정 보 보 호 학 회 지
제24권 제2호, 2014. 4

결함주입기법을 이용한 차량용 고신뢰성 임베디드 시스템의 안전성 검증방안

이 동 우*, 류 대 현**, 나 종 화***

요 약

자동차 전장제품 활용의 급속한 증가에 대응하기 위하여 자동차 분야에서는 ISO 26262 안전설계절차를 도입하여 차량용 임베디드 시스템의 안전성을 확보하려고 노력하고 있다. ISO 26262는 자동차에서 발생 가능한 비정상상태(abnormal state)를 식별하고 그의 영향을 분석하며 전체 시스템의 안전을 검증하는 것을 목표로 하고 있다. 다양한 종류의 부품이 연동되는 복잡한 시스템의 안전 검증은 결함수목법과 고장모드영향분석법을 활용하는 위험분석법이 보편적으로 사용된다. 결함주입시험은 이러한 위험분석의 기반도구로서 안전성을 향상시키기 위하여 사용된 고장감내 기능의 동작여부 및 그에 따른 시스템의 안전성을 검증하는 목적으로 사용된다. 본 논문에서는 차량용 고신뢰성 임베디드 시스템에서 사용되는 고장감내 메카니즘들의 기능과 안전을 검증하는 방법과 사례를 소개한다. 최근의 복잡한 차량용 임베디드 시스템의 개발은 상위수준의 모델을 개발하여 지정된 위험 및 고장을 초래하는 결함을 시스템에 주입하고 그의 결과를 분석하여 안전을 검증하는 것이 일반적인 방법이다. 개발 목표 차량의 임베디드 시스템 모델을 개발하고, 식별된 결함의 결함모델을 준비한 뒤, 시스템 모델 기반 결함주입 도구를 이용하여 결함주입을 수행하는 시험방법과 그 결과에 대하여 논의한다. 하드웨어는 SystemC 하드웨어 설계언어를 이용하여 개발하고, 소프트웨어를 컴파일하여 실행화일을 확보하여 시험대상인 결함모델을 개발하고 이를 대상으로 결함주입시험에 대해 설명한다.

Ⅰ. 서 론

최근 신형 고급승용차들의 해킹이 보도되면서 사회 문제화 되고 있다. 이러한 문제의 해답은 보안(security) 기능을 구현하는 것이지만, 문제의 출발은 안전(safe)한 시스템을 구현하는 것이다. 차량용 임베디드 시스템은 일반적인 상용 임베디드 시스템과는 달리 고신뢰성이 요구된다. 차량용 임베디드 시스템은 운전 중에 고장 수리가 힘들며, 사고발생 시에 대형 참사로 이어진다. 이러한 이유로 현재의 차량 임베디드 시스템은 신뢰성을 보장하기 위한 인증 시스템을 갖추고 있다. 차량용 임베디드 시스템의 소프트웨어는 ISO26262 인증이나 이에 준하는 인증시스템에 통과한 제품만을 사용한다[10].

그러나 ISO26262인증은 규격에서 제시된 기본적인 안전설계절차대로 개발되었다는 의미로서 설계자가 인지하지 못한 설계규격에 대해서는 대책이 없는 것이 현실이다. 글로벌 자동차 업체들 간의 치열한 경쟁에 따른 개발 기간과 비용의 감소로 인하여 예측하지 못하는 많은 설계결함이 발생 할 수 있다. 또한 태양의 흑점 폭발, EMI, 전원의 급격한 변동 등의 차량운용환경에 따른 외부적 요인 역시, 차량 임베디드 시스템에 발생할 수 있는 고장의 원인이 될 수 있다. 따라서 차량 임베디드 시스템은 유사시 발생할 가능성이 있는 결함을 방지하거나, 그의 따른 피해를 최소화 하도록 안전하게 설계하는 것이 중요하다.

차량용 임베디드 시스템의 설계자는 설계초기단계부터 고장감내 기법을 고려해야 한다. 고장감내 기법은 결함이 발생할 경우, 결함을 제거하고, 이로 인한 피해를 최소화 하는 방법으로서 일반적으로 다중화(redundancy) 메커니즘을 활용한다[8]. 다중화는 임베디드 시스템 설계과정에서 다양하게 구현할 수 있으며, 대표적으로 하드웨어 다중화, 소프트웨어 다중화, 정보

* 항공대학교 항공전자공학과
** 한세대학교 IT학부 (dhryu@hansei.ac.kr)
*** 항공대학교 항공전자공학과 교수

다중화, 시간 다중화로 구현한다[1]. 고장감내 설계 기법은 시스템의 특성에 따라 다양하게 적용 가능하며, 적용한 고장감내 설계 방법에 따라 시스템의 신뢰도 및 가용성 등이 결정된다. 그러나 시스템에 적용한 고장감내 기법에 따른 신뢰도의 차이를 정량화하는 것은 많은 비용과 시간이 필요하다.

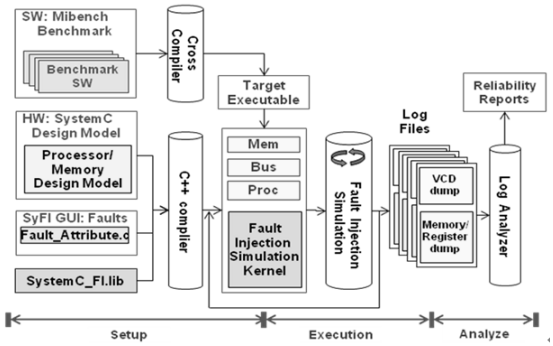
본 논문에서는 고장감내 기법의 기능을 평가하고 신뢰도를 정량적으로 측정하기 위한 방법으로 결함주입 기법을 사용하였다. 결함주입 기법은 시스템에 임의적인 결함을 주입하고 시스템의 고장 여부를 분석한다. 이를 통해 대상 임베디드 시스템의 고장률 및 신뢰도를 계산한다.[3]. 완벽한 고장감내 기법이 적용된 시스템의 경우, 결함이 주입되어도 시스템의 동작에 영향을 주지 못할 것이다. 그러나 고장감내 기법이 적용되지 않은 시스템 설계자가 인지하지 못한 결함이 발생하는 경우에 고장(failure)이 발생할 수 있다. 결함주입 방법은 결함을 주입하는 방법에 따라 사보추어(saboteur)[4], 뮤턴트(mutant)[4], 커널기반 결함주입[5,6] 방법이 있다. 본 논문에서는 커널기반 결함주입 기법인 SyFI[5]를 사용하여 시뮬레이션 모델에 결함을 주입하고, 대상 임베디드 시스템의 고장률 및 신뢰도를 평가한다.

본 논문은 RISC 프로세서를 기본모델로 하고, 고장감내 기법을 적용한 고장감내형(fault tolerant) RISC 프로세서(FTR)를 개발하였다. RISC 프로세서는 효율적인 하드웨어의 설계를 위하여 최근 등장한 Electronic System Level (ESL) 설계방법인 SystemC 하드웨어 개발도구를 사용하였다[7]. 고장감내 기법은 하드웨어 다중화 기법을 적용하였으며, 각 모델에 따라 다중화 구조를 달리 하였다. 본 논문에서는 fetch모듈을 3중화한 Fetch Redundancy RISC(FRR)와 프로세서의 전체 모듈을 3중화한 Redundancy -Execute RISC(RER)를 시험한다[9].

본 논문의 구성은 다음과 같다. 2장에서는 커널기반 결함주입 기법인 SyFI를 설명한다. 3장에서는 기반 프로세서인 RISC 프로세서를 설명하고, 고장감내 기법이 적용된 FRR, RER을 설명한다. 4장은 설계한 각 프로세서에 결함을 주입하고, 결과를 분석한다. 5장에서 결론을 맺는다.

Ⅱ. SyFI : SystemC Kernel-based Fault Injection

SyFI는 SystemC 모델을 대상으로 커널기반 결함주입 실험을 수행하기 위한 환경이다. 그림 1은 SyFI의 실험절차를 설명한다. SyFI는 (1) 설정, (2) 실행, (3) 평가의 3단계를 통해 결함주입 시뮬레이션을 수행한다.



〔그림 1〕 SyFI 결함주입 환경[5]

2.1 설정단계

SyFI는 설정단계에서 (1) SystemC 하드웨어 디자인 모델, (2) 대상 하드웨어의 소프트웨어, (3) 결함주입 실험 설정파일을 생성한다. SystemC 하드웨어 디자인 모델은 RISC 프로세서 시뮬레이션 모델을 사용하며, 소프트웨어는 RISC 프로세서에서 수행 가능한 테스트용 어셈블리 코드를 사용한다. 하드웨어 시뮬레이션 모델과 소프트웨어의 설정 후, 결함주입 시뮬레이션을 수행하기 위한 설정 파일을 작성한다. 표 1은 결함주입 설정파일에서 설정하는 속성들을 보여주고 있다. 결함속성은 결함발생위치, 결함 발생 시간, 결함 발생빈도, 결함 유형으로 설정된다. 결함 발생 빈도는 일시적(transient) 결함과 영구적(permanent) 결함으로 설정할 수 있다. 간헐적(intermittent) 결함은 결함 발생 이후에 일정 간격으로 반복적으로 발생하는 결함 시나리오를 지원한다. SyFI는 결함의 유형을 직접 선택하거나, 임의적으로 발생할 수 있도록 설정 할 수 있다. 결함 발생위치, 시간은 시뮬레이션 커널의 준비단계에서 임의적으로 설정하거나, 사용자가 선택할 수 있다.

〔표 1〕 결함주입 시나리오 속성〔5〕

Fault Attributes	Value
Fault Type	Transient, Permanent, Intermittent
Fault Time	Random or Deterministic
Fault Location	Random or Deterministic
Fault Model	Stuck-at-0(1), Stuck-at-multi-bit Bit flip, Open, short, Bridge
Fault Interval	periodic or aperiodic

2.2 실행단계

실행단계에서는 시뮬레이션 파일을 실행하고, 시험 모델에 결함을 주입한다. 시뮬레이션을 완료하면 결과 파형을 기록한 VCD(value change dump) 파일을 출력한다. 그 밖에도 분석에 필요한 메모리/레지스터 덤프 파일, 추적 메시지 등을 확보 할 수 있다. 본 실험환경에서는 VCD 파일과 함께, ram 파일, 레지스터 파일을 분석파일로 추출하고 있다. 결과 파일은 로그 분석기를 통해 결함에 의한 동작 특성의 변화를 세부적으로 분석한다.

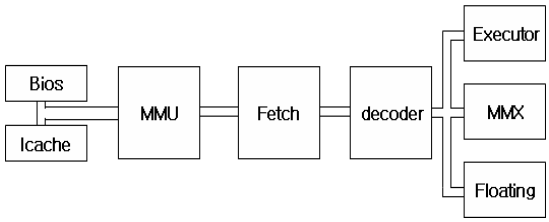
2.3 평가단계

SyFI 분석기는 시뮬레이션 수행으로 산출된 VCD 파일과 메모리 덤프파일을 결함주입을 수행하지 않고 산출된 무결함 시험결과와 비교한다.

Ⅲ. 고장 감내형 하드웨어 설계

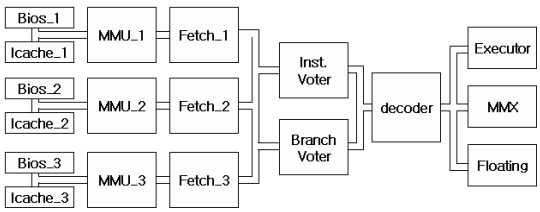
3.1 RISC 프로세서 설계

RISC 프로세서는 Synopsys에서 SystemC로 개발한 마이크로프로세서 모델이다. 그림 2는 RISC 프로세서의 구성도이다. RISC 프로세서는 인출(fetch), 디코더(decoder), 실행(executer)의 3단으로 구성되어 있다.



〔그림 2〕 RISC 프로세서 구성도〔9〕

테스트 소프트웨어는 어셈블리 프로그램으로 작성되며, 전용 어셈블러를 사용하여 기계어로 번역한다. 번역한 기계어 코드는 시뮬레이션 시작과 동시에 명령어 캐시에 로드(load)된다. 로드된 명령어 코드는 MMU을 거쳐 fetch로 전달된다. Fetch로 로드 된 명령어는 decoder로 전달되어 데이터와 컨트롤 신호를 만들어 준다. RISC 프로세서는 Register 모듈이 Decoder 내부에 위치하도록 설계 되었다. 데이터와 제어신호는 Executor로 전달되어 연산을 수행한다. 명령어의 수행결과는 Decoder로 보내져 Register에 저장된다.

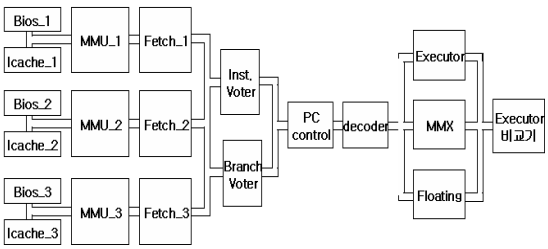


〔그림 3〕 FRR 프로세서 구성도〔9〕

3.2 FETCH REDUNDANCY RISC (FRR)

Fetch Redundancy RISC(FRR)은 RISC 프로세서를 기본 구조로 하여, fetch부를 다중화한 고장감내 프로세서이다. 그림 3는 FRR모델의 구성도를 보여준다. 그림에서 보는 바와 같이 FRR은 명령어 인출을 수행하는 bios, icache, MMU 모델을 각각 삼중화 하여 신뢰도를 향상 시킨다. 각 모듈에서 나오는 데이터 값과 제어신호는 비교기로 보내져서 결함 값을 검출하고, 제거한다. 인출부 모듈의 삼중화 되었지만, 병렬적으로 처리되기 때문에 추가적인 클락 소모는 없다. 다만 voter 동작에 의해 각 명령어에 따라 1 clock이 소모된다. FRR은 Fetch부에서 발생되는 결함을 검출하고 복구 할 수 있

다. 그러나 decoder, executor 쪽에서 발생되는 결함을 검출 할 수 없는 단점이 있다. 이를 보완하기 위해 RER 모델을 개발하였다.



〔그림 4〕 RER 프로세서 구성도[9]

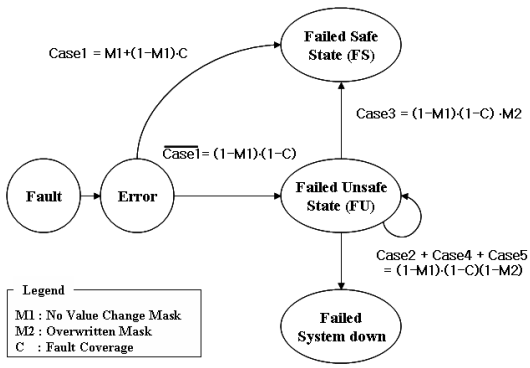
3.3 Redundancy Execute RISC(RER)

RER는 시간 다중화기법을 사용하여 decoder 와 executor 부에서 발생하는 결함을 검출하고 복구 한다. 그림 4은 설계한 RER의 구조를 보여주고 있다. 그림에서 보는 바와 같이 pc_control과 Exec voter가 추가되었다. Pc_control은 fetch로부터 받은 명령어를 일정 간격으로 2번씩 decoder에 전달한다. 따라서 동일 명령어를 2번씩 처리하게 된다. Exec voter는 동일한 명령어의 두 결과를 비교하고 일치 한다면, Register에 결과값을 저장한다. 그러나 명령어 수행 중에 결함이 발생하여 결과가 동일하지 않을 경우, pc_control 결함 발생 신호를 인가하여, 이전 명령어를 다시 수행한다.

Ⅳ. 실험결과 및 분석

그림 5은 시뮬레이션 모델에 결함 주입에 따른 상태 변화를 보여주고 있다. 그림에서 보는 바와 같이 시스템의 상태변화를 5가지의 경우로 분류 할 수 있다.

- case1: 주입된 결함 값이 마스킹 되거나, 고장감내 기법에 의해서 결함이 정정되어 안전적인 상태를 유지하고 있음을 의미한다.
- case2: 주입된 결함이 시스템에 지속적으로 유지되어 시스템의 운용에 위험요소로 작용하는 상태를 의미한다.
- case3: 주입된 결함 값이 시스템에 반영되었으나, 프로그램 수행도중 결함 값이 자연적으로 마스킹 되어, 안정적인 상태를 유지하는 경우이다.
- case4: 결함에 의해 시뮬레이션 모델의 동작이 중지하는 경우이다.
- case5: 결함주입 값으로 인해 시뮬레이션 동작에 문제가 생기는 경우이다. 시뮬레이션 모델은 C++ 언어로 개발되었다. 그러므로 C++언어의 취약점 중 하나인 메모리 관리 문제가 시험모델에서 발생할 수 있다. 본 실험에서는 종종 배열로 선언되어 있는 레지스터 모듈에서 오버플로어가 관찰되었다. 그 이유는 레지스터 번지를 선택하는 신호에 결함이 주입되어, 32개로 정해져 있는 레지스터 번지보다 큰 값을 조회하면서 발생한다.



〔그림 5〕 결함주입에 따른 시스템 상태 변화[9]

〔표 2〕 결함주입 실험결과

	fault	case1	case2	case3	case4.5
RISC	T	79.3%	2.1%	9.65%	8.95%
	P	31.95%	25.05%	1.5%	41.5%
FRR	T	96%	1%	2%	1%
	P	91.15%	16.2%	1.8%	4.35%
RER	T	93.60%	1.15%	2.45%	2.80%
	P	85.05%	8%	0.4%	6.55%

Ⅳ. 실험결과 및 분석

표2는 설계한 고장감내형 프로세서에 결함주입 시뮬레이션을 수행한 결과이다. 표의 case1,3은 결함주입 이후에 시스템이 안전한 생태이며, case 2,4,5는 불안전하거나 시뮬레이션 동작이 멈춘 상태를 의미한다. 결함주

입 실험은 프로세서 모델에 따라 일시적 결함(T)와 영구적 결함(P)를 각 500회씩 실험 하였다. 표에서 보는 바와 같이 RISC 프로세서에 비해 고장감내형 기법이 적용된 FRR, RER의 안전성이 더 높음을 확인 할 수 있다. 이와 같은 차이는 일시적인 결함에 비해 영구적 결함을 주입했을 때 더 확실히 나타난다. RER는 FRR에 비해 고장감내형 기법을 더 적용했음에도 불구하고, 결함주입 실험을 통한 결함처리 능력은 더 떨어지는 것으로 나타났다. RER가 FRR에 비해 결함 처리율이 떨어지는 이유는 1) 반복 수행에 의한 결함검출은 영구적인 결함을 주입할 경우, 결함검출이 불가능 하며, 2) 명령어를 반복수행하고, 제어하기 위해 요구되는 추가적인 모듈과 제어선이 결함에 치명적인 취약부로 작용하여, 결함 처리율을 저하시키는 것으로 분석하였다.

Ⅴ. 결 론

본 논문은 다양한 기법을 적용한 고장감내형 프로세서를 설계하고, 이를 검증하기 위한 방법으로 결함주입 시뮬레이션 기법을 설명하였다. RISC 프로세서를 기본 모델로 하여, 프로세서의 인출 부를 삼중화 한 FRR 프로세서, 실행부에 결함을 검출하기 위해 시간다중화 기법을 적용한 RER 프로세서를 systemC로 설계하였다. 커널기반 결함주입 툴인 SyFI를 사용하여, 결함주입 실험을 수행하고, 프로세서의 결함처리능력을 평가 하였다. 평가결과 FRR 프로세서가 가장 좋은 결함처리율을 갖고 있음을 확인 할 수 있었다. 고 신뢰성을 요구하는 차량용 임베디드 시스템을 설계할 경우 설계 시뮬레이션 모델을 대상으로 결함주입 시뮬레이션을 진행하여, 결함 처리율을 비교 분석하고, 안전한 시스템 설계에 활용 할 수 있을 것으로 사료된다.

참 고 문 헌

[1] Barry W. Johnson, "Design and Analysis of Fault Tolerant Digital Systems", Addison Wesley Publishing Company, 1988

[2] Lsrael Koren and C. Mani Krishna, "Fault-Tolerant Systems", Morgan Kaufmann, 2007

[3] Shubu Mukherjee, "Architecture Design For Soft Errors", Morgan Kaufmann, 2008

[4] Daniel Gil, Juan Carlos Baraza, Joaquin Gracia and Pedro Joaquin Gil, “VHDL Simulation-Based Fault Injection Techniques”, Fault Injection Techniques and Tools for Embedded systems Reliability Evaluation, pp159-176, 2003

[5] Dongwoo Lee, Jongwhoa Na, "A Novel Simulation Fault Injection Method for Dependability Analysis" IEEE Design & Test Computers, 11-12. 2009

[6] Jongwhoa Na, Dongwoo Lee, "Simulated Fault Injection Using Simulator Modification Technique", ETRI Journal, Volume 33, no1. Feb, 2011

[7] Open SystemC Initiative (OSCI), http://www.systemc.org/home

[8] Daniel J. Sorin, "Fault Tolerant Computer Architecture", Morgan & Claypool Publishers, pp19-22, 2009

[9] 이동우, “Electronic System Level 수준의 시뮬레이션 기반 System on Chip 설계 및 신뢰성 검증 환경에 관한 연구”, 한국항공대학교, 2008

[10] 채승엽, 김원종, “ISO 26262에 따른 차량용 ECU 소프트웨어와 SoC 대응방안”, 정보처리학회지, vol 19. no 3, pp73-81, 2012

〈저 자 소 개〉

이 동 우 (李東雨)
학생회원

2006년 2월 : 한세대학교 정보통신학과(학사)
2008년 2월 : 한국항공대학교 항공전자공학과(석사)
2008년 ~ 현재 : 한국항공대학교 항공전자공학과(박사과정)
관심분야 : 고신뢰성 임베디드 시스템, 시뮬레이션

류 대 현 (Ryu, Dae-Hyun)
종신회원

1985년 2월 : 부산대학교 전자공학과 석사
1997년 2월 : 부산대학교 전자공학과 박사
1987년 3월~1998년 2월 : 한국전자통신연구원
1998년 3월~현재 : 한세대학교 IT 학부
관심분야 : 센서네트워크, 영상처리, 정보보호

나 종 화 (羅宗和)
종신회원

1985년 2월 : 서강대 전자공학과 학사
1988년 : Wayne State University 석사
1995년 : University of Arizona 박사
2005년 ~ 현재 : 한국항공대학교 항공전자공학과 교수
관심분야 : 고신뢰성 임베디드 시스템