특집 - 컴퓨터

디지털 고장포용 시스템의 개념 및 특성

이 대 현*, 윤 재 영*, 김 학 배**
(*연세대 대학원 전기공학과 석사과정, **연세대 공대 전기공학과 교수)

1. 서 론

근래의 고속·고성능 컴퓨터의 활용분야는 복잡한 수식의 계산이나 모의실험(simulation) 등의 비실시간 응용에서부터 항공기, 인공위성, 핵발전소, 화학플랜트, 첨단의료기기, 전화망 스위치, 및 증권업무용 컴퓨터에 이르기까지 광범위해졌다. 이는 단순한 컴퓨터 H/W, S/W 및 관련 구현기술의 발전과 그 경제성 증대 등의 긍정적 동기와 함께 응용분야의 복잡성과 이에 따른 신속, 정확한 작업처리의 잠재적 기대 등으로 급격히 증가된 고성능 컴퓨터의 수요 요구에 기인한 것이다.

일반적으로 고속·고성능 컴퓨터는 다중프로세서의 병렬처리방식에 의해 특별한 architecture를 갖고 동기적(concurrent)인 연산을 수행하는데, 이러한 컴퓨터가 실시간 응용에 사용될 때, 특히 특수 목적의 제어작업을 수행할 때는 그 신뢰성이 매우 중요한 문제로 부각된다. 예를 들어 미국 국립항공우주연구소(NASA)에 의한 미래형 항공기(fly-by-wire)의 디지털제어기의 고장에 의한 시스템 전체의 crash 확률이 "10^{-10}/1시간" 이라는 점[1]만을 고려해볼 때도 고성능 컴퓨터의 신뢰성 보장에 대한 연구는 무척 중요하면서도 어려운 문제임을 알 수 있다. 또한, 디지털 시스템의 여러 가지 내·외적인 원인으로 인해서 발생되는 결함(fault)은[1] 기존의 아날로그 시스템보다 민감한 디지털 시스템에서 쉽게 오류(error) 및 고장(failure)을 일으킬 수 있고, 이것은 종종 전체 시스템에 치명적인 영향을 일으켜서 되돌이킬 수 없는 피해를 야기하거나 시스템 재사용에 심각한 문제를 발생하게 한다.

이러한 문제점을 해결하기 위해서 초기에는 완벽한 설계 및 제조 과정을 거쳐 원천적으로 시스템 요소의 결함 발생을 제거하려는 고장회피(fault-avoidance)기법이 깊은 관심과 함께 고려되었으나 많은 경제적인 문제와 기술적인 한계로 인해서 실용화에 큰 걸림돌을 갖게 되었다. 이러한 여러 가지 문제를 극복하기 위해서, 시스템에서 예측될 수 있는 결함 및 고장의 효과를 고려하여 각 부분별로 적절히 설계해서 일시적 또는 영구적인 결함 또는 고장이 발생하더라도 적절한 고장지역에 대한 검출(isolation) 및 분리(elimination), 그리고 신속한 회복(recovery)을 통해서 고장에 의한 영향을 최소화하면서 시스템에 할당된 작업을 정상적으로 수행할 수 있도록 고안된 것이 고장포용(fault-tolerant) 시스템이다[2].

본 논문은 고장포용 시스템에서 고려되는 결함, 오류 및 고장의 개념과 특성을 살펴보고, 고장에 대한 강인성을 부가하기 위한 공간 여분(spatial redundancy) 및 시간 여분(time redundancy)을 바탕으로 다양한 고장포용의 설계 기법을 설명하며, 또한 고장포용 시스템의 성능을 평가하기 위한 다양한 기준중 확률적 접근방식에 바탕을 둔 신뢰도(reliability)와 가용성(availability), 그리고 유지성(maintainability)에 대해 설명한다. 또한, 고장포용기법의 연구용 목적으로 개발되어 일부는 실제 활용되고 있는 대표적인 고장포용 시스템들인 FTMP[3], STAR[4], SIEF[5], C.vmp[6], 등에 대해서 간단히 살펴보도록 하겠다.

2. 고장포용 시스템의 기본 성질

고장포용이란 시스템의 일부 요소에 고장이 일어나더라도 전체 시스템이 잘못된 행동을 일으키기 전에 고장을 회복시키거나 고장의 효력을 제거하여 안전한 작업수행을 보장할 수 있게 하는 기법을 말한다. 초기에는 이러한 고장을 일으키는 문제를 해결하기 위해 고장회피(fault-avoidance) 시스템에 대한 고려가 제기되었으나, 이는 경제적인 문제와 제조상의 기술적인 한계 등으로 인해 실용화에 어려움이 있으며, 완벽하게 모든 종류의 고장을 피할 수 있는 시스템을 만드는 것이 현실적으로 불가능하였다. 그래서 디지털 컴퓨터의 급속한 발달과 관련 응용분야의 확대와 함께 고장포용분야에 대한 관심도가 매우 커지고 있는 현황이다.

1) 현장(field)데이터는 디지털 시스템의 H/W 요소들에 발생하는 90% 이상의 faults는 일시적인(transient) 것임을 통계적으로 단정한다[2].

고장포용의 이해와 응용은 과거 몇년동안 디지털 컴퓨터 설계의 필수적 요소로서 괄목할 만한 성장을 하였음에도 불구하고 여러 방면의 설계에 있어서 아직도 일반적이지 못한 것이 사실이다. 초기에 고장포용기법이 적용된 범위는 항공우주산업과 원자력 제어시스템 등과 같이 시스템의 고장 결과로 인명피해나 막대한 경제상의 손실이 야기하는 응용시스템에 제한되었으나, 일상생활에 있어서 컴퓨터에 의한 자동화된 기계의 역할이 점차 커짐에 따라 그 응용범위가 매우 다양해지고 있다. 예를 들어, 증권컴퓨터나 은행의 자동현금출납기 등의 일상생활에 필요한 디지털 컴퓨터의 오류는 많은 사람들의 필수활동에 심각하게 부정적인 영향을 끼치게 된다.

고장포용기법의 설계시의 기본개념은 시스템에 여분(redundancy)의 자원(resource)을 제공하는 것이다. 여분은 H/W, S/W 또는 시간으로 구성되며 모든 구성요소가 정상적으로 작동하는 동안에는 여분의 사용에 따른 어떤 성능상의 이득도 시스템에 주지 못한다. 여분을 사용하는 접근방식은 기본적으로 차의 트렁크에 예비 타이어를 가지고 다니는 것과 같은 맥락이다. 물론 고장포용기법에 사용되는 어떤 여분도 시스템의 설계 및 구현 비용을 증가시키므로 시스템은 많은 여분을 가질 수 없으며 (때로는 시스템의 여분이 너무 많은 경우 이의 관리 및 운용을 위한 기기의 구조가 너무 복잡해지거나 overhead 시간이 너무 커지는 결과 또한 무시 못함) 주어진 비용에서 시스템의 신뢰도를 극대화시킬 수 있는 고장포용기법을 사용해야만 한다. 그러기 위해서는 시스템의 고장의 유형과 특성에 대한 깊은 이해가 필요하다. 이러한 고장의 유형과 빈도를 특성화(또는 수식화)하기가 어려운 점 또한 적절한 고장포용기법의 개발에 장애가 되기도 한다. 즉, 영구적인 하드웨어의 결함은 잘 모델링 되어질 수 있지만, 일시적이고 간헐적인(intermittent) 결함을 정확히 모델링하기는 매우 어렵다는 점이다.

그러므로 새차가 예비 타이어는 가지고 다녀도, 여분의 배터리 등을 가지고 다니지 않는 것처럼 어떠한 시스템도 모든 가능한 결함 및 고장에 대해서 고장포용을 위한 여분을 가질 수는 없는 것이다. 그러므로 영향력과 빈도의 잠재력이 가장 큰 결함 및 고장을 target으로 집중적으로 관련 고장포용기법을 설계함으로써 시스템의 신뢰도를 높이는 효율적인 방법이 사용되어야 한다. 실제로 대부분의 고장포용 시스템은 여러 가지 결함 및 고장이 동시에(coincident) 일어나는 일이 드물기 때문에 (짧은시간의 작업수행중에는) 하나의 모듈 또는 요소의 고장만을 포용하는 단순한 설계방식이 주로 활용된다.

예를 들어, error-correcting codes는 메모리를 보호하기 위해 사용되며, 프로세서는 일시적으로 지속되는 결함으로부터 회복하기 위해 instruction retry를 사용하기도 한다. 또한 다중프로세서 시스템에서는 고장난 프로세서의 일을 다른 프로세서로 자동적으로 할당하기도 하며, 네트워크 시스템에서도 노드들 사이에 여러 가지 경로(path)를 만들어 하나의 고장이 일어나더라도 네트워크가 단절되지 않도록 하는 등 여러 가지 고장포용기법을 사용할 수 있으며 이외에도 많은 종류의 응용이 가능하다.

3. 고장의 정의와 분류

고장은 발생원인, 일시적 행동 및 결과지속시간 등의 여러 관점에서 다양하게 분류될 수 있다. 먼저, 가장 기본이 되는 결함(fault), 오류(error) 및 고장(failure)을 구분해 보면 다음과 같다.

- 결함(fault) : 요소(component)들의 defect, 환경에 의한 물리적인 방해, 작동자의 실수, 또는 부정확한 설계 등의 결과로 야기되는 하드웨어나 소프트웨어의 잘못된 상태를 말한다.
- 오류(error) : 결함으로부터 나타나는 현상으로 프로그램이나 데이터 구조에서의 틀린 function 값으로 작업에 대해서만 정의될 수 있음, 오류는 fault site와 어느 정도 거리가 떨어져서 발생할 수도 있다.
- 고장(failure) : 오류의 축적 또는 파급으로 인해 유도되는 요소 및 시스템, 또는 프로그램 및 작업단위의 잘못된 결과를 일컬으나 좁은 의미로 하드웨어의 물리적인 변화로 정의되기도 한다.

또한 이러한 fault, error 및 failure의 잠재하고 있는 시간을 의미하는 용어들은 다음과 같이 정의되고 이는 고장포용 시스템 설계에 대단히 중요한 변수가 된다.

- Fault latency : fault가 일어나고 error로 명시화될 때까지의 기간을 나타낸다. fault들은 그 자체로는 외부세계에서 나타나지 않으며 그들이 error를 일으켰을 때 알 수 있기 때문에 이러한 latency는 시스템의 전반적인 신뢰도에 영향을 미친다.
- Error latency : error가 생기고 그것이 error로서 인식되거나 시스템 failure를 야기시킬 때까지의 기간을 나타낸다. fault latency와 같이 error latency도 시스템 전반의 신뢰도에 중요한 역할을 한다.
- Fault-Tolerance Latency(FTL) : error로부터 회복하는데 필요한 일련의 단계들을 실행하는데 요구되는 시간으로 정의되며, 고장포용 컴퓨터나 안정성이 중요시되는 실시간제어 시스템을 설계하고 평가하는데 중요하다.

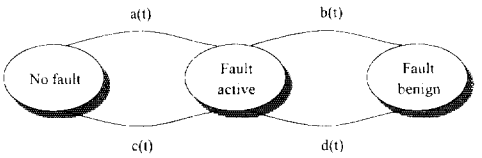
Fault들을 일시적인 행동(temporary behavior)과 출력 행동(output behavior)에 따라 분류하기도 한다. 이때 fault는 물리적으로 error를 야기시킬 때 active하다고 말하며, 그렇지 않을 때 benign하다고 말한다. 먼저, 일시적인 행동에 따라 fault들을 다음과 같은 세가지로 분류할 수 있다.

- Permanent fault : fault가 고쳐지거나 고장의 영향을 받은 부분이 교체되지 않는 한 시간이 지나도 사라지지 않고 지속되는 fault
- Intermittent fault : fault-active상태와 fault-benign 상태를 반복하는 fault
- Transient fault : 어느 정도 시간이 지나면 사라지는 fault

그림 1은 이러한 상황을 잘 나타내고 있다. 이때 a(t)와 b(t)는 fault의 상태가 변화하는 비율을 나타내며, t는 시스템 동작개시와 함께 시작되는 시간이다.

예를 들면, intermittent fault들은 회로상의 느슨한(loose) 접촉 때문에 야기될 수 있고, transient fault는 불안정한 환경의 영향 때문에 일어날 수 있다. Transient fault의 예는 적절하게 shielding 되어있지 못한 메모리에 ElectroMagnetic Interference(EMI)가 발생한다면, 메모리 칩에 구조적인 해를 입히지 않고서도 메모리의 내용이 바뀌며, 메모리가 다시 rewritten될 때 fault는 사라지게 된다. 대부분의 fault는 transient fault이고 단지 소수만이 permanent fault라는 사실이 많은 실험을 통하여 나타났다. 그러나 transient fault들은 종종 시스템이 error나 failure가 일어났다는 것을 알아차리기 전에 사라지기 때문에 발견하기가 어려운 점이 있다.

또한 fault는 fault가 야기시키는 error의 성질에 따라 특성 지어질 수도 있다. 이러한 출력 행동에 따른 구분은 malicious와 nonmalicious 두 가지로 할 수 있다. 예를 들어 unit B_1,...,B_n에 출력결과를 제공하는 unit A를 생각해보자. 만약 A가 nonmalicious fault를 가지고 있다면 그것이 야기하는 error들은 B_i의 모든 unit들에게 같은 방식으로 해석될 것이다. 예를 들어 A가 논리값 0에 stuck된 출력 line을 가지고 있다면 이 line으로부터 입력을 받는 모든 B_i의 unit들은 모두 이 line의 논리값을 '0'으로 인식할 것이다. 만약 B_i가 unit A로부터 받은 같은 물리적 신호를 서로 다른 값으로 해석한다면 이는 다루기가 훨씬 난해한 malicious fault로 정의 된다.



Fault type	Condition
Permanent	$a(t)>0$, $b(t)=c(t)=d(t)=0$
Transient	$a(t)>0$, $b(t)=0$, $c(t)>0$, $d(t)=0$
Intermittent	$a(t)>0$, $b(t)>0$, $c(t)=0$, $d(t)>0$

그림 1. fault 분류의 state diagram

대부분 fault들은 먼저 logical level에서는 이진수로 표현되는 신호(bit)들로 게이트들과 메모리요소들로 모델링 되어 설명될 수 있다. 다음은 logic-level fault 모델들을 간략히 나타낸 것이다. 이때 low-level 고장포용방법이 비정상적인 논리값을 만들어내지 못하게 fault들을 검출하고 회복하기 위해서 사용된다.

- Stuck-at 0 or 1 : 라인들과 게이트들, 핀들 등의 논리값들이 영구적 '1' 또는 '0' 의 값을 갖는다.
- Inverted : 원래 bit 값과 항상 반대의 값을 갖는다.
- Bridging : 두 개 이상의 인접해 있는 신호 라인들이 물리적으로 함께 short되어 있다. 몇몇 논리집합에서 이러한 것들을 "wired-AND" 나 "wired-OR" 함수로 추가적으로 소개하고 있다.
- Short or Open : open 또는 short 되어 있는 연결들과 관련된다.
- Unidirectional : 회로의 기하학적 성질 때문에 몇몇 faults들은 다중의 신호 라인에 영향을 끼칠 수 있다. 메모리-선택(memory-select) 라인의 open 회로는 하나의 word를 부정확하게 모두 '1'로 읽을 수 있다. 즉 error상에서 여러 가지 bit들은 모두 같은 논리적 방향을 갖는다. 다시 말해, 올바른 값으로 '0'을 가지고 있는 것들은 모두 부정확한 '1'의 값으로 변형된다.

레지스터, 산술논리 단위, 프로세서 등과 같이 더 높은 레벨에서 fault는 일반적으로 진리표와 상태를 나타내는 표 등의 모듈의 행동의 변화로써 나타난다. 그러므로 이러한 레벨에서의 fault 모델링은 일반적으로 행동 레벨에서의 시뮬레이션을 하는데 추상적이 될 수 있다. 그러므로 때로는 정확성이 감소되기도 한다. 그림 2는 여러 가지 에러의 가능한 원천들을 나타내고 있다.

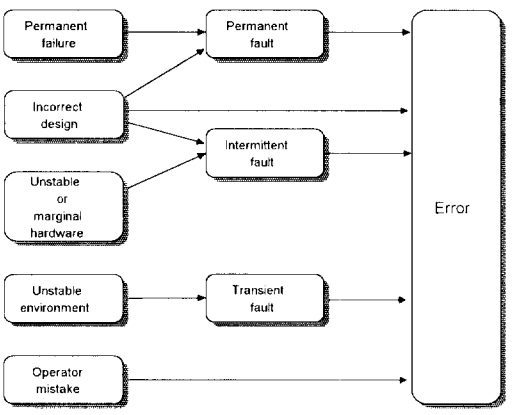


그림 2. 에러의 여러 가지 원인들

4. 고장검출 및 회복기법

고장포용 디지탈시스템에서, 앞에서 설명된 다양한 종류의 결함, 오류 및 고장이 발생했을 때, 이를 해결하기 위해서 검출(detection), 차폐(masking), 진단(diagnosis), 회복(recovery) 및 보상(repair/reconfiguration) 등의 과정을 거치는데 서로 독립적으로 구현되기도 하고, 검출과 진단 및 회복/보상이 서로 밀접하게 연관되어서 구현되기도 한다. 특히, 고장포용 시스템에서는 적절하게 여분(redundancy)을 사용하여 고장을 차폐 및 회복하게 되는데, 그 종류에 따라 (i)H/W 여분, (ii)S/W 여분, (iii)시간 여분, (iv)Information 여분 등이 있고, 구현방법에 따라서 정적(static), 유동적(dynamic), 그리고 혼합(hybrid) 방식이 있다. 또한, 사용되는 여분모듈(spare)도 작업대기 상태의 특징에 따라 hot, warm, 그리고 cool spare로 구분된다. 고장검출을 위해서는 duplication, coding 기법, check-sum, 그리고 온라인(on-line)으로 프로세서의 데이터 라인과 어드레스 라인을 계속 관찰해서 고장을 검출하는 watchdog timer를 이용한 time-out check 방법 등이 있다. 만약 고장검출이 적절하게 이루어지지 않은 경우에는 잠재된 에러가 파급되어 시스템의 여러 부분에 영향을 미칠 수 있는데 이를 방지하기 위해서 각 모듈의 입력과 출력단자에서의 논리적 값을 검사해서 고장 여부를 알아내어 모듈의 제거 및 bypassing을 통해서 고장의 영향이 전달되는 것을 막아내는 고장억제(fault containment/confinement)기법도 요구된다. 특히, 고장억제기법에서는 일정한 영역을 고장억제영역(fault containment zone)으로 설정하여 서로 독립적으로 모듈화시키는데, H/W 및 S/W 모두를 이용해서 구현 가능하다.

4.1 고장검출 및 회복기법

고장검출 및 회복기법(fault detection & correction or recovery)은 H/W적으로 구성되기도 하지만 가장 일반적인 형태는 coding theory를 기반으로 개발된 기법들이다. 온-라인 기법과는 다르게 오프-라인 고장검출은 고장진단을 하고 있을 때 작업을 수행할 수 없으므로 이에 대한 적절한 시스템 설계가 중요하다.

● Duplication : 가장 간단한 구조를 가지며 저렴한 비용뿐만 아니라 시스템 성능에 영향을 거의 주지 않기 때문에 가장 많이 사용되는 고장검출 기법으로서 동일한 작업을 각각 다른 프로세서 및 모듈에서 수행해서 그 결과를 서로 비교함으로써 고장발생 유무를 알아낼 수 있는 방법이다. 그림 3에서와 같은 swap-and-compare 기법은 중복된 고장이 동시에 모든 모듈에서 발생하지 않게 하기 위해서 모듈을 적절하게 분리해서 구현하는 것으로 C.mmp

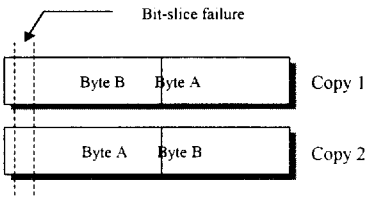


그림 3. Swap-and-compare check scheme in C.mmp

다중프로세서[6]에서 사용되었다. 그러나, 동일한 프로세서를 사용할 경우에 동일한 결점을 가질 수 있기 때문에 동시발생적(coincident) 고장이 발생할 가능성이 있으므로 고신뢰도를 보장할 수는 없다.

● Parity codes : 전체가 n-bit로 구성된 information code 가운데 하나의 bit 이상에서 동시발생이 가능한 고장을 k-bit라고 가정한다면, ${}_{(n+c)}C_{(k)}+1 \leq 2^c$의 식에 의해서 필요한 parity bit c를 설정한다. 즉, 만약 하나의 bit 만이 고장 가능하다면, $n+c+1 \leq 2^c$에 의해서 parity bit 설정이 가능하다.

● Checksum : 서로 데이터를 주고받을 때 전송단 쪽에서 모든 word를 더함으로써 checksum을 설정한 후 데이터와 함께 적절한 데이터 코딩을 통해서 전송하고, 또한 수신단 쪽에서는 전송된 데이터를 받아서 데이터와 checksum을 분리해서 전송된 데이터의 checksum을 전송단에서 받은 checksum값과 서로 비교함으로써 고장을 검출할 수 있다. Checksum 기법은 크게 세가지로 나눌 수 있는데, single-precision checksum과 double- precision checksum, 그리고 Honeywell check sum이 있다. N-bit word를 전송할 때 single과 double은 각각 word의 합을 구하는 과정에서 무시해도 좋은 비트의 범위를 말하는 것으로 각각 n-bit와 2n-bit이상은 무시할 수 있음을 말한다. 만약, 동일한 bit 위치에 일관되게 에러가 발생하는 경우 각각 두 개의 word를 연결시켜서 합을 구하는 Honeywell checksum을 이용해서 에러를 검출할 수 있다.

● Watchdog timers : 일정한 시간이내에 원하는 값이 어떤 특정 프로세서 및 모듈에 주어지지 않는 경우에 timeouts에 의한 고장으로 판단되는데, 이때 고장을 검출하기 위해서 어떤 프로세서의 데이터 및 어드레스 라인을 주기적 또는, 비주기적으로 watchdog timer를 통해서 체크해야 한다.

이 외에도 컴퓨터에서뿐만 아니라 통신시스템 및 일반적인 디지털 시스템에서 고장검출 및 회복을 위해 가장 적절하면서 많이 사용되는 정보여분 시스템으로서 주로 coding 기법을 응용한 것이 많은데 arithmetic codes, cyclic codes

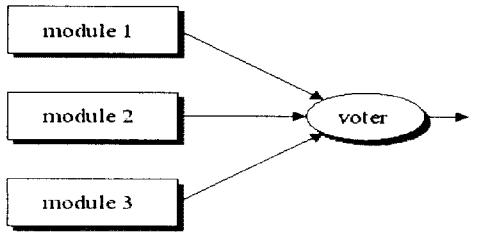
등이 있으며 이들은 주로 H/W 적으로 구현된다.

4.2 여분 시스템(redundant system)

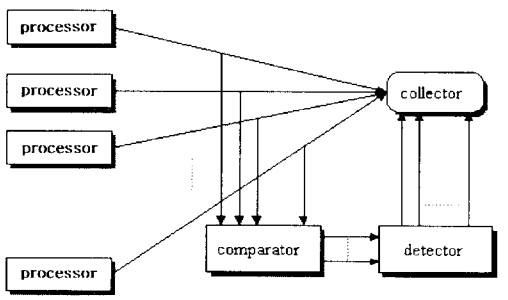
앞에서 언급된 대부분의 고장 검출 및 회복 방법들도 여분(주로 information redundancy) 활용을 기본으로 개발되었지만 앞으로는 보다 구체적인 H/W, S/W 및 시간 여분을 바탕으로 하는 여분 시스템에 관해 설명한다.

● H/W 여분 시스템 : H/W로 구성되는 여분시스템은 동일한 작업을 할 N개(≥3)의 프로세서나 모듈을 구성하여 고장이 발생하더라도 고장차폐를 통해서 고장과 관계없이 작업수행을 완결할 수 있도록 하거나 또는 여분모듈(spare module)를 추가설치해서 고장이 발생할 경우에 적절한 스위칭을 통해서 정상적인 작업수행을 도모한다. 전자의 경우 voting이 필수적인데 많이 사용되는 voting 기법으로 majority voting, k-plurality voting, 그리고 median voting 등의 방법을 사용하며, N-modular 여분(NMR)기법은 만약 m개의 고장을 적절히 차폐하기 위해서 필요한 최소한의 N개의 프로세서 및 모듈로서 구현해 놓은 것이다. 일반적으로 non-malicious 고장인 경우에는 (2m+1)개의 프로세서 및 모듈로서 구성되고 malicious 고장인 경우에는 (3m+1)개로 최소한 구성되어져야 한다. 가장 대표적이면서도 간단한 N-modular 여분 시스템으로 TMR(Triple Modular Redundancy)은 하나의 모듈이 고장을 일으키는 경우에 고장을 차폐할 수 있는 기능을 가지며[7, 8, 9], sift-out 여분은 고장이 발생한 모듈을 제거할 수 있는 기능을 가진다.

● S/W 여분 시스템 : S/W 여분을 응용한 시스템으로는 N-version programming 기법과 recovery-block approach 기법이 있다. 그러나, 아직까지 S/W 신뢰도를 보장하기 위해서 필요한 최소한의 version의 개수나 common-mode failure를 최소화하기 위한 절차, 그리고 어떻게 S/W가 테스트되어져야 하는가에 대해서 설계자의 입장에 알맞은 해답이 나오지 않은 상태이다. N-version programming 기법은 동일한 작업을 수행하기 위해 여러 개의 S/W를 구현해서 병렬로 작업을 수행한 후 그 출력값을 voting하는 것으로 NMR 기법과 유사하다. Recovery-block approach 기법은 적절한 acceptance test를 통해서 작업수행 결과의 타당성을 검토한 뒤 적절하지 않은 경우에 또 다른 version으로 작업을 수행하는 기법이다. 즉, 동일하게 여러 개의 version program이 동작하지는 않으므로 비용을 절감할 수 있다. 그러나, acceptance test에 대한 타당성이 우선 검증되어야 한다. 여러 개의 S/W version은 requirements specification과 programming language, numerical



(a) Triple modular 시스템 with single voter



(b) Sift-out 여분

그림 4. H/W 여분 시스템

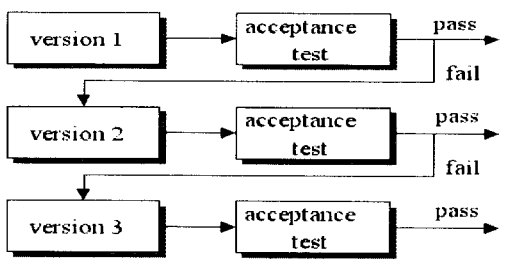


그림 5. S/W 여분 시스템(recovery-block approach)

algorithms, 그리고 사용되는 tool의 특성 및 프로그래머의 자질에 의해서 영향을 받는다.

● 시간 여분 시스템 : 앞에서 제시한 방법들이 전방에러회복(forward error recovery)방법이라면, 시간 여분 시스템은 후방에러회복(backward error recovery) 방법이다. 가장 간단한 방법으로는 하나의 instruction을 다시 수행하는 retry[7]와 회복기점(recovery point)으로 회귀하여 작업을 다시 수행하는 방법인 roll-back[2]과 모든 작업을 다시 수행하는 restarting [9]이 있다. 이 가운데서 roll-back은 회복기점을 어떻게 설정할 것인가를 시스템 설계에서 주로 고려해야 하는 대상인데, 데이터의 신뢰성을 높이기 위해서는 회복기점을 자주 설정해서 점검해야하지만,

회복기점이 많을수록 작업수행 완결시간이 길어지고 많은 메모리가 요구되어짐을 고려할 때 여러 자원(resource)간의 trade-off 문제를 발견할 수 있다. 그래서, 각각의 회복기점에서 변화된 값만을 적절히 저장해 진단시간 및 메모리의 요구를 크게 줄인 회복캐쉬(recovery cache)를 이용해서 어느 정도 단점을 보완하기도 한다. 후방에러 회복방법에서는 무엇보다도 고장이 발생했을 때 어느 정도 수행되어진 작업을 무시하고 원상복귀시켜야 하는데, 간단한 프린팅작업에서 미사일 발사작업에 이르기까지 복구되지 않는 작업을 수행하는 시스템인 경우에는 이러한 방법을 적용하기가 힘들 것이다.

5. 고장포용 시스템 평가 및 분석

시스템 설계자는 구현된 고장포용 시스템의 성능을 적절히 평가함에 의해서 미미한 부분을 수정, 보완함으로써 개선된 특성을 갖도록 변화시킬 수 있다. 시스템 성능평가의 기준은 시스템이 사용되는 목적에 따라서 다양하게 설정되지만, 일반적으로 많이 사용되는 성능평가의 기준은 신뢰도(reliability), 유지성(maintainability), 가용성(availability)등으로 표현된다. 이러한 평가지표는 분석적(analytic)인 방법 및 실험적(experimental) 방법에 의해서 비교, 분석 및 평가가 가능하지만, 실험적 방법은 일반적으로 오랜 기간동안의 실험결과를 필요로 하기 때문에 많은 비용과 시간이 소요되므로 대부분의 경우에 분석적인 방법을 이용해서 필요한 목적에 맞는 시스템을 구현한다. 일반적으로 시스템에서 발생하는 고장은 결정적(deterministic)인 것이 아니기 때문에 확률적인 모델을 사용해서 모델링 분석을 하는데, markov 모델, renewal 프로세서, poisson events 등의 stochastic 모델들을 기본으로 하여 수학적으로 계산한다. 먼저, 고장포용 시스템의 기본평가 인자들에 대해 알아본다. 다음에서 설명되는 모든 인자들의 정성적인 속성(attribute)을 포괄적으로 나타낸 종합적인 평가인자가 "의존도(dependability)"라고 정의되어 있으나 이의 정량화(quantification)가 응용분야에 따라 가변적이어서 쉽지 않은 문제로 남아 있다.

● 가용성(availability) : 가용성은 시스템이 시간 t=0에서 동작하기 시작했을 때 우리가 관심있는 시간 $t=t_1$에서도 시스템이 안정하게 동작하고 있을 확률을 말한다. MTTF(mean-time-to-failure)와 MTTR (mean-time-to-recovery)는 모두 시간의 함수로서[2] 이들을 이용한 시스템의 유용성은,

$$A(t)_{system} = MTTF / (MTTF + MTTR) \quad (1)$$

로 주어진다. 이 식에서 알 수 있듯이 MTTF가 MTTR에 비해서 상대적으로 매우 큰 경우에 고장이 발생하더라도 시스템의 회복이 매우 빠르므로 시스템의 가용성은 거의 1에 가까운 값을 갖게 된다. 그러나, MTTF가 MTTR에 대해 상대적으로 매우 큰 값이 아닐 경우에 MTTR은 시스템의 가용성에 큰 영향을 끼치게 된다. 즉, 고장포용 시스템에서는 고장이 일어나는 시간에 비해서 회복하는 시간이 매우 작도록 설계되어져야 하는데 이것을 stiffness 라고 한다.

● 신뢰도(reliability) : 시스템이 시간 t=0에서 동작한 후 $[0, t_1]$까지 시스템 failure가 발생하지 않고 주어진 작업을 계속 수행할 조건부 확률을 신뢰도라고 한다.

$$R(t)_{system} = Pr(no\ fault) + [Pr(non\text{-}failure|fault\ occurring) \times Pr(fault\ occurring)] \quad (2)$$

일반적으로 대부분의 시스템에서 신뢰도 모델링이 시스템의 구조적 복잡성으로 인해서 어렵기 때문에, 상태통합(state aggregation) 또는 상태재구성(decomposition)기법을 이용해서 시스템의 상태를 최소화시키고, 이들의 전이상태도 지수함수의 형태나 weibull 프로세서로 간략화 시켜서 해결한다. 만약, 실패율이 비율 λ로 poisson 프로세서에 의해 지배되고 지수함수의 상태전이를 갖는 경우에 수명(lifetime)분포는 $Fl(t)=1-\exp(-\lambda t)$이고, weibull 함수의 상태전이에는 $Fl(t)=1-\exp(-[\lambda t]^{\alpha})$로 표현할 수 있다. 또한, 이것이 시스템 failure가 발생하지 않을 확률분포이기 때문에 시스템의 신뢰도가 된다. 시간 t 이전까지 시스템의 failure가 발생하지 않은 상태에서 시간 t 에서 시스템 failure가 일어날 확률을 위험함수(hazard function)이라고 정의하면 위험함수 h(t)는,

$$h(t)dt = Pr(system\ failure\ in\ [t,t+dt]|system\ has\ not\ failed\ up\ to\ t) = \frac{f_l(t)\,dt}{1-F_l(t)} \simeq \lambda \quad (3)$$

로 얻을 수 있다.

일반화하면, $I(t)=\int_0^t h(x)dx$ 라고 정의할 때, 주어진 시간 [0,t]에서 시스템이 failure를 일으킬 확률은 $1-e^{-I(t)}$로 얻을 수 있다. 예를 들어서, 각각 지수함수분포를 갖는 모듈의 신뢰도가 R_m이고 voter의 신뢰도가 R_v의 단일 voter를 갖는 일반적인 TMR 시스템의 경우에 신뢰도는 $R = R_v \times (3R_m^2 - 2R_m^3)$과 같게 되는데, 이는 일정 작업수행시간(mission) 영역에서 단일 모듈을 사용하는 것과는 달리 더 좋은 신뢰도를 가지게 됨을 알 수 있다.

2) 이 두가지 평가 인자는 모두 deterministic 값을 갖는다는 장점을 내포하나, 평균값으로만은 모든 sample 값의 분포상황을 알 수 없다는 사실을 고려할 때, 빈도가 낮은(드문) 사건들의 경우도 중요시되는 상황(예, real-time 시스템)에서는 시스템의 고장포용능력을 평가하기에는 충분하지 못한 인자들이다.

● 유지성(maintainability) : 시스템의 고장을 방지하기 위해서는 적절한 유지, 보수 작업을 수행해야 한다. 이것을 위해서 유지성은 테스트 기법과 많이 관련되고, 이것은 단지 고장난 모듈을 적절히 고립화시킬 뿐만 아니라 회복작업의 타당성을 입증하는 것을 모두 유지성과 관련된 작업으로 간주한다. 그러므로, 유지성을 지닌 시스템을 설계하기 위해서 시스템 설계자는 (i)미수정된 회로에 대한 테스트, (ii)현재의 시스템에서 최소한의 수정, (iii)확장된 회로 수정, 및 (iv)새로운 설계와 같은 단계를 거치게 된다.

다음으로 특정한 고장포용 시스템에 대해 다양한 조건하에 이러한 평가인자 값을 분석적으로 유도할 수 있는 유용한 stochastic 모델들을 설명하고자 한다.

● Markov modeling : 기본적인 stochastic process $\{X_n, n=0,1,2,\cdots\}$가 제한된 수의 가능한 값을 가지는 경우에 시간 n에서 상태(state)변수가 i라면 $X_n=i$로 표현가능하고, 이 때 다음의 상태 j로 전이될 확률이 상태변수가 i에 있을 때마다 고정된 P_{ij}값을 가진다고 할 수 있다. 즉,

$$\Pr\{X_{n+1}=j \mid X_n=i, X_{n-1}, \cdots, X_1=i_1, X_0=i_0\}=P_{ij} \quad (4)$$

이때, 앞으로 전이되어질 상태변수 X_{n+1}은 이전의 모든 상태변수에 대해서 독립적이고, 단지 현재의 상태변수인 X_n에만 의존적이게 되는 markovian 특성을 갖는데, 전이를 일으키는 사건이 이산분포를 갖게될 경우 확률분포함수(probability density function, *pdf*)가 poisson 분포, 즉 지수함수형태의 분포를 가진다. 따라서, 상태변수 사이에서 전이가 일어날 때 *pdf*는 전역시간(global time)과는 무관하고 단지 현재에 사건이 머무르고 있는 상태에서의 지역시간(local time)에만 영향을 받게된다. 이와는 달리 사건 발생이 연속적인 경우에는 counting process로 볼 수 없기 때문에 더 이상 poisson 프로세스, 즉 지수함수 분포에 지배되지 않고, 전이율(transition rate)은 일반적인 함수의 *pdf* 형태를 갖게 된다. 이를 semi-markov model 이라고 하는데 이 경우에는 더 이상 markovian 특성을 가지지 않는다. 다음 상태변수로 전이되어질 확률을 알기 위해서는 단지 현재의 상태뿐만 아니라 현재 상태에서 머무르고 있는 시간(residence or holding time)에 대한 정보도 필요로 한다. 수학적인 markov model을 사용해서 신뢰도 모델링을 할 경우에 복잡한 시스템 구성을 상태통합 및 제거 등의 근사화(approxiamtion)방법에 의해 모델상태수 및 계산량이 대폭 감소될 수 있다. 예를 들어서 TMR 시스템의 markov 프로세서를 이용하면 그림 6과 같이 모델링 할 수 있다.

그림 6에서 상태 2는 TMR시스템의 모듈 중에서 하나가 고장을 일으킨 경우를 나타내고 있고, 상태 3은 세개의 모듈 중에서 두개 이상의 모듈이 고장을 일으켜서 전체적인 TMR시스템이 고장을 일으키는 상태를 나타내고 있다. 전이 1과 전이 3은 모듈 세개중에서 하나가 고장을 일으킬 전이율을 나타내고 전이 2는 회복률을 나타내고 있다. 이 시스템이 고장을 일으킬 확률은 상태 3에 있을 확률이며, 신뢰도는 1 − Pr[state 3]과 같다.

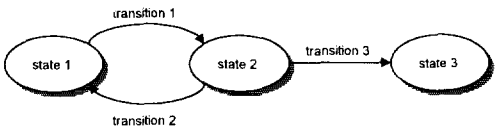


그림 6. TMR 시스템의 Markov 모델링

6. 실제 응용시스템

고장포용시스템으로서 개발된 응용시스템으로서는 이미 많은 수가 군사, 산업, 교육, 연구 등의 목적으로 활용되고 있지만, 본 논문에서는 이들중 가장 대표적인 FTMP, STAR, SIFT, 그리고 C.vmp 등을 간략하게 소개한다.

● Fault-Tolerant Multiprocessor (FTMP) [3] : 시스템 고장이 시간당 10^{-10} 정도로 요구되는 초고신뢰도의 fly-by-wire 프로젝트의 목표 하에 개발된 디지털 제어컴퓨터 구조로서 1979년에 첫번째 기본형태가 완성되었다. FTMP는 여분의 serial 버스를 통해서 서로 정보를 주고받는 독립적인 다중프로세서 캐쉬메모리 모듈과 일반적인 메모리 모듈을 기본으로 모든 정보의 처리와 전달에서 TMR을 수행함으로써 고장을 검출, 수정할 수 있게 설계되었다.

● Self-Testing and Repairing Computer (STAR) [4] : 유동적 여분을 사용하는 실험용 컴퓨터를 설계하려는 목적으로 제작되었다. 일시적인 고장을 막기 위해서 교환가능한 하부시스템과 rollback 기법을 사용하고 있으며, 1969년에 제작되었다.

● Software Implemented Fault Tolerance (SIFT) [5] : 프로세싱 모듈의 복제를 통해서 고장포용을 제공하는 고신뢰도 컴퓨터로서 항공기 제어장치에 사용하려는 목적으로 개발되었다. 고장이 발생할 경우에 고장모듈의 고립화를 통해 전체적인 시스템의 고장을 방지하기 위해서 프로세싱 모듈을 연결하는 버스를 위해 특별하게 설계된 여분의 버스시스템을 사용하였다. SIFT S/W는 SRI-developed SPECIAL 언어를 이용해서 체계적인 구조를 갖는다.

● A Voted Multiprocessor (C.vmp) [6] : C.vmp에 대한 개발은 전자기적 noise, 미숙한 사용자, 그리고 연속 동작과 같은 속성을 가지는 산업환경에

서 사용할 수 있는 고장포용 시스템에 대한 연구에서부터 시작하였는데, 주된 고장포용기법으로 버스 레벨의 voting을 사용하였다.

7. 결 론

고장포용은 시스템이 어떤 요소의 결함으로 인해 유도된 오류 또는 이의 축적으로 subsystem에 고장이 발생한다고 해도 전체시스템의 주어진 기능 또는 작업의 정상적인 수행을 보장할 수 있는 능력으로 정의되는데, 진동, 습기, 온도, 전자장, 먼지, 및 화학약품 등의 열악한 환경 하에서도 연속적이고 신뢰성 있는 작업수행을 해야하는 고신뢰도의 시스템에서는 고장포용 기법의 필요성이 더욱 부각된다. 앞에서 설명된 바와 같이 효과적인 이상검출(detection/diagnosis)을 통해 각 요소의 고장을 분리(isolation)하고 그 영향으로부터 신속하고 정확하게 회복(recovery)하는 접근방식은 응용시스템의 특성에 따라 다양하게 설계될 수 있고, 또한 이를 평가하는 인자 및 방법도 다양함을 살펴보았다. 이러한 고장포용 기법은 시스템의 성능 및 신뢰도에 시간이 더욱 치명적으로 영향을 미치는 실시간 시스템인 경우 그 중요성과 복잡성이 더욱 증가한다[1,2,7].

결론적으로 이러한 고장포용기법에 대한 연구는 고신뢰도, 고성능 컴퓨터의 설계 기법 제시와 평가 방향의 구체화와 함께 컴퓨터 공학 및 관련 제어공학, 그리고 앞서 언급된 모든 산업분야에서의 응용 시스템의 성능 및 신뢰도 개선을 주도할 수 있다.

참 고 문 헌

[1] K. Shin, C. Krishna, and Y. Lee, "A unified method fro evaluating real-time computer controller and its application," *IEEE Trans. on Automat. Contr.*, vol. AC-30, no. 4, pp. 357-366, Apr. 1985

[2] P. Lala, *Fault Tolerant and Fault Testable Hardware Design*, Prentice Hall, 1985

[3] A. Hopkins, Jr. T. Smith, III, and J. Lala, "FTMP--A highly reliable fault-tolerant multiprocessor for aircraft," *Proc. IEEE*, vol. PROC-66, no. 10, pp. 1221-1239, Oct. 1978

[4] A. Avizienis and G. Gilley, "The STAR (Self-Testing And Repairing) computer: An investigation of theory and practice of fault-tolerant computer design," *IEEE Trans. Comput.*, vol. C-20, no. 11, pp. 1312-1321, Nov. 1971

[5] J. Wensley, L. Lamport, and etc., "SIFT: Design and analysis of a fault-tolerant computer for aircraft control," *Proc. of the IEEE*, vol. 66, no. 10, pp. 1240-1255, Oct. 1978

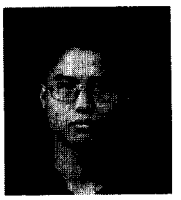
[6] D. Siewiorek, V. Kini, and H. Mashburn, "A case study of C.mmp, Cm* and C.vmp: Part I--Experiences with fault tolerance in multiprocessor systems," *Proc. IEEE*, vol. PROC-66, no. 10, pp. 1178-1199, Oct. 1978

[7] H. Kim and K. G. Shin, "Design and Analysis of an Optimal Instruction-Retry Policy for TMR Controller Computers", *IEEE Transactions on Computers*, vol. 45, no.11,pp. 1217-1225, November, 1996

[8] H. Kim and K. G. Shin, "Sequencing Tasks to Minimize the Effects of Near-Coincident Faults in TMR Controller Computers", *IEEE Transactions on Computers*, vol. 45, no. 11, pp. 1331-1337, November, 1996

[9] K. G. Shin and H. Kim, "A Time Redundancy Approach to TMR Failures Using Fault-State Likelihoods," *IEEE Transactions on Computers*, vol. 43, no. 10, pp.1151-1162, October 1994

저 자 소 개



이대현(李大현)

1974년 7월 21일생. 1997년 연세대 공대 전기공학과 졸업. 현재 연세대 대학원 전기공학과 석사과정.

윤재영(尹在暎)

1974년 12월 11일생. 1997년 연세대 공대 전기공학과 졸업. 현재 동 대학원 전기공학과 석사과정.

김학배(金鶴培)

1965년 10월 10일생. 1988년 서울대 공대 전기공학과 졸업. 1990년 미국 미시간대 대학원 전기공학과(EECS) 졸업(석사). 1994년 8월 동 대학원 졸업(공박). 1994년 9월-1996년 8월 미국 National Research Council(NRC) Rearch Associate st NASA Langley Research Center. 1996년 9월-현재 연세대학교 전기공학과 조교수. 관심분야 : 실시간제어, 자동차공학, 고장허용기법 및 신뢰도 평가.