Journal of the Korea Academia-Industrial cooperation Society
Vol. 14, No. 10 pp. 5123-5129, 2013

http://dx.doi.org/10.5762/KAIS.2013.14.10.5123

국방용 임베디드 시스템의 고신뢰성 검증을 위한 JTAG 결함주입 방법론 연구

이학재[1*], 박장원[1]
[1]LIG넥스원

JTAG fault injection methodology for reliability verification of defense embedded systems

Hak-Jae Lee[1*] and Jang-Won Park[1]
[1]**LIGnex1**

요 약 본 논문에서는 국방용 임베디드 시스템의 신뢰도를 테스트 할 수 있는 JTAG fault injection 기법과 fault, error, failure에 대한 분류 기법들을 새롭게 제안하였다. JTAG fault injection 기법은 JTAG을 사용하여 실제 hardware에서 발생할 수 있는 fault를 software에서 인위적으로 유사하게 발생시킬 수 있다. 이 기법을 적용하여 시스템 취약도에 대한 정량적 분석이 가능하였다. 본 논문의 JTAG fault injection 실험은 통계적인 방법을 적용하였으며, 실제 H/W 결함과 유사한 결함을 메모리의 임의의 위치에 주입하였다. 결함주입 실험 결과는 기존의 fault, error, failure 분류 기법을 보완한 재분류 기법을 적용하여 분석하였다. 그 결과, 약 19%의 failure 탐지율 향상을 보였다. 이 실험결과는 시스템에서 발생할 수 있는 fault, error, failure의 체계적 분류, 프로세스 검증, 신뢰성 개선을 위한 데이터로 활용이 가능하다.

Abstract In this paper, it is proposed that JTAG fault injection environment and the results of the classification techniques that the reliability of embedded systems can be tested. As applying these, this is possible to quantitative analysis of vulnerable factor for system. The quantitative analysis for the degree of vulnerability of system is evaluated by faults errors, and failures classification schemes. When applying these schemes, it is possible to verify process and classify for fault that might occur in the system.

Key Words : JTAG, Fault, Error, Failure, Fault injection

1. 서론

Safety-critical 시스템의 수요 증가와 함께, 최근 반도체 공정의 발전은 Safety-critical 시스템이 고장을 일으키는 기술적 요인으로 작용하고 있다. 이러한 요인으로 인해 칩에 고장을 일으키는 새로운 문제들이 발생하였다. 특히, 예기치 않은 오류를 발생시키는 Soft Error에 대한 민감도가 크게 증가하였다[1]. 이러한 결함을 검출하기 위한 대표적인 기법으로 결함주입[3,4] 기법이 있다. 결함주입 기법은 시스템을 정확하게 검증할 수 있다는 장점이 있다. 하지만 인위적으로 주입한 결함이 실제 시스템에서 발생하는 결함과 얼마나 정확하게 대응하는지, 그리고 수많은 결함 중에 어떤 결함을 주입할 것인가를 결정하기 위해서는 많은 요소들이 고려되어야 한다. 또한, 현재의 결함주입 기법들(H/W 기반 결함주입 기법, S/W 기반 결함주입 기법, Simulation 기반 결함주입 기법)[2]은 시스템 설계 및 생산 비용을 증가시킨다. 따라서, 이러한 비용을 낮추기 위해서는 Safety-critical 시스템을 양산 전·후에 테스트하여야 한다. Safety -critical 시스템의 테스트를 위해서는 Fault 영향 분석을 하는 것이 중요하다. 특

*Corresponding Author : Hak-Jae Lee(LIG nex1)
Tel: +82-31-8026-4932 email: hakjae.lee@lignex1.com
Received August 30, 2013 Revised (1st September 26, 2013, 2nd October 1, 2013) Accepted October 10, 2013

히, 임베디드 시스템은 자원과 타이밍에 민감하므로 실제 시스템과 동일한 환경에서 결함 주입 테스트를 수행하여야 한다.

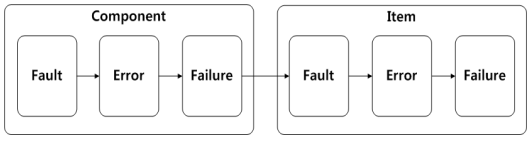
본 논문에서 제안하는 JTAG 기반 결함주입 기법은 국방용 임베디드 시스템 신뢰성 평가에 사용하는 기존의 결함주입 기법을 보완할 수 있고, H/W에서 가장 빈번하게 발생할 수 있는 결함들을 JTAG을 사용하여 유사하게 구현할 수 있다. 또한, 결함(fault), 오류(error), 고장(failure)에 대한 분류 기법을 바탕으로 실험 결과를 분석하여 결함(fault), 오류(error), 그리고 고장(failure)의 상관관계를 분석할 수 있다. JTAG 결함주입의 장점은 1) 결함주입 시 대상 H/W의 파손 위험성이 없고, 2) 실시간으로 칩 내부의 제어와 관찰이 가능하며, 3) 결함주입이 용이하고, 4) 신속/정확한 테스트가 가능하다. 그리고 5) 코드 수정과 같은 복잡한 작업 과정 불필요하다.

2. 본론

2.1 신뢰성 평가 방법 및 관련 연구

2.1.1 Fault, Error, Failure 정의

결함주입 기법을 적용하기 위해서는 먼저 fault(결함), error(오류), failure(고장)에 대하여 파악해야 한다. Fault란, H/W 또는 S/W 내의 불완전성 혹은 fault를 의미한다. Fault의 종류에는 1) H/W fault와 2) S/W fault로 구분할 수 있다. H/W fault는 Permanent fault, transient fault, intermittent fault 등의 물리적 fault를 나타낸다. S/W fault는 S/W의 정의된 특성과 일치하지 않는 S/W의 모든 행위를 나타낸다. Error는 fault에 의해 발생하는 비정상적, 부정확한 상태를 의미한다. Failure는 수행해야 할 임무를 수행하지 못함을 의미한다. Fig. 1은 fault, error, failure 확산을 나타내고 있다.



[Fig. 1] Progression of Fault, Error, and Failure[5]

2.1.2 Fault, Error, Failure 분류

Fault 분류 기법은 fault 유형을 분류 및 정리한 것이다. 대표적인 fault 분류 기법으로는 orthogonal defect classification (ODC)[6]과 fault emulation operator (FEO)가 있다[7]. ODC는 S/W fault 유형을 분류한 것이다. ODC fault 유형을 실제로 적용한 예로는 FEO가 있다. FEO는 ODC fault 유형 중 가장 빈번히 발생한 fault 유형 13가지만을 나열한 것이다. FEO를 사용하기 위해서는 fault emulation을 해야 한다. Fault emulation이란, 실제 fault와 거의 동일한 fault를 인위적으로 발생시키는 것이다. 여기서 실제 fault는 H/W 또는 S/W에서 실제로 발생하는 fault이다. 인위적으로 발생시킨 fault는 실제 fault와 유사한 fault를 나타낸다. Fault emulation의 대표적인 기법으로는 Generic S/W fault injection technique (G-SWFIT)가 있다[8].

대표적인 Error 분류기법으로는 instruction level error (ILE)[9]가 있다. ILE는 마이크로프로세서의 컨트롤 로직에서 fault로 인해 발생하는 error 유형을 나타낸다. ILE로 분류한 error 유형을 적용하면 어셈블리 명령어 실행 시 발생하는 error가 시스템에 어떠한 영향을 주는지 파악할 수 있다. ILE는 error 유형을 5가지 Group과 13가지 type으로 구분되어 있다.

Failure 분류는 장치에서 발생하는 서비스에 의해 정의된다. 본 논문에서 적용하는 failure 분류는 Data violation, Time out, Complete with delay, Error without effect의 4가지 유형으로 나타낼 수 있다.

2.2 Fault, Error, Failure에 대한 재분류

Safety-critical 시스템은 단 하나의 fault가 발생하더라도 그 fault가 치명적이라면 생명과 재산에 큰 피해가 발생한다. 즉, 치명적인 fault는 발생 빈도수가 낮더라도 시스템의 신뢰성을 평가하는데 매우 중요한 요소가 될 수 있다. Safety-critical 시스템에서는 fault의 빈도수로 분류한 FEO를 적용하는 것은 적합하지 않다는 것을 의미한다. FEO는 실험 환경에 따라서 재분류 될 수 있다.

2.2.1 Fault 재분류

본 논문의 실험에서는 임베디드 보드에서 실제로 발생한 fault를 ODC 기반으로 재분류하였다. 본 논문의 사전 실험을 바탕으로 재분류한 fault 유형은 Table 1과 같다. 재분류한 FEO 테이블은 Missing fault 유형이 발생하지 않았다. 그 이유는 실제 random 결함주입에서는 어셈블리 명령어가 Missing이 되는 경우가 발생하지 않았기 때문이다.

2.2.2 Error 재분류

기존 ILE를 적용한 경우에는 error 구분이 분명하지 않은 경우가 발생하였다. Error 구분이 분명하지 않은 경우는 Table 2와 같이 error 분류 기법을 보완하여 구분이 모호한 error를 추가로 파악하였다. 그리고 이 fault들의

전파 과정을 추적하여 시스템의 취약 부분을 보다 명확하게 찾을 수 있었다.

[Table 1] Reclassification of ODC - FEO table

ODC categories	Nature	ODC fault type
Assignment	Missing	MVIV, MVAV, MVAE
	Wrong	WVAV, WVAL(added)
Checking	Missing	MIA, MLAC, MLOC
	Wrong	WAEC(added), WLEC(added)
Interface	Missing	MRS
	Wrong	WPFV, WAEP
Algorithm	Missing	MFC, MIFS, MIEB
	Wrong	WFC(added)

[Table 2] ILE Groups & Types

Group	Type
Group 1 : Operation Errors	1, 2
Group 2 : Operand Errors	3, 4, 5, 6
Group 3 : Execution Errors	7, 8
Group 4 : Timing Errors	9, 10, 11, 12
Group 5 : Order Errors	13
Group 6 : Operation & Operand Errors	14(added)

2.2.3 Failure 재분류

Table 3은 failure 유형을 Data violation, Time out, Complete with delay, Error without effect, Exception의 재분류된 5가지 유형으로 나타낸다. 새로 추가된 Exception은 예상하지 못한 예외 상황이 발생 경우이다.

[Table 3] Failure type[9]

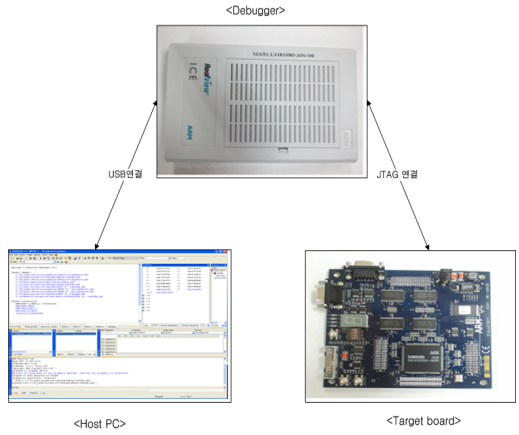
Failure type	Description
Data violation	The program terminates successfully. However, the memory content is different from that of the golden run.
Time out	The program does not complete in an expected time.
Complete with delay	The program completes with delay in the expected time.
Error without effect	In spite of the errors on the interface, the program terminates correctly.
Exception (added)	During the operation of computer systems, unexpected abnormal condition occurs and the program is being performed under the influence.

2.3 JTAG 결함주입 실험 및 분석 방법

결함주입은 target board의 메모리를 대상으로 하여 수행하였다. 실험은 메모리의 특정 주소 공간을 결함주입 구간으로 선정하여 진행하였다. 결함주입은 결함주입 대상 주소의 단일 비트(single bit)에 일시적인 fault(transient fault)를 주입하였다. 이 기법은 메모리의 결함주입 대상 주소인 32bit값 중 1bit만을 변경하는 기법이다. 이 후, 다음 번 결함주입 차례에서는 새로운 fault를 반복 주입하였다.

2.3.1 JTAG 결함주입 환경 및 구성요소

JTAG 결함주입 환경은 Fig. 2과 같이 Host PC, debugger, target board로 구성하였다. Host PC는 결함주입 script 파일을 로드하여 결함주입 수행 및 실험 결과 모니터 기능을 담당한다. Debugger는 target board를 감시하면서 동시에 Host PC에서 결함주입 수행 명령을 받아 JTAG 결함주입을 수행한다. 그리고 생성된 결함주입 code를 Host PC에서 target board로 다운로드 한다. Target board는 debugger에서 데이터를 다운로드 받아 연산 후, 연산 결과를 다시 debugger로 전송한다.



[Fig. 2] JTAG fault injection environment

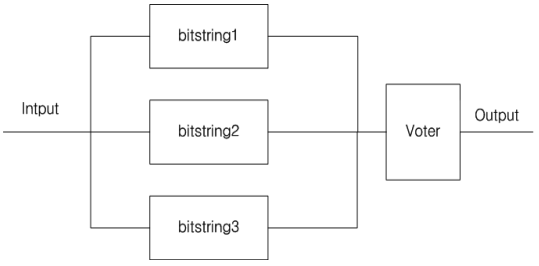
2.3.2 JTAG 결함주입 실험 절차 및 대상

JTAG 결함주입 절차는, 먼저 H/W를 초기화하여 결함주입 환경을 세팅한다. 그리고 Host PC에서 결함주입 대상 파일을 로드한다. JTAG 결함주입 실험 구간을 설정하여 결함주입 값을 정한다. 다음, 실험 구간에 fault을 주입하고 프로그램을 수행한다. 결함주입 수행 결과를 확인하고 실험을 완료한다. 이상의 결함주입 수행 절차를 반복하여 결과를 수집하고 fault, error, failure 분류 기법을 적용하여 실험 결과를 분석한다.

수행할 결함주입 대상 프로그램은 Mibench[10]의 bitstrng(bitcount), rad2deg(basicmath), pbmsrch (stringsearch)이다. 본 실험은 normal과 3-version을 대상으로 실험한다. Normal program은 고장 감내 기법을 적용하지 않은 원본 프로그램이다. Fig. 3은 bitstrng의 normal program 구조도를 나타내고 있다.

[Fig. 3] Normal program structure of bitstrng

N-version program은 하드웨어의 NMR(N-Modular Redundancy)과 유사한 개념의 결함허용 기법이다[11]. N-version program의 기본 개념은 소프트웨어 모듈을 N번 설계하고 코드화하는 것이고 이 모듈들에 의해 생성된 N개의 결과를 비교하는 것이다. 본 논문에서 사용하는 test S/W의 N-version은 3-version programming을 적용하였다. 3-version programming은 H/W의 TMR과 유사하게 소프트웨어 모듈을 3개 설계하고 코드화한 것이다. Fig. 4는 bitstrng의 3-version program 구조도를 나타내고 있다.

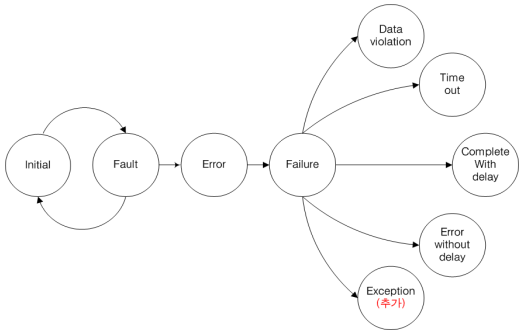


[Fig. 4] 3-version program structure of bitstrng

Normal program과 N-version program의 결함주입 수행은 JTAG 결함주입 절차를 따른다. JTAG 결함주입 수행 결과는 fault, error, failure에 대한 기존 분류와 재분류 기준을 적용하여 실험 결과 분석에 활용한다.

2.3.3 JTAG 결함주입 결과 분류

결함주입 실험 결과와 정상 파일을 비교하여 결함주입 실험 결과를 분류한다. Fig. 5는 결함주입 결과 유형 분류 다이어그램이다. Fault 결과 유형 분류 다이어그램에서 시스템의 상태는 1) Initial 2) Fault 3) Error 4) Failure의 4가지 상태로 천이 된다.



[Fig. 5] Classification results diagram[9]

2.3.4 신뢰도 측정 및 결함 횟수 산정

결함주입 대상 프로세서들과 결함주입 환경이 준비되면 각 프로세서에 결함주입 실험을 수행하여 신뢰도를 측정하기 위하여 architectural vulnerability factor (AVF)를 구한다[12]. 본 논문에서는 통계적인 결함주입을 적용한 AVF 측정법을 사용한다. [Form. 2]는 AVF를 나타내고 있다[12].

$$\text{AVF} = \frac{ACE\text{로 판정된 실험 횟수}}{\text{결함 주입 실험 총 횟수}} \qquad \text{[Form. 2]}$$

신뢰성 있는 데이터를 산출하기 위해 충분한 횟수의 결함주입 시뮬레이션 실험이 필요하다. [Form. 3][13]은 결함주입 실험에 필요한 실험 횟수를 산출하는 식이다.

$$n = \frac{4z_{\alpha/2}^2 \times \hat{p}(1-\hat{p})}{w^2} \qquad \text{[Form. 3]}$$

Table 4는 Test S/W인 bitstrng의 사전 실험 결과를 나타내고 있다. 사전 실험은 Mibench의 bitstrng, rad2deg, pbmsrch 의 각각에 대하여 normal program과 3-ver. program으로 구분하여 수행하였다.

[Table 4] Number of experiments required 90% reliability

Test S/W	Fault Model	normal program abnormal results/ effective fault	3-ver. program abnormal results/ effective fault
bitcount (bitstring)	st-0	95/104 (2,874)	105/291 (154)
	st-1	160/193 (1,320)	186/612 (119)
rad2deg (basicmath)	st-0	70/88 (1,059)	232/378 (433)
	st-1	125/194 (494)	334/552 (418)
pbmsrch (stringsearch)	st-0	93/104 (2,302)	76/278 (103)
	st-1	162/201 (1,131)	166/576 (111)

Table 5은 failure rate에 대한 test S/W의 사전 실험 결과를 나타내고 있다. Table 5을 살펴보면, rad2deg에 대한 normal program과 3-version program failure rate가 차이가 크지 않은 것을 확인 할 수 있다. 그 이유는 Time out이나 Exception이 발생하여 출력 데이터를 비교하기 전에 시스템 failure가 발생한 경우가 많기 때문이다. 따라서 3-version program이더라도 효과가 적은 것을 확인할 수 있다.

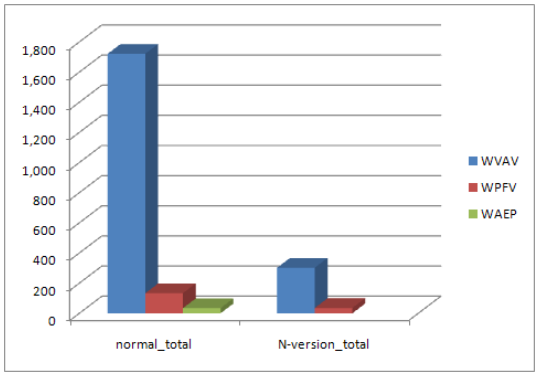
[Table 5] Pre-test result of Test S/W

Test S/W	Fault model	Error without effect	Data violation	Time out	Exception (added)	failure rate (%)
bitstrng (normal)	st-0	9	54	21	20	91.3
	st-1	33	83	46	31	82.9
bitstrng (3-ver.)	st-0	186	21	42	42	36.4
	st-1	426	21	111	54	30.4
rad2deg (normal)	st-0	18	33	4	33	79.5
	st-1	69	45	15	65	64.4
rad2deg (3-ver.)	st-0	146	86	38	108	61.4
	st-1	218	64	46	224	60.5
pbmsrch (normal)	st-0	11	44	18	31	89.4
	st-1	39	68	49	45	80.5
pbmsrch (3-ver.)	st-0	202	32	12	32	27.3
	st-1	410	84	42	40	28.8
Total		1,767	635	444	725	41.1

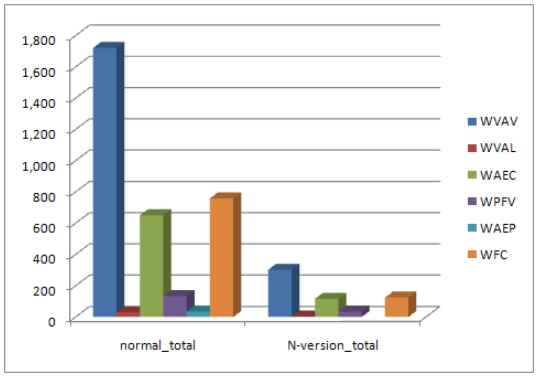
2.4 결함주입 실험 결과 분석

2.4.1 normal program과 3-version program 비교

2.4.1.1 Fault에 대한 기존 분류와 재분류 비교



[Fig. 6] Normal and 3-ver. results of existing fault classification

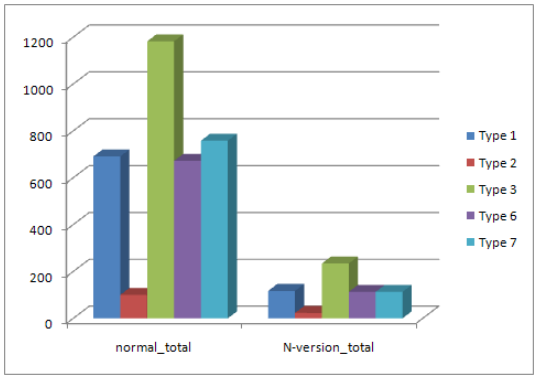


[Fig. 7] Normal and 3-ver. results of fault reclassification

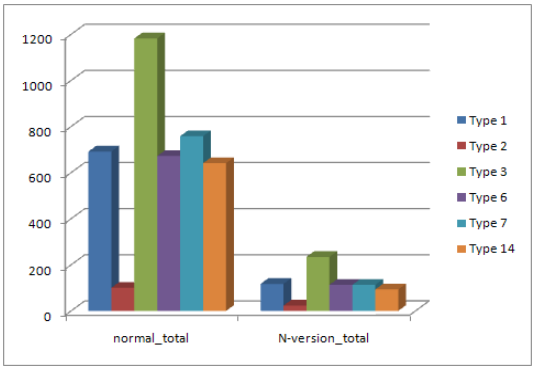
Fig. 6을 살펴보면 normal과 3-version 모두 변수에 잘못된 값이 할당되는 결함 유형(WVAV)이 가장 높게 발생하였다. Fig. 7에서는 normal과 3-version 모두 변수에 잘못된 값이 할당되는 결함 유형(WVAV)이 가장 높게 나타났다. 그리고 기존에는 없던 “분기 조건에서 산술식이 잘못된 결함 유형(WAEC)”이 다음으로 높은 비율을 나타내고 있다.

2.4.1.2 Error에 대한 기존 분류와 재분류 비교

Fig. 8을 살펴보면 normal과 3-version 모두 type3(incorrect register addressed)가 가장 높게 발생하였다. Fig. 9에서는 normal과 3-version의 type14(incorrect operation & operand code used)는 각각 15.9%, 13.4%를 나타내고 있다.



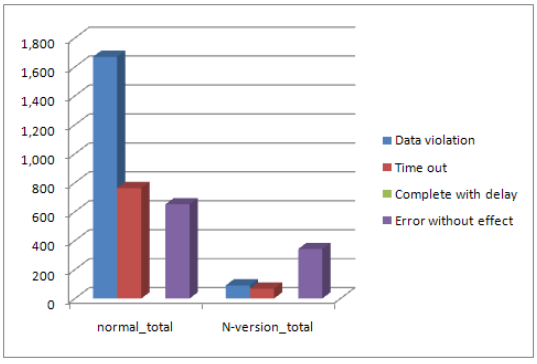
[Fig. 8] Normal and 3-ver. results of existing error classification



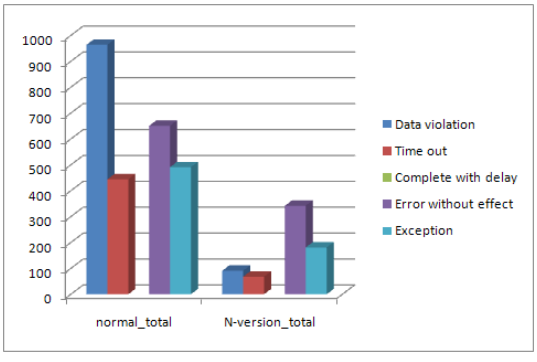
[Fig. 9] Normal and 3-ver. results of error reclassification

2.4.1.3 Failure에 대한 기존 분류와 재분류 비교

Fig. 10을 살펴보면 normal은 Data violation이 가장 높게 나왔다. 3-version은 Error without effect가 가장 높은 비율을 나타낸다. Fig. 11에서는 normal과 3-version의 exception은 각각 19.3%, 26.5%를 나타내고 있다.



[Fig. 10] Normal and 3-ver. results of existing failure classification



[Fig. 11] Normal and 3-ver. results of failure reclassification

2.4.2 Test S/W의 본 실험 결과

본 실험에서는 3가지 test S/W인 bitstring, rad2deg, pbmsrch에 결함주입을 수행하고 각각의 test S/W를 normal과 3-ver.로 재구분하여 실험하였다. Fault, error, failure에 대한 기존 분류와 재분류를 비교/분석하였다.

[Table 6] Results of test S/W

Test S/W	Fault model	Error without effect	Data viola -tion	Time out	Excep -tion (added)	failure rate (%)
bitstrng (normal)	st-0	84	512	197	191	91.5
	st-1	144	368	201	136	83
bitstrng (3-ver.)	st-0	30	2	9	11	42.3
	st-1	66	5	13	6	27
rad2deg (normal)	st-0	68	118	15	119	78.8
	st-1	119	77	25	110	64
rad2deg (3-ver.)	st-0	69	41	18	52	61.7
	st-1	105	30	21	101	40.9
pbmsrch (normal)	st-0	85	333	143	237	89.3
	st-1	151	262	182	166	80.2
pbmsrch (3-ver.)	st-0	24	2	1	7	29.4
	st-1	48	11	6	4	30.4
Total		993	1,761	831	1140	79

Table 6을 보면, 총 5981개의 fault 대상에 대하여 기존의 failure 분류를 적용한 경우는 60%(3585개)의 failure를 탐지율을 보였으며, failure 재분류를 적용한 경우는 79%(4725개)의 탐지율을 나타냈다.

3. 결론

본 논문에서는 국방용 임베디드 시스템의 신뢰성 검증에 사용할 수 있는 JTAG 기반 결함주입 기법을 제안하였다. JTAG 기반 결함주입 기법은 디버거를 사용하여 target 시스템에 영향을 주지 않고 경제적으로 디버깅할 수 있다. JTAG 결함주입 실험을 분석한 결과, 기존의 fault, error, failure 분류보다 이를 보완한 재분류 기법을 적용한 경우가 약 19% 정도 높은 failure 탐지율을 보였다. 또한, 기존 fault, error, failure 분류를 보완한 기법들을 적용하여 시스템에서 어떤 부분이 fault에 취약하고 어떤 부분이 fault에 강인한지 확인하였다. 그리고 fault 취약구간을 찾아 어떤 요인이 failure rate를 높이는 원인인지 파악할 수 있었다.

References

[1] P.Shivakumar, M.Kistler, S.W.Keckler, D.Burger, and L.Alvisi, “Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic”, in International Conference on Dependable systems and Networks, pp.389-398, June 2002.

[2] Madeira.H, Rela, M., Moreira, F,. and Silva, J., "RIFLE : A General Purpose Pin-level Fault Injector", 1st european Dependable Computing Conf, pp199-216, 1994.

[3] Yangyang Yu, Barry W. Johnson, “Fault Injection Techniques”, Kluwer Academic Publisher, pp7-39, 2003.

[4] K. K. Goswami, R. K. Iyer. "DEPEND : A Simulation-Based environment for system Level Dependability Analysis." IEEE Transactions on Computers, vol.46 1997, pp.60-74.
DOI: http://dx.doi.org/10.1109/12.559803

[5] Christian Esposito, "Hands on the ISO 26262 Standard", pp168-174, 2010.

[6] R. Chillarege, “Orthogonal Defect Classification”, Ch. 9 of “Handbook of S/W Reliability Engineering”, M. Lyu Ed., IEEE Computer Society Press, McGraw-Hill, 1995.

[7] J. Duraes, H. Madeira, "Emulation of S/W Faults by Educated Mutations at Machine-Code Level", Proceedings of the Thirteenth IEEE International Symposium on S/W Reliability Engineering, ISSRE’02, November 2002, Annapolis MD, USA.

[8] Ang Jin, Jian-hui Jiang, "Fault Injection Scheme for Embedded systems at Machine Code Level and Verification", 2009 15th IEEE Pacific Rim International Symposium on Dependable Computing.
DOI: http://dx.doi.org/10.1109/PRDC.2009.68

[9] Michail Maniatakos, "Instruction-Level Impact Analysis of Low-Level Faults in a Modern Microprocessor Controller", IEEE Transactions on Computers, Vol. 60, No. 9 September 2011.
DOI: http://dx.doi.org/10.1109/TC.2010.60

[10] Mibench, http://www.eecs.umich.edu/mibench/

[11] K.J.Heo, "A Study for N-version Programming reliability Model Using Neural Net", Kyung-nam Univ., 1996.

[12] X.Li, S.V.Adve, P.Bose, and J,A,Rivers, “Architecture-Level Soft Error Analysis: Examining the Limits of Common Assumptions”, in International Conference on Dependable systems and Networks(DSN), pp.266-275, 2007.

[13] Shubu Mukherjee, “Architecture Design for Soft Errors”, pp.146-150, Morgan Kaufmann Publishers, 2007.

이 학 재(Hak-Jae Lee) **[정회원]**

- 2012년 2월 : 한국항공대학교 일반대학원 항공전자공학과 (항공전자공학석사)
- 2012년 3월 ~ 현재 : LIG넥스원 연구원

<관심분야>
항공전자, 신뢰성

박 장 원(Jang-Won Park) **[정회원]**

- 2013년 2월 : 숭실대학교 공과대학 -산업정보시스템공학과 (산업정보시스템공학사)
- 2013년 3월 ~ 현재 : LIG넥스원 연구원

<관심분야>
신뢰성, 정보통신, 인간공학,