□특 집□

결함허용 시스템의 설계 고려사항 및 동향

건국대학교 전자계산학과 김 문 회*

목 차

Ⅰ. 서 론

결함허용 시스템이란 하드웨어 오동작, 소프트웨어 에러 또는 정보 오염이 일어날지라도 주어진 임무를 올바르게 수행할 수 있는 시스템을 말한다. 최근 결함허용 시스템에 대한 중요성이 트랜잭션 처리, 실시간 제어, 그리고 인간의 안전에 관련된 응용 분야에서 급속히 증대하고 있다[14]. 예를 들면, 다음과 같은 사례에서 그 중요성을 엿볼 수 있다.

(1) 금년 증권 거래소의 컴퓨터 시스템이 13번이나 고장을 일으켰는데 매번 장시간 증권 거래가 중지되었었다.

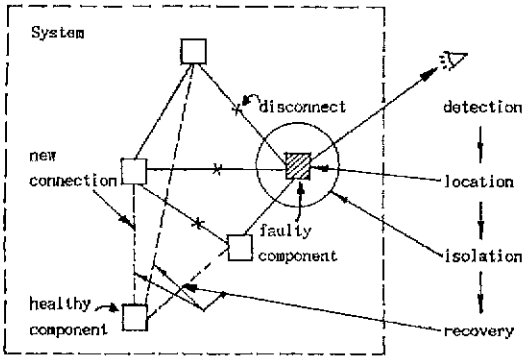
(2) 1992년 철도청의 기차표 예매 시스템이 고장이 나서 철도 이용객들에게 큰 불편을 끼쳤고 기차표 매표소에 극도의 혼잡을 야기시켰었다.

(3) 1990년 미국 AT & T사의 장거리 전화 교환 시스템이 작은 결함으로 인해 고장을 일으켜 수십만 통화의 장거리 전화가 연결되지 못해 미국내 전체에 극도의 혼란을 야기시켰었다.

이와 같은 예에서 볼 수 있듯이 이러한 시스템들이 적절한 결함허용 능력이 있었더라면 극도의 혼란을 피하고 그로 인한 막대한 경제적 손실을 미연에 방지할 수 있었을 것이다.

결함허용 시스템에서는 4가지 기본 기능이 고려되어야 한다; 결함 감지(fault detection), 결함 발생 장소 검출(fault location), 결함 파급 효과 방지(fault isolation) 그리고, 결함으로부터의 복구 및 시스템 재구성(fault recovery and reconfiguration). 결함이 발생되면 적절한 대응 조치가 취해질 수 있도록 반드시 감지되어야 하며 결함이 감지되면 그 원인이 무엇이고 발생 장소가 어디인지를 찾아서 결함이 시스템의 다른 부분의 동작에 나쁜 영향을 미치지 못하도록 고립시켜야 한다. 그 다음 결함이 있는 부품을 여분(redundant)의 결함이 없는 부품으로 교체하거나 시스

* 종신회원



(그림 1) 결함허용의 4가지 기본 기능

템을 재구성함에 의해 시스템이 발생된 결함으로부터 복구되어야 한다. 그림 1은 이러한 일련의 과정을 보여주고 있다. 본고에서는 이러한 결함허용 시스템의 설계시 고려되어야 할 사항들과 현재까지 개발되어 사용되고 있는 결함허용 시스템들을 간략히 소개한다.

본고의 II장에서는 기존의 결함허용 기법들을 간략히 살펴보고 III장에서는 설계시 고려 사항들과 설계시 당면할 수 있는 문제점들을 기술하고 IV장에서는 현재 개발되어 사용되고 있는 결함허용 시스템들을 적용된 결함허용 기법과 함께 소개한다.

II. 기존의 결함허용 기법

기존의 결함허용 기법은 하드웨어 결함허용 기법, 정보 결함허용 기법과 소프트웨어 결함허용 기법 등 시스템 구성요소별로 구분할 수 있다. 모든 기법들이 근본적으로 여분의 부품(redundancy)의 사용에 근거를 두고 있다. 여러 결함허용 기법들이 각 구성요소별로 제안되어 왔는데 그중 대표적인 것들만 간단히 살펴보면 다음과 같다.

2.1 하드웨어 결함허용 기법

• Triple modular redundancy (TMR) : TMR은 결함 차단(fault masking)에 기초를 둔 기법으로 기본 개념은 하드웨어를 삼 중복하여 그 출력을 비교하여 2개 이상의 같은 출력을 TMR의 출력으로 선택하는 것이다.

• Duplication with comparison : 하드웨어를 이 중복하여 그 출력을 비교함으로써 결함을 감지하는 기법이다. 단지 결함 감지의 기능만 존재한다.

• Stand-by sparing : 동작중인 하드웨어에서 결함이 감지 되었을 때 대기중인 여분의 하드웨어로 대체하는 기법으로 결함을 감지할 수 있는 기법과 더불어 사용되어야 한다.

• Watchdog timer : 시스템으로 하여금 주기적으로 timer를 초기화 시키도록 하여 초기화 시키지 않을 때 시스템에 결함이 있음을 감지하는 기법이다.

• Self-purging redundancy : N개의 독립적인 하드웨어 모듈의 출력을 비교하여 다수의 같은 출력을 시스템 출력으로 선택하고 시스템 출력과 다른 출력을 가진 모듈은 스스로 더 이상 계산에 참여하지 않도록 하는 기법이다.

이외에도 Pair and a spare 기법, N-modular redundancy 기법, N-modular redundancy with spares 기법, Sift-out modular redundancy 기법, Pair of comparing pairs 기법 등이 있으며 이들에 대한 자세한 설명은 [9,14]에서 찾을 수 있다.

2.2 정보 결함허용 기법

정보 결함허용 기법은 시스템 입출력 또는 시스템내의 모듈간의 정보 교환시 발생되는 noise, 하드웨어 결함 또는 소프트웨어 결함 등에 의해 정보에 이상이 생기는 것을 감지하거나 방지하기 위해 여분의 정보를 추가하는 것으로써 대표적으로 다음과 같은 것이 있다.

• Parity code : 정보가 2진수로 표시될 때 한 정보에 1의 갯수가 홀수개 또는 짝수개 인지를 알리는 여분의 정보를 추가한다. 정보결함 감지에 사용되며 결함이 짝수개 발생하면 감지할 수 없는 단점이 있다.

• m of n code : 정보의 길이가 n bit일때

1의 갯수가 반드시 한 정보에 m개가 되도록 코드화 한 것으로 1의 갯수가 m개가 아닐 때 결함이 있음을 감지하는 기법이다.

• Checksum : 한 정보 블럭의 정보 내용의 합을 정보 블럭에 추가하는 기법으로 정보 결함 감지를 위한 것이다.

• Berger code : 한 정보내에 있는 1의 개수를 2진수로 표시하여 정보에 추가하는 기법이다.

• Hamming Error Correcting Code : ECC는 결함 검출뿐만 아니라 단일 결함의 교정을 해주는 기능이 있다.

이외에도 Duplication code, Cyclic code, Arithmetic code, Residue code, Inverse-residue code, Horizontal & vertical parity 등이 있는데 이들에 대한 자세한 기술은 여러 관련 문헌에서 찾아볼 수 있다.

2.3 소프트웨어 결함허용 기법

소프트웨어 결함허용 기법은 소프트웨어 모듈의 중복이나 재수행(rollback and retry), 또는 이 두가지 방식의 혼용에 기초를 두고 있다.

• Check pointing : 소프트웨어 실행중에 검사시점을 설정하여 오류가 발생하는지를 검사하여 이상이 없으면 계속 수행하고 이상이 감지되면 그 이전의 검사시점으로 되돌아가 재수행 하는 방식이다[9].

• Recovery block(RB) : 재수행 (rollback and retry)에 근거한 기법으로 검사시점에서 오류가 감지되면 지정된 이전 시점으로 되돌아가 같은 기능을 가진 다른 소프트웨어 모듈을 실행하는 방식으로 단일 프로세스내에 적용될 수 있다[19].

• Conversation : RB의 확장형으로 단일 프로세스가 아닌 서로 정보를 상호 교환하는 프로세스들간에 적용가능한 기법으로 RB와 마찬가지로 재수행(rollback and retry)에 기초를 두고 있다[19].

• Distributed recovery block(DRB) : Recovery block을 분산 환경으로 확장 적용하여 하드웨어 결함과 소프트웨어 결함을 동일한 방법으로 극복할 수 있도록 한 기법이다[14].

• N Self-checking programming : 두 개 이상의 자가진단(self-checking) 부품이 실행되면서 그 중의 하나(수행부품)가 주어진 기능을 수행하고 나머지(대기 부품)는 여분으로 대기 상태에 있다가 자가진단에 의해 수행 부품에 결함이 발견되면 대기 부품중의 하나가 새로운 수행 부품이 되어 주어진 기능을 수행하는 기법이다[16],

• N-version programming : 하드웨어 결함허용 기법 중 TMR과 유사한 기법으로 N개의 독립적인 소프트웨어 모듈이 수행한 결과를 비교하여 다수의 동일한 결과를 채택하는 기법이다[14].

이외에도 목적한 응용 분야에 적합하도록 설계된 여러가지 형태의 소프트웨어 결함허용 기법이 있다[7,14,16,17].

III. 설계 고려사항

결함허용 시스템을 설계할 때 고려되어야 할 사항은 여러가지가 있다. 예를 들면, 그 응용분야의 특성에 따라 요구되는 처리 방식(순차처리, 병렬처리, 또는 분산처리), 결함허용의 정도(degree of fault-tolerance), 시스템 사용 환경의 영향으로 나타날 수 있는 결함의 유형 및 발생 빈도, 결함허용 기법이 적용되는 시스템 계층(granularity), 개발 비용 등이다. 이와 같은 사항을 복합적으로 고려하여 적합한 결함허용 기법을 선정 또는 고안하는데 그 중 몇가지를 본장에서는 간단히 살펴본다.

3.1 소프트웨어 결함허용과 하드웨어 결함허용

응용분야의 특성을 검토하기에 앞서 결함허용을 소프트웨어 위주로 할 것인가 아니면 하드웨어 위주로 할 것인가를 결정하는 것은 매우 중요한 문제중의 하나이다. Triple modular redundancy, Duplication with comparison, Self-purging 등과 같은 하드웨어 결함허용 기법은 결함 감지 및 결함 복구를 신속히 행할 수 있는 반면 Recovery block, N-version programming, N

Self-checking programming과 같은 소프트웨어 결함 기법은 다양한 유형의 결함에 대처할 수 있는 능력(flexibility)이 뛰어나다. 이러한 하드웨어 결함허용 기법과 소프트웨어 결함허용 기법 중 어느 한쪽에만 치우쳐 사용할 경우 시스템에 충분한 결함허용 능력을 부여하기에는 제약이 따르므로 하드웨어 결함허용 기법과 소프트웨어 결함허용 기법이 조화롭게 각 시스템 구성요소에 적용되는 것이 바람직하다. 표 1은 각 시스템 구성요소에 적용될 수 있는 결함허용 기법들을 보여주고 있다.

〈표 1〉 각 시스템 계층에 적용가능한 결함허용 기법

시스템 계층	적용가능한 결함허용 기법
응용	● Recovery block ● Distributed recovery block ● N Self-checking programming ● N-version programming
운영 체제	● Process migration ● Diagnostic procedure ● Reliable interprocess communication
노드	● Duplication with comparison ● Stand-by sparing (hot/cold) ● Self-checking procedure ● Watchdog timer ● Mirrored memory ● Mirrored disk
IC chip	● Triple modular redundancy ● Duplication with comparison ● Sift-out modular redundancy ● Self-purging ● Self-diagnosis
Logic gate or Bus	● Triple modular redundancy ● Error correcting code ● Parity code

3.2 하드웨어 시스템의 구조

응용분야에 적합한 처리 방식에 따라 사용되는 하드웨어 시스템의 구조가 설정되는 것이 바람직하다. 하드웨어 시스템의 구조에 따라 적용할 수 있는 결함허용 기법이 달라질 수 있다. 예를 들어, 단일 프로세서 시스템의 경우 프로세서에 결함이 발생하면 결함으로부터 복구할 수 있는 방법이 없다. 그러므로 여분의 프로세서를 준비하여 실행중인 프로세서에 결함이 발생하면 여분의 결함이 없는 프로세서로 대체하여야 한다. 반면 다중 프로세서 시스템이나 분산 시스템의 경우 하나의 프로세서나 노드에서 결함이 감지되더라도 여분의 프로세서나 노드로 대체하지 않고도 결함이 있는 프로세서나 노드를 시스템으로부터 격리시킨 후 나머지 결함이 없는 프로세서와 노드들을 가지고 시스템을 재구성하여 시스템이 정상적인 기능을 수행하도록 할 수 있다. 즉, 시스템에 내재된 여분의 부품(inherent redundancy)이 있을 경우 이를 활용하여 결함허용 능력을 부여할 수 있다. 물론 시스템에 내재된 여분의 부품이 있을지라도 시스템이 더 좋은 결함허용 능력을 갖추도록 새로운 여분의 부품을 사용할 수도 있다.

3.3 요구되는 결함허용의 정도

응용분야에 따라 요구되는 결함허용의 정도가 달라질 수 있다. 예를 들면, 결함이 발생하여 시스템이 오동작을 일으키면 막대한 재산 손실을 가져오거나 인간의 안전을 위협할 수 있는 우주항공 시스템이나 핵반응기 제어 시스템과 같은 고신뢰도를 요하는 응용분야에서는 신속한 결함 감지 및 결함으로부터의 복구 능력 등 높은 정도의 결함허용 능력을 필요로 하는 반면 일반 응용 분야에서는 단순한 결함 감지 능력으로도 충분한 경우가 대부분이다. 요구되는 결함허용의 정도에 따라 적용되는 결함허용 기법이 달라질 수 있다. 예를 들면, 정보상에 결함의 유무를 감지하는 기능만이 필요하면 Parity code를 사용하면 되는 반면, 결함의 유무뿐만 아니라 정보상의 단일 비트 오류로부터 복구하는 기능이 필요하면 Hamming code와 같은 Error correcting code를 사용하여야 한다. 일반적으로 요구되는 결함허용의 정도가 높을수록 적용되는 결함허용 기법이 복잡해지고 필요한 여분의 부품의 양도

증가한다. 예를 들면, Parity code의 경우에는 정보상의 데이타 비트에 하나의 여분 비트를 추가하면 되지만 Hamming code의 경우에는 데이타가 8비트로 구성되면 4개의 여분 비트를 추가하여야 한다.

3.4 후향 복구와 전향 복구

후향 복구(backward recovery) 기법은 일명 "rollback and retry"기법이라 불리우기도 하는데 시스템 동작중에 주기적으로 또는 미리 정해진 시점에서 시스템 상태를 검증하여 저장한다. 만일 검증시 결함이 감지되면 시스템을 그 전 시점에 저장된 상태로 재설정하여 그 시점으로부터 시스템 동작을 재수행함으로써 결함으로부터 복구하는 기법이다. 이 기법의 단점은 결함으로부터 복구하는데 많은 시간이 소요된다는 점이다. 그러므로, 시간적인 제약 조건이 주어진 응용 분야에는 부적합하다.

전향 복구(forward recovery)기법은 후향 복구 기법과는 다르게 결함의 감지시 시스템의 상태를 그 전 시점으로 재설정하지 않고 계속적으로 현 상태로부터 동작을 수행해 가며 발생된 결함으로부터 복구하는 기법이다. 이러한 기법을 적용할 때 설계시 저장된 전 시스템 상태 정보중 어떠한 정보를 그리고 어떻게 전향 복구를 이루는데 사용할 수 있는가를 고려하여야 한다. 이러한 기법의 대표적인 예로는 Triple modular redundancy 기법, Distributed recovery block 기법, N Self-checking programming 기법, N-version programming 기법 등을 들 수 있다. 이 기법들의 단점으로는 후향 복구 기법보다 많은 양의 여분 부품이 필요한 것인데 최근에는 하드웨어 가격의 저렴화로 이러한 기법들을 실제 시스템에 적용하려는 시도가 활발하다.

3.5 결함 유형 및 발생 빈도

결함 유형은 다음과 같은 다섯가지 관점에서 구분하여 볼 수 있다.

• 결함원인 : 결함 원인은 여러가지가 있다. 즉, 설계상의 실수, 외부 환경으로부터의 영향, 구현시의 실수, 또는 부품결함 등을 들 수 있다.

• 결함속성 : 결함이 하드웨어에서 발생하였는지 소프트웨어에서 발생하였는지에 의해 구분될 수 있다.

• 결함지속시간 : 결함은 영구적일 수도 있고, 일시적일 수도 있다 또한, 주기적이든 비 주기적이든 반복적으로 나타날 수도 있다.

• 결함범위 : 결함이 미치는 영향이 국부적일 수도 있고, 시스템 전체 동작에 영향을 미칠 수 있다.

• 결함의 상태 : 발생된 결함 상태가 항상 일정할 수도 있고 시간에 따라 그 상태가 변할수도 있다.

이와 같이 다양한 결함 유형에 따라 다른 결함 허용 기법이 적용되어질 수 있다. 예를 들면, 결함이 일시적이었다면 결함이 발생된 부품을 추후에 재사용할 수 있지만 영구적이었다면 다른 결함이 없는 부품으로 교체하여야 할 것이다.

3.6 결함허용 기법이 적용되는 시스템 계층

표 1에서 본 바와 같이 시스템 계층에 따라 적용되어지는 결함허용 기법이 달라진다. 뿐만 아니라 시스템 동작 수행 단위에 따라 적용되어지는 결함허용 기법이 달라진다. 시스템 동작 수행 단위는 낮은 것부터 높은 것 순으로 마이크로 인스트럭션(μ-instruction), 인스트럭션, 태스크 등의 세가지 계층으로 구분할 수 있는데 동작 수행 단위가 낮을수록 신속한 결함 감지 및 복구를 수행할 수 있으므로 상위 계층에서의 전향 복구를 지원할 수 있으나 짧은 주기로 결함이 존재하는지를 검사하여야 하므로 그만큼 오버헤드가 커질 수 있는 단점이 있다.

3.7 기타 사항

일반적으로 결함허용 기법의 기본 개념은 단순하지만 실제로 결함허용 시스템을 설계하는 데에는 많은 어려움이 따른다. 그 원인들을 몇가지 열거하여 보면 다음과 같다.

• 결함의 유형 및 발생빈도를 실제로 동작해 보기 이전에는 알기가 어렵다. 예를 들면, 하드

〈표 2〉 결함허용 시스템

시스템	용 도	적용된 결함허용 기법	응용분야
1. Self-Testing And Repairing computer (STAR)	Unmanned space flights of 10 years or longer	· Reconfiguration · Stand-by sparing · Watchdog timer · Comparison · TMR	Long-life application
2. Fault-Tolerant Spaceborn Computer (FTSC)	Unattended service in space	· Watchdog timer · Pair and a spare	Long-life application
3. Fault-Tolerant Building Block Computer (FTBBC)	Military application	· Duplexing · Error correcting code · Self-checking	Long-life application
4. Tandem 16 Nonstop Computer	Transaction processing market	· Duplication · Checksums · Parity error check · Error correcting memory · Watchdog timer	High-availability application
5. Tandem Nonstop Cyclone Computer	Transaction processing market	· Duplication · Error correcting memory · Mirrored disk	High-availability application
6. Stratus/32 System	Transaction processing market	· Duplication with comparison · Pair and a spare	High-availability application
7. Electronic Switching System	Telephone switching	· Self-checking · Stand-by sparing · 4-of-8 code · single parity bit	High-availability application
8. Synapse N+1 Computer	Transaction processing industry	· Self-dispatching · Self-test	High-availability application
9. COPRA	Aerospace application	· Roll-back and retry · Parity error check · TMR	Critical-computation application
10. Computer Voted Multiprocessor (C.vmp)	Real-time computation	· TMR · Deactivation of system components	High-availability application
11. Sperry Univac 110/60	범 용	· Comparison · Parity error check · Diagnostic program	범 용

12. Pluribus	ARPA network	· Parity error check · Stand-by sparing	High-availability application
13. AXE	Telephone exchange system	· Stand-by sparing · Self-test	High-availability application
14. Computer Aided Traffic Control System (COMTRAC)	Controlling Japanese train Shinkansen	· Comparison · Stand-by sparing	Critical-computation application
15. Space Shuttle Computer	Space flight	· NMR · Built-in-test · Bus time-out test · Watchdog timer	Critical-computation application
16. Software Implemented Fault Tolerance (SIFT)	Commercial transport aircraft	· TMR	Critical-computation application
17. Fault-Tolerant Multiprocessor (FTMP)	Commercial transport aircraft	· TMR	Critical-computation application
18. August System CS-3001 Control Computer	Industry control and monitoring	· TMR	Critical-computation application
19. Multi-Microprocessor Flight Control System (M^2FCS)	Flight control	· Duplication with comparison · Self-checking	Critical-computation application
20. Agusta A129 Integrated Multiplex System (IMS)	Helicopter control	· Dual redundancy · Watchdog timer	Critical-computation application
21. Triplex 32	범 용	· TMR	범 용
22. Basic Fault-Tolerant System (BFS)	Network management	· Neighbor test	High-availability application
23. FASP	Space-based signal processing	· Error correcting code · Self-test	Critical-computation application
24. Sequoia series 400	Transaction processing	· Error correcting code · Duplication with comparison · Mirrored memory · Mirrored disk · Dual bus	Transaction processing

25. VAX ft3000	범 용	· Error correcting code · Duplication with comparison · Mirrored memory · Mirrored disk	범 용
26. Hitachi ft6100	범 용	· Pair and a spare · Dual bus · Mirrored memory · Mirrored disk	범 용

웨어 부품에 일어나는 영구적 결함에 대해서는 타당한 모델(즉, Stuck-at-fault 모델 등)이 존재하지만 일시적 결함이나 반복적 결함에 대한 모델은 적합한 것이 존재하지 않는다.

• 다양한 종류의 결함허용 기법 중 적합한 기법을 선택하여야 하고 또한, 시스템 계층별로 다른 결함허용 기법이 적용되는데 이 특성이 다른 결함허용 기법들을 조화롭게 통합하여야 시스템의 결함허용 능력이 증대될 수 있다.

• 시스템은 반드시 요구되는 성능(performance requirement)을 충족시켜야 한다. 결함 감지 및 복구 기능을 시스템에 부여하게 되면 필연적으로 그에 따른 시간적 오버 헤드가 있다. 즉 성능의 저하를 가져올 수 있다. 특히, 실시간 제어 응용분야에서는 이러한 오버헤드를 극복하고 요구되는 성능을 충족시키기가 쉽지 않다.

• 시스템에 결함허용 능력을 부여하게 되면 시스템의 복잡도가 증가한다. 시스템의 복잡도가 증가하면 시스템 설계나 구현시 오류를 범할 가능성이 높아진다.

이러한 점 이외에도 다음과 같은 여러 실제 제약 조건이 결함허용 시스템의 설계에 영향을 준다.

- 개발 비용
- 개발 기간
- 시스템의 무게
- 설계 변경의 용이성
- 시스템의 크기
- 유지 보수성
- 시스템의 전력 소모 등

IV. 결함허용 시스템의 동향

현재 결함허용 시스템은 주로 시스템의 결함이 인간의 안전에 위협을 줄 수 있거나 막대한 재산상의 손실을 끼칠 수 있는 유인 우주 비행이나 핵반응기 제어와 같은 고신뢰도를 요하는 응용분야(critical computation application), 무인 우주 항공과 같이 장시간의 오류 없는 시스템의 동작 수행을 요하는 응용분야(long-life application), 또는 트랜잭션 처리나 전화 교환망과 같은 높은 가용성이 요구되는 응용 분야(high-availability application) 등에 사용되고 있다. 표 2는 현재 상용화 되어 있거나 실제로 개발되어 사용되고 있는 결함허용 시스템을 보여주고 있다. 표 2에서 보듯이 주로 하드웨어 결함허용 기법이 실제 시스템에 적용되고 있고 소프트웨어 결함허용 기법을 적용한 시스템은 극히 드물다. 소프트웨어는 하드웨어 결함허용 기법이나 정보 결함허용 기법의 효율성을 극대화 하는데 필수적이나 소프트웨어 결함허용 기법은 아직 연구 수준에 머물러 있고 실용화되어 있지 못한 실정이다. 그러나, 현재 이 분야의 발전추세에 비추어 가까운 장래에 실용화 되어질 것으로 보인다.

현재 많이 활용되고 있는 하드웨어 및 정보 결함허용 기법은 Triple modular redundancy, Stand-by sparing, Duplication with comparison, Parity code, Error correcting code 등이며 최근 새로운 상용 결함허용 시스템에는 이 중복에 기초를 둔 Mirrored memory, Mirrored disk 기법과 소프트웨어에 의한 온라인 수리(On-line repairing) 기능 등을 활용하고 있다.

V. 맺음말

결함허용 시스템은 서론에서 언급한 바와 같이

매우 중요하며 그 수요가 나날이 증대하고 있다. 현재 많은 결함허용 시스템들이 개발되어 사용되고 있고 하드웨어 가격의 하락으로 경제성 있는 상용 결함허용 시스템들이 출현하여 여러 분야에서 활용되고 있다. 그러나, 보다 나은 결함허용 시스템에 대한 필요성이 급속히 증대하고 있으며 이에 대한 연구도 활발히 진행되고 있다. 본고에서는 결함허용 시스템의 설계시 고려 사항과 현재 개발되어 사용되고 있는 결함허용 시스템들에 대해 간략히 소개하였다.

참 고 문 헌

1. Armstrong, C. V. M. and Fathi, E. T., "A Fault-Tolerant Multimicroprocessor-based Computer System for space-based Signal Processing", IEEE Micro, vol. 4, no. 6, pp. 54~65, 1984.
2. Avizienis, A. and Kelley, J. P. J. "Fault Tolerance by Design Diversity: Concepts and Experiments", Computer, vol. 17, no. 8, pp. 67~80, Aug. 1984.
3. Brenstein, P. A., "Sequoia: A Fault-Tolerant Tightly Coupled Multiprocessor for Transaction Processing", Computer, pp. 37~45, Feb. 1988.
4. Emmerson, R. and McGowan, M., "Fault Tolerance Achieved in VLSI". IEEE Micro, vol. 4, no. 6, pp. 34~43, 1984.
5. Friedman, A., and Simoncini, L., "System-Level Fault Diagnosis", Computer, vol. 13. no. 3, pp. 47~53, March 1980.
6. Hayes, J., "Testability considerations in Microprocessor-Based Design", Computer, vol. 13, no. 3, pp. 17~26, March 1980.
7. Huang, H. H., "Fault-Tolerant design of a modern receiving system", Proceedings of FTCS-10, pp. 375~378, 1980.
8. Johnson, B. W., "Fault-tolerant Microprocessor-based Systems", IEEE Micro, vol. 4, no. 6, pp. 6~21, 1984.
9. Johnson, B. W. *Design and Analysis of Fault-Tolerant Digital Systems,* Addison Wesley, 1989.
10. Kawakubo, K., Nakamura, H., and Okamura, I., "The architecture of a fail-safe and fault-tolerant computer for railway signalling device", Proceedings of FTCS-10, pp. 372~374, 1980.
11. Kim, K. H. and Yang, S. M. "Fault Tolerance Mechanisms In Real-Time Distributed Operating System: An Overview", Proceedings of Pacific Computer Communication Symp., pp. 220~229, Oct. 22~24, 1985.
12. Kim, K. H., "Error Detection, Reconfiguration and Recovery in Distributed Processing Systems", Proc. IEEE 1st Conf. on Distributed Computing Systems, Oct. 1979, pp. 284~295.
13. Kim, K. H., "Issues in Design of Temporary Blockout Handling Capabilities into Tightly Coupled Computer Networks", Software for Strategic Systems Conference. Oct. 19 88.
14. Kim, K. H., "Design of Real-Time Fault-Tolerant Computing Stations", Lecture note in the NATO Advanced Science Institute on Real-Time Computing, Sint Maarten, Oct. 1992.
15. Lala, P., *Fault Tolerant & Fault Testable Hardware Design,* Prentice Hall, 1985.
16. Laprie, J-C, *et al.,* "Definition and Analysis of Hardware- and Software-Fault-Tolerant Architectures", Computer, pp. 39~51, July 1990.
17. McGill, W. F. and Smith, S. E., "Fault Tolerance in Continuous Process Control", IEEE Micro. vol. 4, no. 6, pp. 22~33, 1984.
18. Pradhan, D. ed., "Fault-Tolerant Computing Theory and Techniques", vol. II, Chap. 8, 1986.
19. Randell, B., "System Structure for Software fault tolerance", IEEE Trans, on Software Engr., June 1975, pp. 220~232.
20. Rennels, D., "Distributed Fault-Tolerant Compyter System", Computer, vol. 13, no. 3, pp. 55~65, March 1980.
21. Schmitter, E. J. and Baues. P., "The Basic Fault-Tolerant System", IEEE Micro, vol. 4, no. 1, pp. 65~74, 1984.
22. Siewiorek, D. P., "Fault Tolerance in Com-

mercial Computers", Computer, pp. 26~37, July 1990.

김 문 회

1979 서울대학교 공과대학 전기공학과 (B.S.)
1981 서울대학교 공과대학 전기공학과 (M.S.)
1983 ~1985 University of South Florida, M.S. in Computer Science
1986 ~1991 University of California, Berkeley, (Ph.D.)
1986 ~1991 UCB ERL Post Graduate Researcher (연구원)
1991 ~현재 건국대학교 전자계산학과 조교수

관심 분야 : 소프트웨어 공학, Real-time distributed computing system, Fault-tolerant computing system, Network management.