□특 집□

소프트웨어 결함 허용 측면에서의 결함 허용 실시간 시스템에 관한 고찰

한국과학기술원 전산학과 이홍규* · 이귀영**

목 차

Ⅰ. 서 론

실시간 시스템(real-time system)은 시스템내의 태스크(task)들이 각자에 부여된 시간 제한(time constraints)을 충족시키며 수행되는 시스템을 의미한다. 이러한 시간 제한 특성은 연산결과(computation result)의 정확성과 태스크의 수행시간 준수가 함께 요구되는 물리적 처리 환경(physical process environment)에서의 내장 컴퓨터 시스템(embedded computer system)이 갖추어야 할 필수적인 조건이다[1,2].

실시간 시스템은 일반적으로 자신보다 큰규모의 시스템을 제어대상으로 하고 있다. 제어대상 시스템(controlled system)의 예로는 핵발전 시스템이나 전화교환 시스템, 공정제어 시스템, 군사 시스템, 우주항공 시스템과 같은 것이 있다[2]. 이러한 실시간 시스템 중, 전화교환 시스템과 같이 태스크 수행에 대한 시간 제한이 태스크 수행 결과의 유효성에 결정적인 영향을 주지않은 시스템을 소프트 실시간 시스템(soft real-time system)이라 한다. 이와는 달리, 비행체 제어, 우주항공 시스템, 공정제어 시스템과 같이 태스크에 주어진 시간 제한이 엄격히 준수될 것이 요구되는 시스템을 하드 실시간 시스템(hard real-time system)이라 한다. 실시간 시스템은 시스템의 오동작이 심각한 결과를 초래할 수 있다. 특히 하드 실시간 시스템은 정확한 시스템의 동작이 요구되어, 결함허용(fault tolerance) 구조의 필요성이 절대적이다.

결함허용기법(fault tolerant technique)은 하

* 종신회원
** 정회원

드웨어와 소프트웨어의 신뢰도(reliability)를 향상시키는 기법의 하나로서, 하드웨어와 소프트웨어의 제작 시에 함유되어 있을 수 있는 결함 요인과 실행 중 외부적인 요인에 의해 발생되는 결함에 의해 발생되는 오류(error)로 인한 손상을 방지하는 데에 목적을 두고 있다. 그러나 소프트웨어는 주로 제작 시에 프로그램에 함유될 수 있는 오류를 방지하는 것에 주안하고 있고, 하드웨어는 실행시간(run time)에 발생될 수 있는 외부적 요인에 의한 오류방지가 대부분이다[3-5].

근래에 와서 실시간 시스템의 응용분야가 점차 확산됨에 따라 실시간 소프트웨어의 크기도 함께 커지고 있다. 이에 따라 실시간 소프트웨어에 함유되어 있을 버그의 절대수도 함께 증가한다[6]. 이는 하드웨어의 결함과 함께 소프트웨어의 결함 역시 시스템의 신뢰성에 크게 영향을 주게 됨을 의미한다. 결국 실시간 시스템의 신뢰도를 보장하기 위해서는 시스템의 설계에 하드웨어의 결함허용과 함께 소프트웨어의 결함허용에 대한 고려도 있어야 한다.

본 논문에서는 소프트웨어의 결함허용기법을 잘 반영할 수 있는 결함허용 실시간 시스템의 구조를 살펴보도록 한다. 논문의 구성은 다음과 같다. 2장에서는 실시간 소프트웨어의 결함허용 기법에 대한 간략한 고찰과 이러한 결함허용기법을 수용하기 위해 요구되는 시스템 구조의 요구사항을 정리하고 3장에서는 이제까지 설계되어 있는 실시간 시스템을 이러한 요구사항에 촛점을 맞춰 소개한 후 4장에서 요약과 결론으로 끝을 맺는다.

II. 소프트웨어의 신뢰도 향상기법과 실시간 시스템에서의 요구사항

본 장에서는 소프트웨어의 결함 발생 원인에 대하여 알아보고, 이에 따른 신뢰성 향상기법에 관하여 살펴보도록 한다. 그리고 신뢰성 향상 기법이 적용될 수 있는 시스템의 요구사항을 정리하여 본다.

2.1 소프트웨어의 신뢰도 향상 기법

소프트웨어의 결함은 실시간 시스템의 고장(failure) 원인이 될 수 있다. 소프트웨어의 결함 원인은 첫째, 부적합한 명세(specification)에 있다. 대부분의 소프트웨어 결함이 부적합한 명세에 그 원인을 두고 있다[6]. 둘째, 소프트웨어 구성요소(component)의 설계 오류(design error)에 의한 결함이 있다.

소프트웨어의 신뢰성 향상기법에는 결함방지(fault prevention)와 결함허용(fault tolerance)이 있다[5]. 결함방지는 결함회피(fault avodence)와 결함제거(fault removal)로 구성된다. 결함회피란 시스템을 구성하는 동안 결함요소(fault element)가 시스템에 포함되는 것을 방지하는 것으로 다음과 같은 방법을 통해 구성되는 소프트웨어의 질(quality)을 향상시킬 수 있다.

- 정확한 요구명세의 작성
- 증명된 설계방법의 이용
- 모듈화 및 자료 추상화를 지원하는 프로그래밍언어의 사용

이러한 결함회피의 과정을 거친 이후에도 소프트웨어 구성요소 상의 결함이 전무함을 보장할 수는 없다. 결함회피 이후의 과정인 결함제거는 설계과정에서 발생될 수 있는 설계오류를 발견하고 이를 제거하는 과정이다. 결함제거 과정에서 사용되는 기법으로는 설계검토(design review), 프로그램 검증(program verification), 프로그램 검사(program inspection), 프로그램 테스트(program test) 등이 있다. 이러한 기법의 사용에도 불구하고 시스템에 잔존하는 잠재적 결함(potential faults)을 모두 제거할 수는 없다.

결함허용은 이러한 잔존결함에 의하여 시스템이 동작 중 오류(error)가 발생되는 것을 막는 기법으로 응용프로그램의 특성에 따라 다음과 같이 계층 구분을 할 수 있다.

- **완전결함허용(Full fault tolerance)** : 시스템이 오류 발생 이후에도 심각한 기능적 손실이나 성능상의 손상이 없이 일정 시간동안 지속적으로 동작할 수 있는 정도.
- **단계적 성능저하(Graceful degradation)** : 시스템이 결함 발생 이후 수용가능한 정도의 기능적, 성능적 손실을 허용하여 계속 운영이 가

능하도록 하는 정도

• **고장안전(Fail safe)** : 고장발생 이후 시스템의 동작은 정지하나 무결성은 유지하는 정도.

완전결함허용은 실질적으로 구현하기가 대단히 어려우며, 고장안전은 실시간 시스템의 특성에 부합되지 않는다. 일반적인 결함허용 시스템에서는 단계적 성능 저하 단계의 결함허용 기능이 제공된다.

결함허용은 다음의 4단계로 구성되어 있다[5].

• **오류검출(Error detection)** : 결함에 의하여 발생되는 오류를 감지하는 단계

• **손상제한(Damage confinement and assessment)** : 소프트웨어에서 오류가 발생한 경우에는 시스템내에 오류를 발생시킨 결함의 영향 범위를 알아내는 단계

• **오류복구(Error recovery)** : 결함허용에서 가장 중요한 단계로 오류로 손상된 시스템을 무결한 상태로 만들어 지속적인 동작을 할 수 있도록 하는 단계

• **결함관리 및 서비스재개(Fault treatment and continued service)** : 시스템의 동작을 재개하는 단계

소프트웨어 결함허용기법에서 이와 같은 4단계가 모두 제공되고 있지는 않다. 소프트웨어의 결함허용기법에서 가장 일반적인 것으로는 NVP (*N*-version programmin)와 RB(Recovery Block)이 있다. NVP는 위의 4단계중 오류 복구의 단계가 생략되어 있다. 소프트웨어 결함허용의 기본 개념은 설계 다양화(design diversity)이다. 즉 하나의 계산 결과를 도출하기 위하여 다른 과정을 통해 개발된 동일한 기능의 프로그램을 여러개 사용하는 방법으로 하드웨어 중복(hardware redundancy)에 비해 높은 개발비용 부담이 있다.

2.1.1 NVP(*N*-version programming)

NVP는 하드웨어의 NMR(*N*-modular redundancy)과 유사한 개념의 결함허용기법이다[5,8]. NVP의 구성은 (그림 1)과 같이 버전(version)이라 불리우는 독립적인 개발 과정을 통하여 제작된 동일한 기능의 프로그램들과 이를 동작시키는 구동기 프로세스(driver process)로 구성된다. 이때 버전의 갯수는 2개 이상으로 구성되어야 한다.

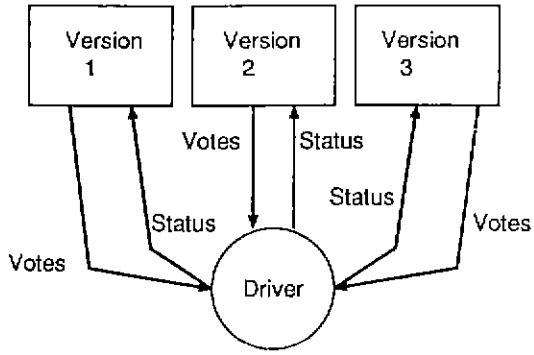
각 버전은 시스템의 응용 프로그램이고, 구동기 프로세스는 이들의 수행을 관리하는 역할을 한다. 구동기 프로세스의 역할은 다음과 같다.

• 버전을 실행(invoke)시킨다.
• 버전의 종료를 기다린다.
• 버전의 계산 결과를 비교하여 정확한 값을 결정한다.

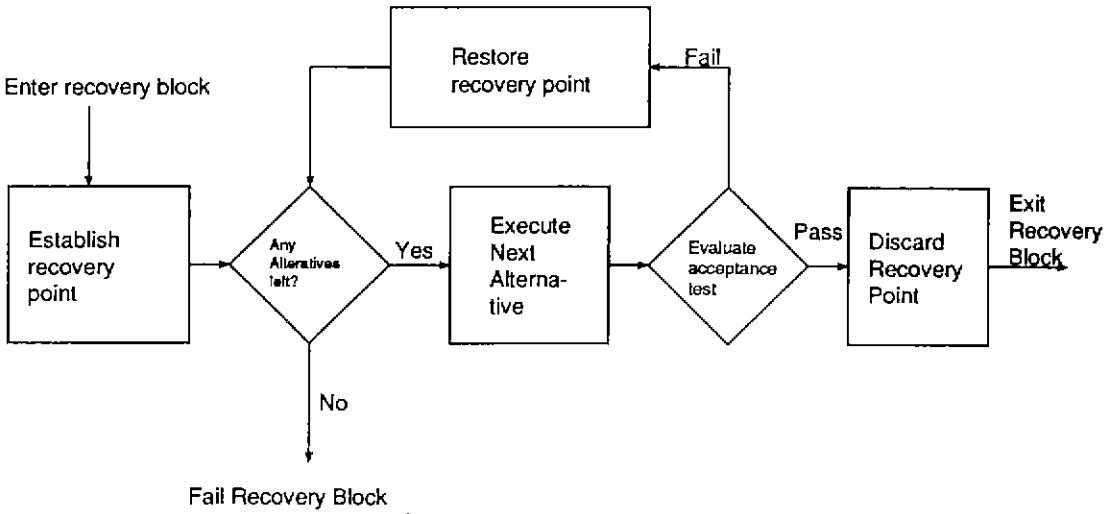
(그림 1)에서 보면 알 수 있듯이, 각 프로그램 버전은 다른 처리기의 프로그램 버전과 연관성 없이 독립적으로 수행한다. 연산 결과를 도출하게 되면, 구동기 프로세스에 결과를 전송하고, 구동기 프로세스로부터의 응답을 대기한다. 구동기 프로세스는 프로그램 버전으로부터 전송되는 연산 결과를 수집하고, 투표처리(voting)하여 올바른 값을 결정한다. 투표처리하여 결정된 값은 버전의 차후 동작에 관한 정보와 함께 구동기 프로세스로부터 버전으로 보내어 진다. 각 버전은 구동기 프로세스에서 보내진 동작 메세지에 따라 다음의 투표에 대한 참여 여부가 결정되게 된다.

2.1.2 RB(Recovery Block)

RB란 블록의 도입부에 복구포인트(recovery point)가 있고, 출구(exit) 부분에 인수검사(acceptance test) 부분이 있는 일반적인 프로그램 블



(그림 1) *N*-version programming



(그림 2) Recovery block mechanism

럭을 말한다[5]. 인수 검사란 블럭의 실행이후에 시스템의 상태가 올바른지 검사하는 것이다. 처음 실행되는 블럭을 주블럭(primary block)이라 하고, 주블럭의 실행이후 인수검사가 실패하였을 때에, 복구포인트로 후진복구(backward revovery)되고 나서, 실행되는 블럭을 대체블럭(alternate block)이라 한다. 실행 메카니즘은 (그림 2)와 같다. 소프트웨어 결함허용의 4단계 측면에서 보면, 오류 검출은 인수검사에서 행해지고, 손상 재한은 후진복구가 사용되는 관계로 필요하지 않다. 결함관리는 대기교환(stand-by spare)을 사용하여 이루어진다. RB의 구문는 프로그램 1과 같다.

Program 1. Recovery Block syntax

```
ensure ( acceptance test )
by
   (primary module)
else by
   (alternative module)
else by
   (alternative module)

          .
          .
          .
 else by
    (alternative module)
 else error
```

2.1.3 NSCP (*N* Self-Checking Programming)

NSCP는 하드웨어 결함허용의 동적중복(dynamic redundancy)에 해당하는 것으로, 실행중에 자신의 동적동작(dynamic behavior)을 검사하는 자기 검사 프로그램(self-checking program)을 이용하여 결함허용 구조를 만든 것이다[3,7]. NSCP는 실질적인 연산을 수행하는 실행요소(acting component)와 동일한 동작을 병렬수행(parallel execution) 하지만 실제로 결과가 시스템에는 전달되지 않은 상태인 핫 예비 프로그램(hot spare)로 구성된다. NSCP에서 결함허용은 실행요소에서 오류가 발생된 것이 발견되면, 예비 프로그램이 실행요소의 임무를 대행함으로서 이루어 진다. NSCP는 RB의 병렬수행 형태라 할 수 있다.

2.2 소프트웨어 결함허용을 위한 실시간 시스템 요구사항

앞에서 알아본 바와 같이 소프트웨어의 신뢰성을 높이기위한 기법 중 실시간 시스템의 설계

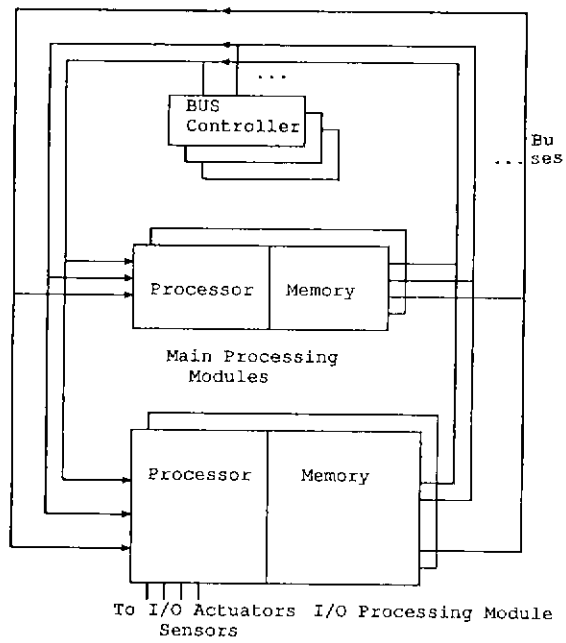
에 반영될 필요가 있는 것은 결함허용기법이다. 결함허용은 소프트웨어가 동작하는 도중 설계시에 발생된 결함요소로 인하여 오류가 발생되는 것을 방지하는 것으로, 결함방지가 실제로 소프트웨어가 사용되기 이전에 행해진다는 점에서 차이가 있다[3].

소프트웨어의 결함허용기법은 물리적중복(physical redundancy)을 이용하는 NVP와 NSCP, 시간적중복(temporal redundancy)을 이용하는 RB가 있었다. 각 기법에서 나타나는 요구사항은 다음과 같이 정리될 수 있다.

- 소프트웨어 결함허용기법은 결함없는 물리적 자원의 사용을 사용할 때 높은 신뢰성을 보장한다. 특히 처리기의 무결함이 보장되어야 한다. 그러므로 시스템의 설계는 결함없는 처리기를 제공할 수 있도록 구성되어야 한다.
- NVP과 NSCP의 반영을 위해 다수개의 처리모듈이 요구된다. 이때 소프트웨어의 설계다양화뿐만 아니라 하드웨어의 설계다양화를 위해서 서로 상이한 프로세서를 제공할 수 있어야 한다.
- NVP에서는 모든 버젼들이 드라이브 프로세스와 동기되어 동작하며 계산 결과를 비교하고 또 자신의 데이타를 수정하기도 한다. 시스템의 처리기들은 적절한 동기 방법이 필요하다.
- 각 처리기에 할당된 태스크를 일관성있게 스케쥴하는 스케쥴러가 필요하다. NVP나 NSCP는 동일기능의 태스크가 유사한 시기에 수행될 필요가 있다.
- RB를 사용할 경우 대체블럭의 수행이 시간제한을 준수할 수 있도록 하는 스케쥴러의 지원이 필요하다.

III. 실시간 시스템의 예

실시간 시스템은 일반적으로 제어대상 시스템에 내장되어 대상 시스템을 제어하게 된다. 여기서 소개되는 시스템들은 하드 실시간 시스템에 결함허용 구조를 갖추어 비행체 제어나 우주항공 시스템과 같은 응용분야에 사용할 목적으로 설계된 시스템이다. 본 장에서는 이러한 시스템의 간략한 구조와 구성기법을 살펴보도록 한다.



(그림 3) SIFT의 구조

3.1 SIFT(Software Implemented Fault Tolerance)

SIFT는 항공기제어와 같은 고신뢰성이 요구되는 응용분야를 위한 실시간 시스템의 개발을 위해 설계된 시스템이다[9]. SIFT는 10^{-9} *failurres/hour*의 고장비(failure rate)를 10시간 동안 유지할 수 있도록 설계되어 있다. SIFT의 구조는 (그림 3)과 같이 주전산기와 전용메모리로 구성되는 주처리모듈과 센서와 작동기에 연결된 입출력처리기와 이에 대한 전용메모리로 구성된 입출력처리모듈, 그리고 이들을 연결하는 다중버시스템으로 구성되어 있다. 주처리모듈은 실제로 제어동작을 처리하는 응용태스크들이 수행되며, 입출력처리모듈에서는 센서(sensor)의 감지와 작동기(actuator)를 제어하는 태스크들이 수행된다.

SIFT는 이름이 의미하는 바와 같이 에러발견과 수정 및 시스템의 재구성 등의 결함허용 기능을 하드웨어를 통해서 제공하지 않고, 소프트웨어를 통해 제공하고 있다. SIFT의 소프트웨어는 응용소프트웨어(application software)와 실행소프트웨어(executive software)로 구성된다. 응

용소프트웨어는 실질적인 제어동작을 관리하는 소프트웨어이고, 실행소프트웨어는 응용소프트웨어의 신뢰성 향상에 관한 작업, 즉 오류발견이나 시스템 재구성 등을 담당하는 태스크로 전역 태스크, 지역태스크 그리고 전역-지역 통신 태스크로 구분된다.

SIFT에서는 시스템의 하드웨어 구조를 이용하여 동일한 태스크를 다수의 처리모듈에서 수행시키고 계산결과를 투표를 통하여 이용함으로써 신뢰성을 높이고 있다. 태스크를 수행하는 처리기의 수는 태스크이 특성과 수행시점에 따라 결정된다.

결함고립화(fault isolation)는 버스와 처리모듈 사이의 경계에서 이루어진다. 손상된 데이타에 의한 오류의 확산을 방지하기 위하여 전용메모리에 대한 기록만이 허용된다. 또한 버스의 고장에 의한 데이타의 손상을 방지하기 위하여 버스를 통하여 데이타를 전송하고, 이를 투표에 의하여 선택하여 이용한다. 결함 발견시 버스에 대하여서는 해당버스의 사용을 중지하고 처리모듈에 결함이 있을때는 그 모듈에 할당된 태스크를 다른 모듈에 재할당한다.

SIFT 이전의 실시간 시스템은 클록(clock)의 동기를 위하여, 중간 클록(middle clock) 알고리즘을 사용하였다. 중간 클록 알고리즘은 주변의 클록과 자신을 비교하여 자신을 중간으로 조정하는 알고리즘이다. 그러나 중간 클록 알고리즘은 고장이 발생한 클록이 다음과 같은 고장형태를 보일 때 클록과 자신을 비교하여 자신을 중간으로 조정하는 알고리즘이다. 그러나 중간 클록 알고리즘은 고장이 발생한 클록이 다음과 같은 고장형태를 보일때 클록을 바르게 조정할 수 없다.

- 클록 A는 클록 B보다 약간 빠르게 동작
- 클록 C는 고장이 발생한 클록
- C는 A에 대해서 A보다 약간 빠른 값을 제공
- C는 B에 대해서 B보다 약간 빠른 값을 제공

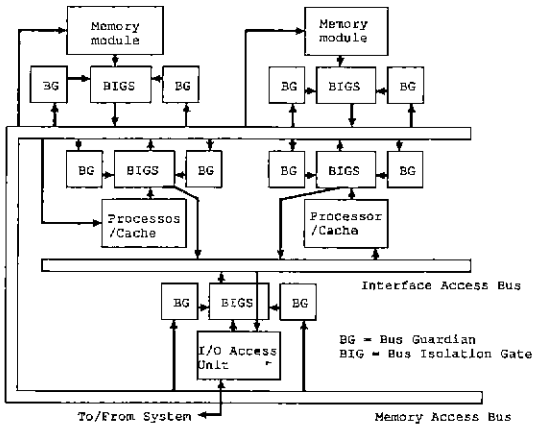
이러한 경우에는 A와 B는 재조정되지 않게 되고, 조금씩 차이가 나게 되어 일정 시간 이후에는 많은 차이를 보이게 된다. SIFT의 클록 동기화 알고리즘은 이러한 상황에서 N개의 클록에 대해 $N>3M$인 M개의 결함을 허용하고 있다. SIFT의 스케쥴링 알고리즘은 각 태스크의 수행주기가 정수비를 가지는 단순주기 스케쥴링 알고리즘의 변형으로 할당된 태스크가 처리기의 처리능력을 넘지 않는 경우 데드라인을 넘기지 않고 모든 태스크들이 수행될 수 있음을 보장한다. 그러나 SIFT는 결함허용을 소프트웨어를 이용하여 해결하기 때문에 80%의 연산 처리를 결함허용에 할당하는 오버헤드가 있다.

3.2 FTMP(Fault-Tolerant Multiprocessor)

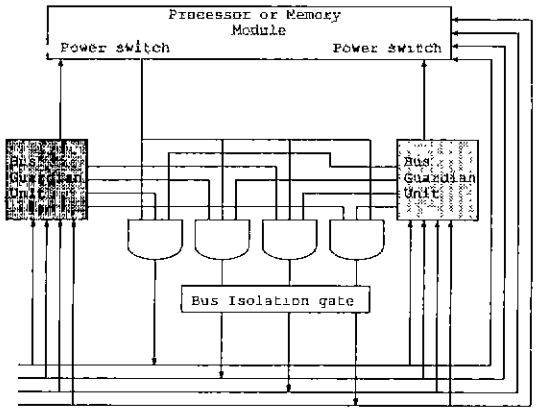
FTMP는 지연관리(delayed maintenance)나 주기적인 관리가 허용되는 환경에서 신뢰도가 높은 시스템을 구성하기 위해 설계된 시스템이다[10]. FTMP는 SIFT에서 소프트웨어에 의해 수행하는 투표나 시스템 재구성 등의 결함허용 기능을 하드웨어가 제공하도록 대체하고 있다. 각 FTMP는 10개의 동일한 처리기와 캐쉬(cache)의 모듈과 역시 10개의 동일한 메모리모듈로 구성되어 있다. 처리기모듈은 3개의 트라이어드(triad)를 구성하며 각 처리기트라이어드는 3개의 처리기모듈로 구성되어 있다. 마찬가지로 3개의 메모리모듈이 하나의 메모리트라이어드를 구성하게 된다. 이때 여분의 메모리모듈과 처리기모듈은 트라이어드내의 모듈이 고장나는 경우 이를 대체하게 된다. FTMP의 블록도는 (그림 4)와 같다. 시스템 하드웨어의 재구성은 버스가디안(bus guardian)을 이용한다. 버스가디안의 구조는 (그림 5)에 나타나 있다. 버스 가디안의 전력과 버스 입력 트라이어드는 다른 버스가디안과 독립되어 있어 고장 독립적이다.

FTMP의 클록 동기 방법은 4개의 클록을 사용하여 1개의 비잔틴 고장에 대하여 결함허용성을 가지도록 설계되어 있다. 동기 알고리즘은 다음과 같다.

- 각 클록은 다른 클록을 관찰하여 위상편차를 측정한다.
- 측정된 위상 편차를 정렬(sorting)한다.
- 중간 크기의 위상편차를 갖는 클록을 결정한다.
- 이 클록을 참조하여 자신의 클록을 조정한다.



(그림 4) FTMP의 구조



(그림 5) Bus guardian의 구조

트라이어드 내의 처리기들은 이러한 고장허용 클록을 이용하여 상호간에 긴밀하게 동기되며 동일한 소프트웨어를 수행하게 된다. 이렇게 트라이어드는 논리적으로 하나의 처리기 역활을 하게 되며, FTMP내에서 한 싯점에 수행될 수 있는 프로세스는 최대한 3개이다. 사용자 관점에서 FTMP는 공유메모리와 단일 버스로 연결된 동일한 3개의 프로세서로 구성된 시스템이다. 프로세서간의 통신은 공유메모리나 프로세서간의 인터럽트에 이용되는 특수 버퍼를 이용하여 이루어지며, 트라이어드에서의 데이타 일치는 투표된 결과를 메모리에 기록하고 기록된 데이타를 트라이어드 내의 멤버들이 읽어들이는 방식을 취하고 있다.

FTMP의 스케쥴러는 디스펫쳐(dispatcher)라 한다. 태스크는 독립적으로 수행됨을 원칙으로 하고 있어 우선순위 제한(precedence constraints)을 갖는 태스크는 허용되지 않는다. 태스크 스케쥴러은 선점(preemptive), 우선순위 방식이다. 우선순위는 태스크의 수행주기에 따라, 짧은 주기의 태스크가 높은 우선 순위를 가진다.

3.3 FTP(Software Implemented Fault Tolerance)

AIPS(Advanced Information Processing System)의 FTP는 FTMP의 한 변형이라 할 수 있다. FTP는 논리적으로 결함허용 기능을 제공하는 단일 처리기로서, 3중 FTP(triplex-FTP)와 4중 FTP(quad-FTP)가 있다. 3중 FTP는 동일한 소프트웨어와 하드웨어로 구성되는 3개의 프로세싱 채널(processing channedl)로 구성되며, 4중 FTP는 4개의 프로세싱 채널로 구성된다.

각 채널은 인터스테이지 모듈(interstage module)을 두고 모든 데이타 입력이 이 모듈을 통과하게하고 여기서 동일화된 데이타를 각 채널에 방송하여 자료 일치(source congruency)를 도모한다.

3중 FTP는 에러출력을 차단(mask)하는 하드웨어 투표기를 이용하는 정적중복(static redundancy) 방식을 사용한다. 이 하드웨어 투표기는 비트단위의 일치를 도모하도록 구성되어 있다.

3중 FTP에서 4중 FTP로의 확장은 인터스테이지 하드웨어의 구성이 훨씬 복잡하게 되어 용이 하지 않다. 4중 FTP에서는 주기적으로 검사 프로그램을 수행시켜 결함있는 채널을 찾고 이를 다음의 투표과정에서 제외시키는 동적 중복 (dynamic redundancy) 기법이 사용되고 있다.

FTP의 잉여채널은 2내지 5 μs의 마이크로프레임(microframe) 단위로 동기된다. FTP에서는 채널간의 병행 처리(concurrent processing)는 허용되지 않고 있다.

3.4 MAFT(Multi-computer Architecture for Fault-Tolerance)

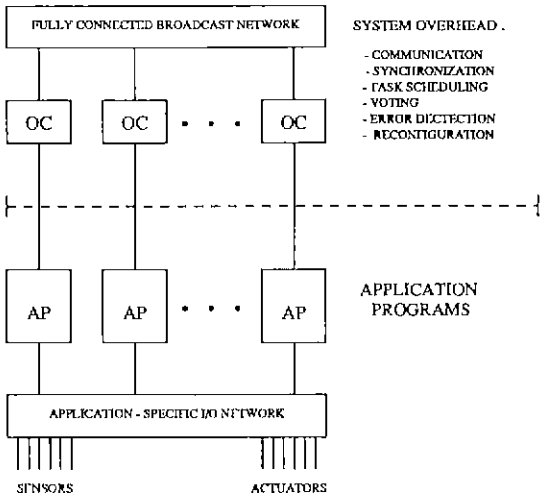
MAFT는 다양한 응용 분야를 지원할 수 있는 범용 결함허용 실시간 시스템의 개발을 목적으로 설계되었다[11]. MAFT는 결함허용의 오버헤드를 응용 프로그램으로부터 분리시키는 구조를 갖고 있다.

MAFT는 최대 8개의 노드를 가지는 분산 시스템이다. 각 노드는 OC(Operation Controller)와 AP(Application Processor)로 구성된다. OC는 주문 설계 VLSI 칩(customdesign VLSI chip)이고 AP는 응용에 적합한 범용 컴퓨터이다. OC는 동기화, 태스크 스케쥴링, 투표, 에러 감지, OC 간의 통신, 시스템 재구성과 같은 기능을 수행한다. AP는 OC에 의해 배당되는 응용 프로그램을 수행한다. 시스템의 AP는 같은 종류로 구성될 필요는 없다. MAFT는 AP의 구성에 상이한 종류의 컴퓨터가 사용되는 것을 허용함으로써 하드웨어와 소프트웨어의 설계다양화를 모두 지원한다.

MAFT는 느슨하게 동기(loosely synchronized)되는 분산 시스템이다. 노드들은 각자의 윈도우(window) 범위안에서 동기된다. 태스크 스케쥴링은 비선점, 정적 우선 순위 할당 방식이다. 태스크에 주어질 수 있는 중복노드의 갯수는 최소 1개로부터 시스템의 총 노즈 갯수까지 허용된다. 재구성 알고리즘은 단계적 성능 저하와 단계적 복귀(graceful restoration)을 제공한다. 비잔틴 동의 알고리즘과 근사 동의 알고리즘이 일치와 수렴을 위해 이용되고 있다. MAFT는 설계 결함 문제를 해결하기 위한 NVP형식의 소프트웨어의 실행을 지원하고 있다. 투표를 위하여 2개의 투표기법을 제공하고 있으며, FTMP와 같은 비트 단위 투표 기법은 사용되고 있지 않다. MAFT는 SIFT나 FTMP처럼 시스템의 하중이 큰 경우에 발생되는 시스템 기능에 의한 부담이 없다.

IV. 결 론

우리는 이제까지 소프트웨어 결함허용의 관점에서 기존의 결함허용성을 지닌 실시간 시스템 구조에 관하여 살펴보았다. 최초의 결함 허용 실시간 시스템 구조인 SIFT는 시스템의 결함허용을 소프트웨어로 해결함으로써 결함허용 오버헤드가 응용 프로그램의 수행에 심각한 영향을 주어 소프트웨어의 중복(redundancy)을 허용하는 경우 심각한 효율성 저하의 우려가 있다. FTMP는 이러한 결함허용 오버헤드의 문제를 투표기법이나, 동기 메카니즘을 하드웨어로 처리하여 해결했다. FTP와 FTMP는 중복 하드웨어를 통해 신뢰성있는 처리기를 제공하여 무결 소프트웨어가 오류를 발생시키는 일이 없도록 했다. MAFT는 소프트웨어와 하드웨어의 설계 다양화를 동시에 지원하는 구조를 갖고 있다.



(그림 6) MAFT의 구조

소프트웨어 설계과정에서 발생한 오류로 인한 소프트웨어의 잔존 결함은 실시간 시스템의 복잡도가 커짐에 따라 심각한 문제로 제기된다. 이제까지의 실시간 시스템은 신뢰도 향상을 하드웨어에 의존하여 왔다. 그러나 소프트웨어의 결함이 시스템의 신뢰도에 심각한 영향을 줄 수 있게된 현상황에서 소프트웨어의 결함허용은 간과될 수 없는 문제가 되었다. 이러한 사실들을 근거로 할때 결함허용 실시간 시스템의 설계에 소프트웨어의 결함허용 구조를 적절하게 수용할 수 있는 요소가 포함된다면 시스템의 신뢰도 향상에 크게 기여할 수 있을 것이다.

참 고 문 헌

1. J. A. Stankovic, "Misconceptions about

Real-Time Computing," *IEEE Computer*, Oct. 1988, pp. 10~19.
2. J. A. Stankovic and K. Ramamritham, Tutorical Hard Real-Time Systems, *Computer Society Press*, 1988, pp. 1~5.
3. J. C. Laprie *et. al.*, "Definition and Analysis of Hardware and Software-Fault-Tolerant Architectures," *IEEE Computer*, Jul. 1990, pp. 39~51.
4. T. J. Shimeall and N. G. Leveson, "An Empirical Comparison of Software Fault Tolerance and Fault Elimination," *IEEE Trans. on Software Engineering*, Feb. 1991, pp. 173~182.
5. A. Burns and A. Wellings, Real-Time Systems and their Programming Languages, *Addison-Wesley Press.*, 1989, pp. 91~24.
6. N. G. Leveson., "Software Safety: Why, What and How," *ACM Computing Surverys*, Vol. 18, No. 2, 1986, pp. 125~163.
7. S. S. Yau and R. C. Cheung, "Design of Self-Checking Software," *Proc. of Int'l Conf. Reliable Software*, 1975, pp. 450~457.
8. A. Avizienis, "The N-version Approach to Fault-Tolerant Systems," *IEEE Trans. on Software Engineering*, Vol. SE-11, No. 12, Dec. 1985, pp. 1491~1501.
9. J. H. Wensley *et. al*, "SIFT: Design and Analysis of a Fault-Tolerant Computer for Aircraft control," *Proceed. of the IEEE*, Vol. 66, No. 10, Oct. 1978, pp. 1240~1255.
10. A. L. Hopkins Jr. *et. al.*, "FTMP-A Highly Reliable Fault-Tolerant Multiprocessor for Aircraft,L *Proceed. of the IEEE*, Vol. 66, No. 10, Oct. 1987. pp. 1221~1239.
11. R. M. Kieckhafer *et. al.*, "The MAFT Architecture for Distributed Fault Tolerance," *IEEE Trans. on Computers*, Vol. 37, No. 4, Apr. 1988. pp. 398~405.
12. A. L. Liestman and R. H. Campbell, "A Fault-Tolerant Schduling Problem," *IEEE Trans. on Software Engineering*, Vol. 12, No. 11, Nov. 1986. pp. 1089~1095.

이 충 규

1978 서울대학교 전자공학과(공학사)
1981 한국과학원 전산학과 졸업(이학석사)
1984 한국과학기술원 전산학과 졸업(공학박사)
1986 미국 미시간 대학 Manufacturing Center연구원
1986 ~현재 한국과학기술원 조교수
1987 영국 Imperial College in London 교환교수
관심 분야 : 실시간 및 고장허용 시스템, 시스템 신뢰도 등

이 귀 영

1992 부산대학교 전자계산학과(이학사)
1992 ~현재 한국과학기술원 전산학과 석사과정
관심 분야 : 실시간 및 고장허용 시스템, 분산처리 등.