 각각의 모듈에서 생성한 고장 검출 버스 신호와 동일한지를 비교하여 고장 발생의 유무를 알린다.

그림 10은 고장 감내 이중화 시스템에서 검출할 수 있는 오류로 상태 비교 오류, 시간 초과 오류 등 두가지로 구분할 수 있으며 각각의 동작에서 발생할 수 있는 오류에 관한 상태 천이도이다.

비교 오류는 각각의 모듈에서 생성되는 주소와 데이터를 고장 검출 버스에 나타난 주소와 데이터를 비교하여 서로 일치하지 않을 경우 발생하며 시간초과 오류는 각 모듈간에 동기가 맞지 않을 경우 발생한다. 그림 10에서 */FTR_** 신호들은 고장 검출 버스 신호 생성부에서 생성하는 신호이며 */BUS_** 신호들은 고장 검출부로 입력되는 신호를 가리킨다. 신호대기 상태에서는 일정시간동안 상대 신호가 발생하지 않을 경우 시간초과 오류를 발생한다. 비교 오류는 */BUS_AS*와 */FTR_AS*신호가 둘 다 액티브

즉 신호가 있을 때 서로의 주소를 비교하여 정보가 같지 않을 경우 발생한다. 또한 데이터는 '쓰기' 동작일 경우에만 비교를 수행하며 (*/BUS_DS AND /BUS_WR*) 이고 (*/FTR_DS AND /FTR_WR*) 일 때 서로의 데이터를 비교하여 같지 않을 경우 비교 오류를 발생시킨다. 그리고 읽기를 수행 할 경우는 읽는 데이터의 패리티를 검사하여 데이터의 오류를 검출 할 수 있도록 한다.

3.2.2 버스 변환 장치(*FTMA, FTIA*)

버스 변환 장치는 프로세서 모듈의 *FTMA*와 입출력 장치의 *FTIA*가 있으며 입출력 버스와 고장검출 버스사이의 인터페이스를 담당하는 블록으로 핫 스탠바이 스페어링 기법의 고장 감내 시스템에서 중요한 역할을 하는 블록이다[6]. 고장을 검출하여 인터럽트를 통해서 중앙처리장치에 보고하며 입출력 장치와도 정보를 교환한다.

그림 11의 *FTMA*는 마스터 및 슬레이브 모듈에 위치하는 블록으로 입출력 장치로의 연산(*I/O Bus "읽기', '쓰기'*)을 수행할 경우 이를 인식하여 그에 해당하는 고장검출 버스(고장검출 버스 '읽기', '쓰기') 신호를 능동적으로 만들어주는 블록이다. 두개의 프로그램 가능 칩과 한개의 *FIFO*, *LIVE INSERTION* 기능을 추가하기 위한 드라이버 칩으로 구성되어 있다. 자신의 프로세서 모듈과는 SBus로 인터페이스가 이루어지며 입출력 프로세서 모듈과는 고장검출 버스와 상호 연결되어서 인터페이스가 이루어진다.

*FTMA*는 마스터와 슬레이브 모듈이 생성한 고장검출 버스 신호들을 비교하여 그 결과가 서로 다를 경우 발생하는 비교오류(*/CMPERR*), 두 모듈간의 동기 불일치로 발생하는 시간초과오류(*/TIMEOUT*), 입출력 장치 모듈과 인터페이스시 장치로부터 응답을 받지 못한 경우 발생하는 응답오류(*/ACKERR*), 고장검출 버스 '읽기' 연산 수행시 패리티 오류(*/RCVERR*)등의 고장을 검출한다.

그림 12의 *FTIA*는 입출력 장치의 모듈에 위치하는 블록으로 고장검출 버스의 신호에 따라서 입출력 장치 모듈 내의 입출력 장치들과의 인터페이스를 담당한다. *FTIA*는 고장검출 버스의 *'읽기', '쓰기'* 신호와 단순 입출력 버스의 *'읽기', '쓰기'* 신호를 입력받아 마스터 및 슬레이브 모듈과 입출력 모듈 사이에 정보를 교환할 수 있게 한다.

3.2.3 기타 블록들

Xbus와 관련된 블럭은 밀결합 동기로 동작하기 직전 단계로 모든 시스템 자원들을 동일하게 유지하기 위하여 사용한다. 즉 시스템 버스에서 발생하는 메모리 쓰기 신호를 하드웨어 방식으로 인식하여 인식된 쓰기 신호를 다른 모든 모듈 즉, 슬레이브 모듈에게 전달하여 메모리의 내용이 동일하게 유지되도록 할 때 사용한다. Xbus 메모리 쓰기는 시스템에 고장이 발생한 후 단일 모듈로 동작하다가 재동기 과정을 거쳐 다시 이중화 모드로 동작할 경우 메모리 정보를 일치시키기 위하여 이용된다. 따라서 밀결합 동기로 정상 동작을 수행할 경우에는 Xbus 와 관련된 블럭은 동작하지 않는다.

클록동기부는 자신과 상대방 모듈의 클록을 모두 받아서 어느 모듈의 클록을 시스템 클록으로 사용할지를 결정하여 밀결합 동기를 이루도록 한다.

3.3 고장 감내 이중화 시스템 운용 절차

본 논문에서 제안한 고장 감내 이중화 하드웨어 시스템을 관리하기 위해서는 동작 모드 및 운용 방법에 관한 관리 소프트웨어가 필요하다[7]. 그림 14는 핫 스탠바이 스페어링 기법의 고장 감내 이중화 시스템의 동작모드를 나타내며 크게 4가지 모드로 구성된다.

그림 13 (a)는 초기모드인 독립동작 상태로 시스템에 전원이 공급된 후 처음으로 수행하는 동작 모드이다. 이 모드에서는 각각 자신의 독립된 클록을 이용하여 동작하며 이중화 모드로 들어가기 직전의 준비 단계로 초기 설정을 한다. 각 모듈에 운영체제가 탑재되기 전 단계에서 기본적인 부트롬 상태에서 동작하게 된다.

(b)는 이중화 모드에서 고장 감내 이중화 시스템이 정상적으로 서비스를 수행하고있는 상태이다. 마스터 모듈은 슬레이브 모듈에 클록을 공급함으로써 밀결합 동기로 동작하며 마스터 모듈은 입출력 모듈과 실제로 인터페이스를 하고 슬레이브 모듈은 가상의 인터페이스 즉, 전기적으로 고장검출 버스에 신호를 보내지 않는다. 밀결합 동기 상태이므로 두 모듈은 같은 시간에 동일 정보를 유지해야 한다. (b)의 이중화 모드로 동작하다가 고장이 발생하면 (c) 모드로 천이하게 되며 이때 마스터 모듈은 슬레이브 모듈에게 마스터 권한을 양도하고 고장 복구 모드로 천이한다. 이 경우 변경된 마스터 모듈은 단일 시스템 모드로 동작하여 입출력 모듈과 실제로 데이터를 송수신하여 처리하게된다. 또한 (c) 모드에서는 이전 마스터 모듈이 자체 진단 과정 및 수

리를 거쳐 고장을 복구했을 경우에 다시 이중화 모드로 동작하기 위한 준비 단계가 필요하게 된다.

(d) 모드에서는 고장 수리를 끝낸 후 재설정 과정을 거쳐 다시 정상 동작 상태로 된다. 복구된 이전의 마스터 모듈 즉, 현재의 슬레이브 모듈은 서비스를 수행하고 있던 현재 마스터 모듈의 변경된 자원을 필요로 하며 변경된 자원을 모두 수신하여 두 모듈이 동일 상태를 유지하도록 한다. 그리고 클록 동기부를 이용하여 밀결합 동기를 설정하여 (b)의 이중화 모드로 천이하게 된다.

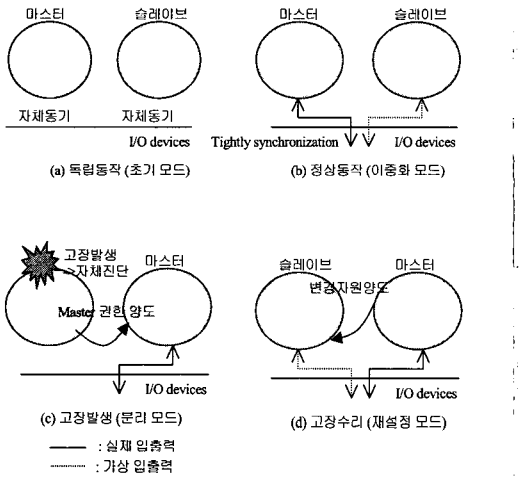

그림 13. 고장 감내 이중화 시스템의 동작모드

(d) 모드에서는 서비스를 일시 정지하고 수행해야 하므로 구현 방식에 따라 시스템의 성능 차이가 발생한다. 그리고 만약 (d) 모드에서 소모하는 시간이 길다면 가용성이 낮은 시스템이 될 수밖에 없으며 결국은 고장 감내 시스템으로 적합하지 않게 된다.

본 논문에서는 그림 14와 같은 재설정 모드 방식으로 구현하였다. 재설정 모드에 들어가기 전에 마스터 모듈은 정상적인 운영체제에서 동작하고 슬레이브 모듈은 부트 롬 상태에서 동작한다.

메모리의 전 영역을 '읽기', '쓰기'를 함으로써 메모리 내용을 동일하게 하고 각종 레지스터 정보를 NVRAM을 이용하여 임시 저장 장치로 사용하여 컨트롤 채널을 통해 전송한다. 그리고 시스템 클록을 결정하고 두 모듈을 메모리만 제외하고 동시에 초기화하여 같은 클록을 사용하게 한다. 이 때 초기화된 두 모듈은 부트 롬 상태로 존재하며 레지스터 정보를 복원함으로써 정상적인 운영체제에서 마스터 모듈의 이전 서비스 일시 정지 부분부터 다시 동작

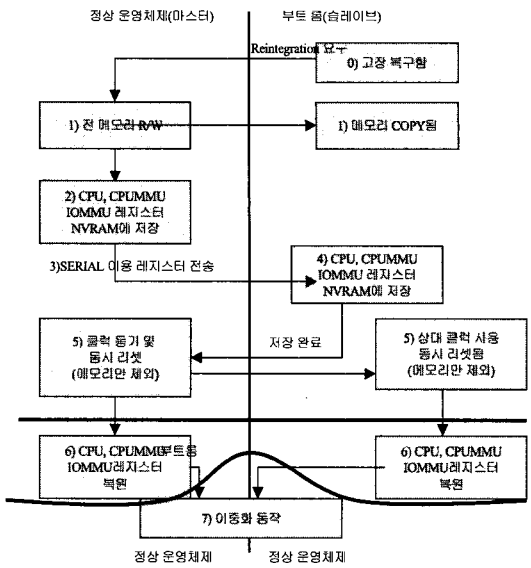

그림 14. 재설정 모드방식

한다. 이 방식은 백그라운드 작업으로 메모리 전 영역을 '읽기', '쓰기'를 수행함으로써 시스템 고유의 서비스를 수행할 수 있으므로 고가용성을 얻을 수 있고 Xbus 및 컨트롤 채널을 사용하여 마스터 모듈과 슬레이브 모듈간의 정보를 동일하게 하며 또한 클록 동기를 통하여 밀결합 동기 상태를 유지 할 수 있게 한다.

Ⅳ. 고장 감내 이중화 시스템의 성능평가

핫 스탠바이 스페어링 기법의 고장 감내 이중화 시스템의 성능 평가를 위하여 마코브 프로세스를 이용하여 모델링 하였고 CARMS(Computer Aided Rate Modeling and Simulation) 툴을 사용하여 시스템 정상동작에 관한 모의 실험을 하였다 [8].

4.1 고장 감내 이중화 시스템 모델링

본 논문에서 제안한 고장 감내 이중화 시스템에 존재할 수 있는 상태들은 총 7개로 그림 15와 같은 상태도를 갖으며 각 상태에 대한 설명은 다음과 같다.

P1 : 이중화 모듈 정상동작 상태
P2 : 이중화 모듈 중 한 보드 오류 발생 상태
P3 : 입출력 모듈 오류 발생
P4 : 단일 모듈 정상동작 상태
P5 : 정상 동작 입출력 모듈로의 대체 상태

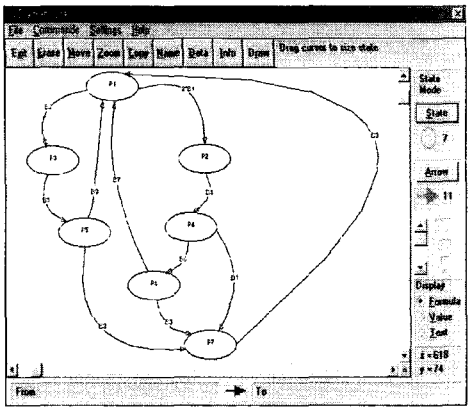

그림 15. 고장 감내 이중화 시스템의 상태 천이도

P6 : 시스템 복구 최종 단계
P7 : 시스템 완전 다운 상태

각 상태에서 다른 상태로 천이될 확률은 다음과 같다.

B1 : 마스터 또는 슬레이브 모듈의 고장율 (*Failure rate*)
B2 : 입출력 모듈의 고장율(*Failure rate*)
B3 : 복구 최종 단계에서 고장 발생할 확률 (*Failure rate*)
B4 : 마스터 또는 슬레이브 모듈이 차폐되고 단일 모듈로 동작할 확률(*Failure rate*)
B5 : 고장난 입출력 모듈을 차폐 시키고 보조 입출력 모듈로 대체할 확률(*Repair rate*)
B6 : 고장 모듈이 복구 시작해서 복구가 최종적으로 끝날 확률(*Repair rate*)
B7 : 메모리 복사 및 시스템 재동기가 성공적으로 완료된 확률(*Repair rate*)
B8 : 이중화 모듈에 대한 복구 및 초기화 완료되어 정상 상태가 될 확률(*Repair rate*)
B9 : 대체된 보조 입출력 모듈로 정상적으로 동작할 확률(*Repair rate*)

4.2 고장 감내 이중화 시스템 모델링 실험 결과

모의 실험은 천이 계수가 많으므로 실험을 간소화하기 위해서 B4~B9(Repair rate)를 동일한 일정한 상수(0.999)로 설정하고 B3도 일정한 상수(10-3)로 설정한다. 단 B1과 B2는 동일하다고 가정하고 1 10^{-4}부터 10^{-7}까지 값을 변화시키면서 가동률을 계산한다. 그 결과 표 1과 같은 B1, B2 값과 시스템 사용 연도에 따라 제안된 시스템이 P1 즉, 이중화 모듈 정상동작 상태에 존재할 확률을 얻을 수 있다.

표 1. 고장 감내 이중화 시스템의 고장률(*B1=B2*)에 따른 가동률

YEARS / B1=B2 Failure Rate	10YEARS	100YEARS
10^{-4}	0.999200574	0.999199859
10^{-5}	0.999919998	0.999919926
10^{-6}	0.999991999	0.999991992
10^{-7}	0.999999199	0.999999199

단일 모듈인 경우는 시간이 경과함에 따라 가동률이 '0' 에 수렴함을 알 수 있으며 제안된 시스템은 시간이 경과한 뒤에도 높은 가동률을 가지고 일정 시간이 지난 뒤에는 거의 일정한 가동률을 유지함을 그림 16의 실험결과로 부터 확인할 수 있다.

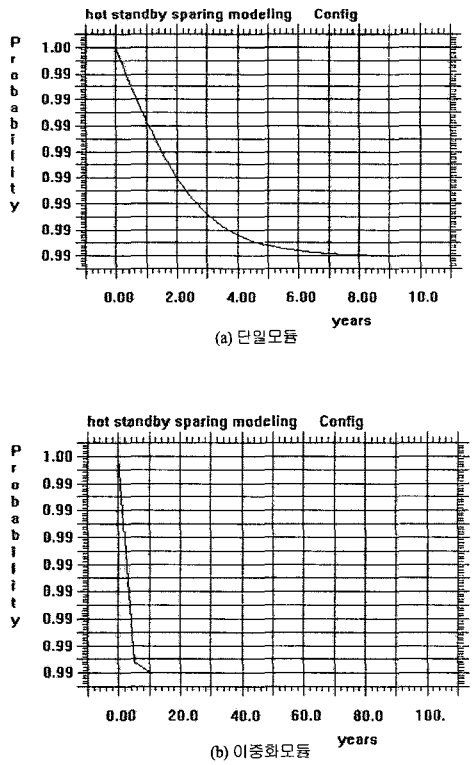

그림 16. 고장감내 이중화 시스템 가동율 비교

V. 결 론

통신망 이용의 활성화와 다양한 실시간 멀티미디어 서비스의 등장에 따라 통신망 장치의 가용성 및 신뢰성은 장치의 고속성 못지 않게 중요한 조건으로 부각되고 있다. 이에 따라 최근 개발되고 있는 통신망 장치들은 다중화 구조를 기반으로 하는 고장 감내형 시스템 구조를 채택하여 오류 발생시에도 극히 짧은 서비스의 중단을 보장하면서 정상 동작을 수행할 수 있도록 하고 있다. 따라서 전자정보 시스템들이 고가용성 및 고신뢰성을 얻기 위해서는 시스템의 안정화와 더불어 고장 감내 기법을 도입해야 함은 명백하다.

본 논문에서는 제어 시스템의 고가용성 및 고신뢰성을 확보하기 위한 방안으로서 핫 스탠바이 스페어링 기법의 고장 감내 다중화 시스템을 제안하였으며 단일 프로세서 모듈을 이중화하여 이중화 모듈간 비교 작업을 이용하여 고장을 검출할 수 있도록 하였다.

일반적인 범용 시스템에서도 고장 검출 버스를 사용 할 수 있도록 버스 변환 기능을 구현한다면 핫 스탠바이 스페어링 기법의 고장 감내 기법을 적용할 수 있다. 고장 감내 기능이 없는 일반 단일 프로세서 모듈은 시간이 경과될수록 가용성이 '0' 에 수렴하며 본 논문에서 제안한 시스템은 시간이 경과하여도 높은 가용성을 그대로 유지함을 알 수 있었다.

핫 스탠바이 스페어링 기법은 시스템 모듈간 클럭 동기 및 재 통합시에 메모리의 내용 등 모든 자원을 일치시키는 어려운점이 있으나 시스템의 고가용성 및 고신뢰성을 얻을 수 있는 좋은 방법이며 시스템 자원을 동일하게 유지시키기 위한 하드웨어 및 드라이버에 대한 연구가 필요하다.

참 고 문 헌

[1] Dhiraj K. Pradhan, Fault-Tolerant Computer System Design, *PRENTICE HALL*

[2] 정우석, 권보섭, 송광석, "동시 쓰기 방식을 이용한 결함 허용 제어 시스템", *JCCI97*, 1997

[3] Ft-SPARC manual, *IMP Ltd.*

[4] Emrys John Williams, Davic Charles Liddell, Fault-tolerant computer system, UK Patent

[5] Texas Instruments, *SuperSPARC User's Guide.*

[6] 정우석, 송광석, 이광선, 신진욱, 박동선, "고장 감내 시스템을 위한 버스 변환장치 설계", *추계통신학회*, 1998

[7] 박혜숙, 송광석, 이상백, 신진욱, 박동선, "고장 감내형 교환제어 시스템을 위한 관리 시스템 설계", *추계통신학회*, 1999

[8] Jan Pukite외 1명, Modeling for Reliability Analysis.

신 진 욱 (Jin-Wook Shin) 정회원

1993년 2월 : 전북대학교 정보통신공학과 졸업

1995년 2월 : 전북대학교 정보통신공학과 석사

1998년 3월~현재 : 전북대학교 전자공학과 박사과정

<관심분야> 영상 처리, 패턴인식, 디지털 시스템 설계

박 동 선 (Dong-Sun Park) 정회원

1979년 2월 : 고려대학교 전기전자공학과 졸업

1984년 : Missouri-Columbia 공학석사

1991년 : Missouri-Columbia 공학박사

1991년 3월 ~ 현재 : 전북대학교 전자정보공학부 교수

<관심분야> 신경망, 패턴인식, 영상처리, 디지털 시스템 설계