# 항공용 임베디드 시스템을 위한 고장감내형 프로세서 설계와 오류주입을 통한 검증

# Fault Tolerant Processor Design for Aviation Embedded System and Verification through Fault Injection

이동우*, 고완진**, 나종화**

Dong-woo Lee*, Wan-jin Ko**, and Jong-wha Na**

## 요 약

본 논문은 고신뢰성 임베디드 시스템의 핵심 부품인 risc 프로세서에 forward 기반의 오류복원 기법을 적용한 fetch redundant risc(FRR) 프로세서와 backward 기반의 오류복원 기법을 적용한 redundancy execute risc(RER) 프로세서를 연구하였다. 제안된 프로세서의 고장감내 성능을 평가하기 위해서 base risc, FRR, RER 프로세서의 SystemC 모델을 제작하고 SystemC 기반 fault injection 기법을 이용하여 오류주입 시험을 수행하였다. 실험결과 세 프로세서의 고장률은 1-bit transient fault를 주입한 경우에는 고장률이 FRR 프로세서는 1%, RER 프로세서는 2.8%, base risc 프로세서는 8.9%로 확인되었으며, 1-bit permanent fault를 주입한 경우 FRR 프로세서는 4.3%, RER 프로세서는 6,5%, base RISC 프로세서는 41%로 확인되었다. 따라서 1-bit 오류가 발생하는 경우에는 FRR 프로세서가 가장 높은 신뢰성을 나타내는 것으로 판명되었다.

## Abstract

In this paper, we applied the forward and backward error recovery techniques to a reduced instruction set computer (risc) processor to develop two fault-tolerant processors, namely, fetch redundant risc (FRR) processor and a redundancy execute risc (RER) processor. To evaluate the fault-tolerance capability of three target processors, we developed the base risc processor, FRR processor, and RER processor in SystemC hardware description language. We performed fault injection experiment using the three SystemC processor models and the SystemC-based simulation fault injection technique. From the experiments, for the 1-bit transient fault, the failure rate of the FRR, RER, and base risc processor were 1%, 2.8%, and 8.9%, respectively. For the 1-bit permanent fault, the failure rate of the FRR, RER, and base risc processor were 4.3%, 6.5%, and 41%, respectively. As a result, for 1-bit fault, we found that the FRR processor is more reliable among three processors.

Key words : embedded system, fault injection, dependability

## I. 서 론

항공용 임베디드 시스템은 상용 임베디드 시스템

* 한국항공대학교 항공전자 및 정보통신공학부 (College of of electronics, Telecommunication & Computer Engineering,Korea Aviation University)
· 제1저자 (First Author) : 이동우
· 투고일자 : 2009년 12월 4일
· 심사(수정)일자 : 2009년 12월 7일 (수정일자 : 2010년 4월 23일)
· 게재일자 : 2010년 4월 30일

과는 달리 높은 신뢰성이 보장되어야 한다. 항공전자 시스템은 운항 중에 고장 수리가 힘들며, 사고발생 시에 대형 참사로 이어진다. 이러한 이유로 현재의 항공 임베디드 시스템은 신뢰성을 보장하기 위한 인증 시스템을 갖추고 있다. 항공용 임베디드 시스템의 신뢰도를 보장하기 위해, 소프트웨어의 경우 DO-178B, 하드웨어는 DO-254 인증 평가 또는 이에 준하는 인증시스템에 통과한 제품만을 사용하고 있다. 그러나 신뢰성 인증 평가는 설계단계에서 작성된 설계 규격에 의해서 파악된 내용을 대상으로 하며, 설계자가 인지하지 못한 설계규격에 대해서는 대책이 없는 것이 현실이다. 또한 항공기의 열악한 운용 환경은 항공 임베디드 시스템의 오류발생률을 증가시킨다. 따라서 항공 임베디드 시스템은 유사시 발생할 가능성이 있는 오류를 방지하고, 오류에 의한 피해를 최소화 하는 것이 중요하다.

항공용 임베디드 시스템의 설계자는 설계초기 단계부터 고장 감내 기법을 고려해야 한다. 고장 감내 기법은 오류가 발생할 경우 오류를 제거하거나 오류에 의한 피해를 최소화 하는 방법으로서 redundancy 기법을 많이 활용한다. Redundancy는 임베디드 시스템 설계과정에서 하드웨어 redundan

-cy, 소프트웨어 redundancy, 정보 redundancy, 시간 redundancy로 구현 할 수 있다. (1)하드웨어 redundancy는 동일한 모듈 N개를 더해주는 n-modular redundancy, spare 모듈을 추가하는 standby sparing 기법 등이 있다. (2)소프트웨어 redundancy는 여러 개발자가 서로 다른 소프트웨어 개발환경에서 동일한 규격(Specification)의 소프트웨어를 개발하는 n-version programming, 소프트웨어의 정합성을 따지는 consistency check 등이 있다. (3)시간 redundancy는 오류가 발생할 경우 이전 상태로 회기 하여, 재 수행하는 rollback 기법이 있다. (4)정보 redundancy에는 데이터에 미리 정해놓은 코드를 추가하여 redundancy를 적용하는 parity code, checksum등의 방법이 있다[1,2].

고장 감내 설계 기법은 시스템의 특성에 따라 다양하게 적용 가능하며, 시스템의 신뢰도 및 가용성 등을 결정한다. 하지만 시스템에 적용한 고장 감내 기법에 따른 신뢰도의 차이를 정량화 하는 것은 매우 어려운 것으로 분석 되었다.

본 논문에서는 고장 감내 기법에 대한 정량적인 신뢰도를 측정하기 위한 방법으로 오류주입 기법을 사용하였다. 오류주입 기법은 시스템에 임의적인 오류를 주입하고, 시스템의 고장 여부를 분석하여, 시스템의 고장률 및 신뢰도를 계산하는 방법이다[3]. 오류주입 기법은 오류를 주입하는 방법에 따라 saboteur, mutant, kernel base fault injection 방법이 있다[4,5]. 본 논문에서는 커널기반 오류주입 기법을 통해 시뮬레이션 모델에 오류를 주입하고, 임베디드 시스템의 고장률 및 신뢰도를 평가한다[5].

본 논문은 risc 프로세서를 기본 모델로 하여, 고장 감내 기법을 적용하한 fault tolerant risc processor(FTR)를 개발한다. Risc processor는 electronic system level (ESL) 설계방법을 적용하였으며, ELS 설계 방법에서 가장 널리 사용되는 SystemC 하드웨어 개발도구를 사용하였다[6]. 본 논문에서는 프로세서의 인출 부를 3중화한 forward 기반의 fetch redundancy risc(FRR)와 연산 부에 rollback 기법을 적용한 backward 기반의 redundancy-execute risc(RER)를 제안한다. 각 프로세서의 신뢰도는 오류주입 기법을 사용하여 평가하였으며, 실험결과 FRR가 오류에 가장 강인함을 확인 할 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서는 커널 기반 오류주입 기법을 설명한다. 3장에서는 기반 프로세서인 risc processor와, 고장감내 기법이 적용된 FRR, RER을 설명한다. 4장은 오류주입에 의한 프로세서 모델의 상태변화를 분석하고, 5장에서는 설계한 각 프로세서에 오류를 주입하고, 결과를 분석한다. 6장에서 결론을 맺는다.

## Ⅱ. SystemC Kernel-based Fault Injection

그림 1은 디자인 모델의 오류에 의한 영향을 평가하기 위한 오류주입 환경을 보여주고 있다. 오류주입 대상은 SystemC로 설계한 하드웨어 시뮬레이션 모델

과 소프트웨어로 구성된 임베디드 시스템이다. 수행 절차에 따라 (1) 설정, (2) 실행, (3) 평가의 3단계로 구성되어 오류주입 시뮬레이션을 수행한다.

### 2-1 Setup

SystemC kernel base fault injection은 설정단계에서 (1) SystemC 하드웨어 디자인 모델, (2) 테스트 벤치 소프트웨어, (3) 오류주입 실험 설정파일을 설정한다. SystemC 하드웨어 디자인 모델은 risc processor 시뮬레이션 모델을 사용하며, 소프트웨어는 risc processor에서 수행 가능한 테스트용 어셈블리 코드를 사용한다. 하드웨어 시뮬레이션 모델과 테스트 벤치 소프트웨어의 설정 후, 오류주입 시뮬레이션을 수행하기 위한 fault_attribute.c를 작성한다. 표 1은 오류주입 실험을 위한 속성들을 보여주고 있다. 오류속성은 오류발생위치, 오류 발생 시간, 오류 발생빈도, 오류유형으로 설정된다. 오류 발생 빈도는 일시적(transient), 영구적(permanent) 그리고 간헐적 오류로 설정 할 수 있다. 일시적 오류는 오류주입 시간에 단 한번 주입되며, 영구적 오류는 오류주입 시간 이후 지속적으로 같은 위치에 오류 값이 주입된다. 간헐적 오류는 오류 발생 이후에 일정 간격으로 반복적으로 발생하는 오류이다. 오류 발생위치와 시간은 시뮬레이션 커널의 준비단계에서 임의적으로 설정하거나, 사용자가 선택할 수 있다.

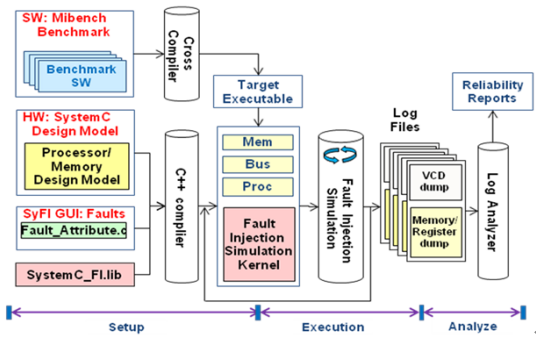


그림 1 SystemC 오류주입 환경
Fig. 1. SystemC fault injection environment

표. 1. 오류주입 시나리오 속성
Table. 1. Fault injection attribute

| Fault Attributes | Value |
|---|---|
| Fault Type | Transient, Permanent, Intermittent |
| Fault Time | Random or Deteministic |
| 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 Execution

하드웨어 시뮬레이션 파일에 바이너리 형태의 소프트웨어를 적재하고 시뮬레이션을 수행한다. 시뮬레이션 커널은 오류주입 설정 파일에 따라 오류를 주입한다. 통계적으로 유의한 실험결과를 산출하기 위해 반복적인 오류주입 실험을 수행한다. 오류주입 실험의 결과파일로 value change dump(VCD)을 출력한다. 그 외에 모델에 따라 각종 결과 파일을 추출 할 수 있다. 본 실험환경에서는 VCD 파일과 함께, ram 파일, 레지스터 파일을 분석을 위한 결과 파일로 추출하고 있다. 결과 파일은 log 분석기를 통해 오류에 의한 동작 특성의 변화를 세부적으로 분석한다.

### 2-3 Analyzer

SyFI 분석기는 시뮬레이션 수행으로 산출된 VCD 파일과 메모리 dump 파일을 통해 시스템의 고장 여부를 판단한다. 오류에 의한 시스템의 고장 유무를 판단하기 위해 오류를 주입하지 않는 golden run 시뮬레이션을 수행한다. Golden run 시뮬레이션을 통해 산출된 VCD와, 메모리 dump파일을 오류주입 시뮬레이션 수행 결과와 비교하여, 오류가 시스템에 미치는 영향을 분석한다.

## Ⅲ. 고장 감내형 하드웨어 설계

### 3-1 RISC Processor 설계

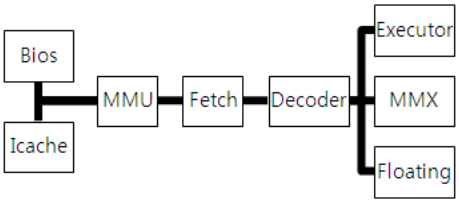


그림 2 RISC 프로세서 구성도
Fig. 2. RISC processor architecture

Base risc 프로세서는 Synopsys사에서 SystemC로 개발한 RISC(Reduced Instruction Set Computer)기반 프로세서 모델이다. 그림 2는 risc 프로세서의 구성도이다. Risc 프로세서의 동작 및 그에 따른 데이터부의 구성은 크게 3개의 단계로 나눌 수 있다. 1)명령어를 instruction cache에서 인출하여 memory management unit(MMU)를 통해 fetch로 불러오는 fetch부, 2) 전달된 명령어를 해석하여 데이터와, 제어신호를 만들어 주고, register를 조회하는 decoder부, 3) 실제 연산을 수행하고 결과 값을 산출하는 executer부로 구성된다. 소프트웨어는 assembly program으로 작성되며, assembler를 통해 기계어로 번역한다. 번역한 기계어 코드는 시뮬레이션 시작과 동시에 instruction cache에 로드(load)된다. 로드된 명령어 코드는 MMU을 거쳐 fetch로 전달된다. Fetch를 통해 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 수행 결과는 decoder로 보내져 register에 저장된다.

### 3-2 FETCH REDUNDANCY RISC (FRR)

Fetch redundancy risc(FRR)은 risc processor를 기본 구조로 하여, fetch부에 고장 감내 기법을 적용한 프로세서 모델이다. Fetch 모듈은 기능의 특성상 memory latency를 최소화 할 수 있는 고장 감내 기법을 적용하여, redundancy에 의한 오버헤드를 최소화 해야 한다. 즉 고장 감내 기법 적용에 따른 memory latency를 최소화함과 동시에 신뢰도를 향상 시킬 수 있는 기법을 적용해야 한다.

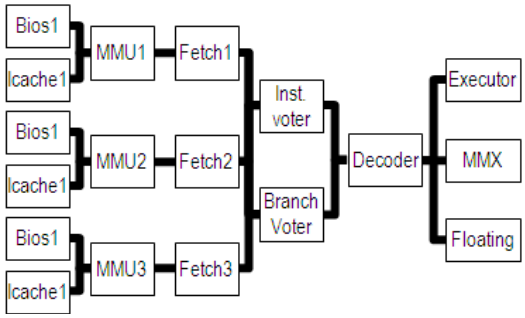


그림 3 FRR 프로세서 구성도
Fig. 3. FRR processor architecture

FRR은 이러한 조건을 만족하기 위해 fetch 부를 Triple Moduler Redundancy(TMR)로 설계하였다. 그림 3은 FRR모델의 구조도를 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 instruction voter와 branch voter를 통해 오류발생 여부를 검사하고, 정상 값을 출력하도록 한다. 인출부의 모듈을 삼중화 하였지만, 병렬적으로 처리되기 때문에 추가적인 클락 소모는 없다. 다만 각 voter의 오류발생 검사 동작에 의해 각 명령어에 따라 1 clock이 소모된다.

### 3-3 Redundancy Execute RISC(RER)

앞서 제시한 FRR은 fetch부에서 발생되는 오류를 검출하고 복구 할 수 있다. 그러나 decoder, executor 쪽에서 발생되는 오류를 검출 할 수 없는 단점이 있다. 이를 보완하기 위하여 decoder와 executor에 고장 감내형 기법을 적용할 필요성이 있다. Decoder와 executor는 fetch에 비하여 memory 접근에 의한 지연은 없으나 연산을 위한 하드웨어 사이즈가 크다. 따라서 RER에서는 하드웨어redundancy가 아닌 시간 redundancy를 통해 신뢰도를 향상시키는 방안을 제안한다. 그림 4는 설계한 RER의 구조를 보여주고 있다. 그림에서 보는 바와 같이 pc_control과 Exec voter가 추가되었다. Pc_control은 fetch로부터 받은 명령어를 일정 간격으로 2번씩 decoder에 전달한다.

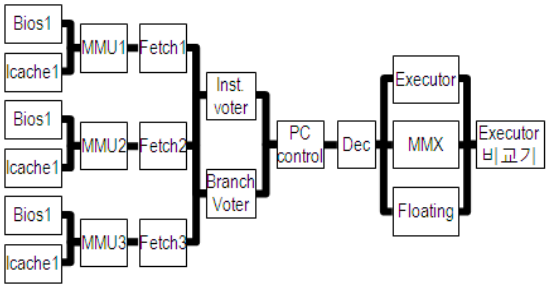


그림 4. RER 프로세서 구성도
Fig. 4. RER processor architecture

따라서 동일 명령어를 2번씩 처리하게 된다. Exec voter는 동일한 명령어의 두 결과를 비교하고 일치 한다면, register에 결과 값을 저장한다. 하지만 명령어 수행 중에 오류가 발생하여 결과가 동일하지 않을 경우, pc_control 오류 발생 신호를 인가하여, 이전 명령어를 다시 수행한다.

## Ⅳ. 오류주입 분석방법

그림 5는 오류 주입에 따른 시뮬레이션 모델의 상태 변화를 보여주고 있다. 그림에서 보는 바와 같이 오류주입에 의한 시뮬레이션 모델의 상태변화는 5가지의 경우로 분류 할 수 있다.

- case1: 주입된 오류 값이 masking 되거나, 고장감내 기법에 의해서 오류가 정정되어 안정적인 상태를 유지하고 있음을 의미한다.
- case2: 주입된 오류 값이 시스템에 지속적으로 유지되어 시스템의 운용에 위험요소로 작용하는 상태를 의미한다.
- case3: 주입된 오류 값이 시스템에 반영되었으나, 프로그램 수행도중 자연적으로 masking 되어, 안정적인 상태를 유지하는 경우이다.
- case4: 오류주입으로 시뮬레이션 모델 동작이 중지하는 경우이다.
- case5: 오류주입 값으로 인해 시뮬레이션 동작에 문제가 생기는 경우이다. 시뮬레이션 모델은 C++기반한 SystemC 로 작성되어 있다. 설계의 편의성을 위해 register는 배열로 잡혀 있다. Register 조회하기 위한 signal값에 오류가 발생할 경우, 배열 오버플로어가 발생하여, 정상적인 시뮬레이션을 수행하지 못하는 경우가 생긴다.

## Ⅴ. 실험결과 및 분석

표2는 고장 감내 프로세서에 오류주입 시뮬레이션을 수행한 결과이다. 실험 결과는 그림5의 case 분류에 따라 각각 분류 하였다. 표의 case1,3은 오류주입 이후에 시스템이 안전한 생태이며, case 2,4,5는 불안전하거나 시뮬레이션 동작이 멈춘 상태를 의미한다. 오류주입 실험은 프로세서 모델에 따라 일시적 오류(T)와 영구적 오류(P)를 각 500회씩 실험 하였다. 표에서 보는바와 같이 risc 프로세서에 비해 고장 감내형 기법이 적용된 FRR, RER의 안전성이 더 높음을 확인 할 수 있다. 이러한 차이는 일시적인 오류에 비해 영구적 오류를 주입했을 때 더 확실히 나타난다. RER는 FRR에 비해 고장감내형 기법을 더 적용했음에도 불구하고, 오류주입 실험을 통한 오류처리 능력은 더 떨어지는 것으로 나타났다.

RER가 FRR에 비해 오류처리율이 떨어지는 이유는 1) 반복 수행에 의한 오류검출은 영구적인 오류를 주입할 경우, 오류검출이 불가능 하며, 2) 명령어를 반복수행하고, 제어하기 위해 요구되는 추가적인 모듈과 제어모듈이 오류에 취약부로 작용하여, 오류처리율을 떨어트리는 것으로 보인다.

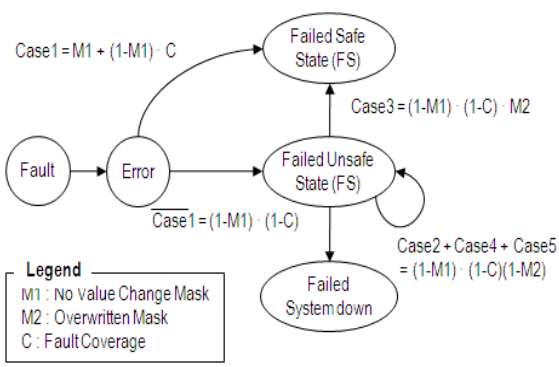


그림 5. 오류주입에 따른 시스템 상태 변화
Fig. 5. System state diagram after fault injection

표 2. 오류주입 실험결과
Table. 2. Fault injection experiment result

| | fault | case1 | case2 | case3 | case4 + case5 |
|---|---|---|---|---|---|
| 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% |

## VI. 결 론

본 연구는 다양한 기법을 적용한 고장감내형 프로세서를 설계하고, 이를 검증하기 위한 방법으로 오류주입 시뮬레이션을 수행 하였다. Risc 프로세서를 기본모델로 하여, 프로세서의 인출 부를 삼중화 한 FRR 프로세서, 실행부에 오류를 검출하기 위해 시간 redundancy를 적용한 RER 프로세서를 SystemC로 설계하였다. 커널기반 오류주입 툴을 사용하여, 오류주입 실험을 수행하고, 프로세서의 오류처리능력을 평가 하였다. 평가결과 프로세서의 failure rate는 프로세서와 오류에 유형에 따라 차이가 보임을 확인 할 수 있었다. 프로세서 failure rate(transient, permanent)는 각각 FRR(1%, 4.3%), RER(2.5%, 6.5%), base risc(8.9%, 41%)로 산출되었다. 이를 통해 FRR 프로세서가 가장 좋은 오류처리율을 갖고 있음을 확인 할 수 있었다.

이와 같이 고 신뢰성을 요구하는 항공용 임베디드 시스템을 설계할 경우 설계 시뮬레이션 모델을 대상으로 오류주입 시뮬레이션을 진행하여, 오류 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] Israel Koren and C. Mani Krishna, "Fault-Tolerant System", *Morgan Kufmann*, 2007

[3] Shubu Mukherjee, "Architecture Design for Soft Errors", *Morgan Kufmann*, 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, vol 26, no6,* pp50-60, 2009

[6] Open SystemC Initiative(OSCI) http://www.systemc.org

## 감사의 글

본 연구는 국토해양부 항공선진화사업의 연구비 지원(과제번호 #07항공-항행-03)에 의해 수행되었습니다.

이 동 우 (李東雨)

2008년 2월 : 한국항공대학교 항공전자공학과(석사)
2008년 3월~현재 : 한국항공대학교 항공전자공학과(박사과정)
관심분야 : SoC Design, Fault Tolerant Design & Simulation

고 완 진 (高完震)

2009년 2월 : 한국항공대학교 항공전자공학과(학사)
2009년 3월~현재 : 한국항공대학교 항공전자공학과(석사)
관심분야 : Fault Tolerant, Reliability

나 종 화 (羅宗和)

1985년 2월 : 서강대 전자공학과 졸
1988년 : Wayne State University석사
1995년 : University of Arizona 박사
2005년 ~ 현재 : 한국항공대학교 항공전자공학과 부교수
관심분야 : 컴퓨터 시스템