고신뢰성 ARM 프로세서 설계 및 오류주입 시뮬레이션을 통한 검증에 대한 연구

이동우*, 고완진**, 김병영**, 나종화**
*한국항공대학교 항공전자 및 정보통신공학부
dongwoo81@kau.ac.kr

A Study on Tools for Agent System Development

Dong-Woo Lee*, Wan-Jin Ko**, Byeong-Young Kim**, Jong-Wha Na**

요 약

반도체 기술에 발달로 SoC, MPSoC와 같은 기술이 주목을 받고 있다. 이와 함께 soft error의 증가, 설계복잡도와 같은 문제점이 나타나고 있다. 본 논문은 FT_ARM(Fault Tolerant ARM)을 설계하여, soft error에 대응 하고자한다. 또한 오류주입 시뮬레이션을 통해 설계한 FT_ARM의 성능을 비교한다.

1. 서론

반도체 기술의 발달로 고도의 집적화를 통한 하드웨어 모듈 간 통합 시스템의 개발이 활발히 진행되고 있다. Reconfigurable SoC, MPSoC(Multi-Processor System on Chip)로 대변되는 이러한 변화는, 시스템의 다기능 및 소형화, 저전력과 같은 소비자의 요구를 충족하게 되었다. 그러나 다기능 구현은 설계복잡도가 기하급수적으로 증가하는 원인이 되었다[1]. 또한 고속화에 따른 칩 내에 결함 허용범위(noise margins) 감소로 오류가 증가하였으며[2], 반도체 집적도 증가에 의한 반도체 칩의 민감도가 증가[2]하였다. 이는 반도체 기술 발전에 따른 변화 속에 해결해야 할 문제점으로 제기 되었다.

이와 같이 증가하는 오류를 극복하기 위한 다양한 연구가 진행되고 있으며, 대표적인 방법으로 고장감내형 시스템을 설계하는 것이다[3][4][5]. 고장감내형 시스템은 시스템에 오류가 발생할 경우 이를 적절히 처리하여, 정상동작을 유지할 수 있도록, 오류검출, 고립, 회복하는 메커니즘을 포함한 시스템이다. 본 논문에서는 고장감내형 ARM 프로세서를 제안하고, 오류주입 시뮬레이션을 통해 제안한 고장감내형 ARM프로세서의 성능을 평가 하고자 한다.

본 연구는 먼저, ARM7 기반의 프로세서 모델을 SystemC로 설계하였다. 이를 바탕으로 TMR(Triple Modular Redundancy)기법을 적용한 Module Level TMR ARM과 Checkpoint 기법을 적용한 Checkpoint ARM을 설계하였다. 설계한 모듈의 오류처리 능력을 검사하기 위해 오류주입 시뮬레이션을 수행하였다. 오류주입 시뮬레이션을 통해 설계된 프로세스 모델에 대한 오류처리 능력을 검증 할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 기존에 연구된 오류회복 기법과 시스템을 설명한다. 3장에서는 설계한 ARM과 고장감내형 기법을 적용한 Module Level TMR_ARM, Checkpoint ARM을 설명한다. 4장에서는 설계한 프로세서별 오류주입 시뮬레이션 결과를 설명하고, 5장에서 결론을 맺는다.

2. 오류회복 기법

1)Forward Error Recovery

시스템에서 오류가 발생 할 경우 프로세스의 진행에 지장이 없이 오류를 보정하는 기법을 통틀어 Forward Error Recovery 기법이라 한다. 이와 같은 기법은 기존 데이터에 오류를 검출, 보정 할 수 있는 정보를 추가(redundant information)한다. 정보 데이터와 오류검출용 추가데이터를 분석하여, 오류 발생 여부를 검출하고, 오류를 보정한다. 또한 ALU, memory, 기타 logic 등으로 구성된 컴퓨터 시스템의 경우 redundant copies와 voter(비교기)를 통해, 각 모듈의 처리결과를 비교하여, 오류를 검출하고 보정하여, 시스템을 진행 시킨다. Forward Error Recovery 기반 시스템으로는 Fail-over System, DMR system, TMR system 등이 있다[7].

2)Backward Error Recovery

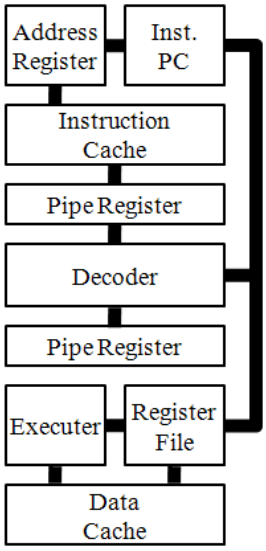
Backward Error Recovery 기법은 오류가 발생하면, 시스템의 상태가 이전상태로 회기하여, 오류를 보정하는 기법이다. Forward error recovery 기법에 비해 적은 하드웨어 자원을 요구하지만, 일정 시간마다 시스템의 중간 상태를 저장해야 한다. 이를 checkpoint라고 한다. Backward Error Recovery는 두 가지 기법으로 분류할 수 있다. 1) 정확한 오류 검출이 요구되는 기법으로 Fujitsu SPARC 64 V, IBM Z-Series, SRTR(Simultaneous and Recundantyly Threaded Processor with Recovery),

CRTR(Chip-Level Redundantly Threaded Processor with Recovery)과 같은 기법이 있다. 2) 확률적인 fault detection이 요구되는 기법으로 Exposure Reduction via pipeline Squash, Fault screening with pipeline squash and re-execution 기법이 있다[7].

3. FT_Arm(Fault Tolerant ARM) 설계

1) ARM 설계

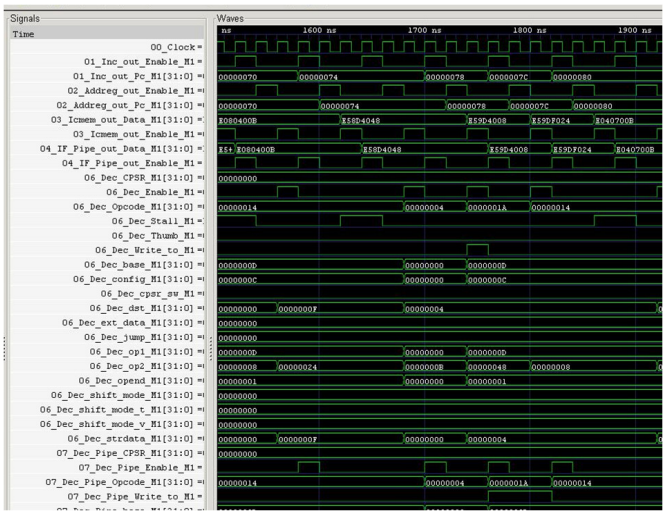
본 연구에서는 ARM cpu을 기본 프로세서로 하고 있다. ARM은 1980년대 스텐포드 대학과 버클리 대학에서 개발한 RICS 구조의 프로세서이다. ARM은 load-store 구조, fixed-length 32bit 명령어, 3-address 명령어 형식등의 특징을 가지고 있다. 보다 자세한 사항은 참고문헌[6]을 참고하기 바란다. 본 연구에서는 ARM 7을 기반으로 , SystemC로 ARM 을 설계하였다. 그림 1은 설계한 arm의 구성도 이다.



(그림 1) ARM 구성도‘

- Increase PC(Inc) : Decoder의 제어 신호에 따라, Register File에서 받은 PC(Program Count)값을 증가시켜 Address Register와 Register File로 전달한다.
- Address Register : 전송받은 PC값을 저장하고, Instruction cache로 주소와 제어신호를 전송한다.
- Instruction Cache : Address Register로부터 주소를 넘겨받아 해당 번지의 명령어를 Instruction Fetch Pipeline Register로 보낸다.
- Pipe Register : 파이프라인 동작을 위한 모듈, 각 단계별 실행결과 데이터를 저장하고, Enable 신호가 입력되면, 파이프라인의 다음 단으로 데이터를 전송한다.
- Decoder : Instruction Fetch 로부터 데이터를 인가받고, 이를 근거로 제어신호와 데이터 신로로 분리하여, Pipe Register로 전송한다.
- Register File : Register file은 3가지 동작을 수행한다. 1) Inc로부터 PC 값을 입력받아, 저장하고, 다시 Inc로 Feed back 한다. 2) 분기명령을 수행하기 위해서 Register file은 CPSR 값을 디코더로 전송하여, 해당 명령어의 실행 여부 판단한다. 3) 디코더로부터 데이터 요청을 받으면, 인가 받은 주소값에 해당하는 번지의 데이터를 Executor 로 전송한다. 3) Executer 혹은 Data cache 로부터 데이터를 전송받으면 내부에 기록한다.
- ALU : Decoder에서 생성된 제어신호와 Register File에서 전송된 데이터 신호를 전송받아, 제어신호에 따른 연산을 수행한다. 연산자에 따라 연산결과를 Register File 또는 Data Cache에 기록한다.
- Data Cache : Executer를 통해 전송받은 신호에 따라 Data read, write 동작을 수행한다. 데이터 읽기 동작에 의한 데이터는 Register File로 전송된다.

이와 같이 작성된 arm 모델은 ADS[8]로 컴파일된 arm용 프로그램 파일을 입력받아 프로그램을 실행한다. 그림 2 은 실제 JPG program을 수행하고, 결과 파일을 wave form 형식으로 출력한 결과이다.

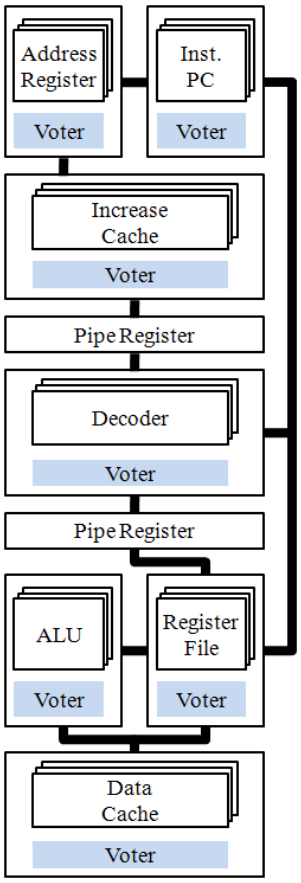


(그림 2) ARM 실행 결과

2) Module Level TMR ARM 설계

TMR ARM은 micro-architecture 레벨에서의 삼중화 구조와 voter로 이루어져 있다. 그림 3은 TMR ARM의 전체 구조도를 보여주고 있다. 그림에서 보는바와 같이 전체 구조는 앞서 소개한 ARM 구조와 동일하다. 단 각 모듈은 3개씩 한 세트를 이루고 있으며, 각 세트 별로 voter가 추가된다. 이와 같은 구조로 인해 각각의 모듈은 동일한 입력 값을 받고, 동일한 작업을 수행하게 된다.

voter는 3개의 모듈의 수행 결과를 입력받고, 비교한다. 만약 오류가 발생하지 않는다면, 3개의 모듈은 모두 동일한 출력 값을 산출하게 된다. 그러나 오류가 발생하면, 오류가 발생한 모듈은 다른 두 모듈과는 다른 출력결과를

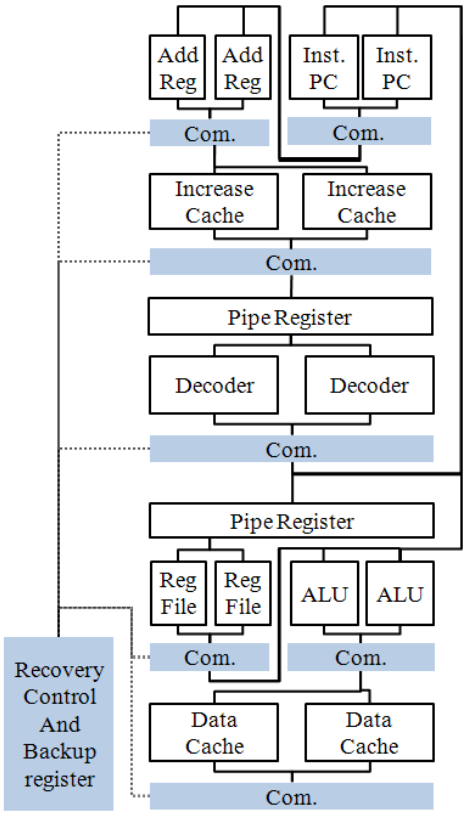


(그림 3) TMR ARM 구성도

voter에 전달할 수 있다. 이때 voter는 출력결과가 동일하지 않는 모듈의 입력 값을 무시한다. 두 개 이상의 모듈에서 동일한 출력결과가 입력된다면, 오류가 없다고 판단하여, 다음모듈의 입력 값으로 인가한다. 이와 같은 방법에 의해 시간지연 없이 오류를 제거한다.

3) Checkpoint ARM 설계

각각의 모듈은 삼중화 했던 TMR과 달리 Checkpoint ARM은 이중화 구조를 가지고 있다. 또한 각 모듈별로 비교기를 가지고 있다. 각각의 모듈이 두 개로 구성되어 있으므로, 오류 발생시 어떤 모듈이 오류를 발생했는지 알 수가 없으며, 단지 비교기를 통해 각각의 모듈의 값이 서로 상이할 경우 해당 모듈에서 오류가 발생 했다는 것만을 알 수 있다. 따라서 Recovery Controller와 Backup Register를 두어 주기적으로 프로세서의 상태 값을 저장하며, 오류 발생시 이전 상태로 rollback하도록 설계 하였다. 그림 4는 이와 같은 Checkpoint ARM 의 구성도를 보여주고 있다. Checkpoint ARM은 TMR ARM 보다는 더 적은 자원을 소모하는 장점이 있지만, 오류 발생시 rollback으로 인해 시간지연이 있을 수 있다는 단점이 있다. 또한 같은 오류가 반복되는 Permanent fault의 경우, 반복되는 불일치에 의해 checkpoint 로의 rollback과 retry가 반복되는 단점이 있다.



(그림 4) Checkpoint ARM 구성도

프로그램		ARM ① Fault Active ② System Failure ③ Fault Recovery			TMR ARM ① Fault Active ② System Failure ③ Fault Recovery			Checkpoint ARM ① Fault Active ② System Failure ③ Fault Recovery		
		①	②	③	①	②	③	①	②	③
JPG Program	Transient 1bit 0	654	400	254 (38.8%)	612	20	592 (96.7%)	593	0	593 (100%)
	Transient 1bit 1	1417	708	709 (50%)	1388	21	1367 (98.4%)	1388	23	1365 (98.3%)
	Permanent 1bit 0	1384	1177	207 (14.9%)	1402	43	1359 (96.9%)	1471	1471	0 (0%)
	Permanent 1bit 1	2000	1608	392 (19.6%)	2000	47	1953 (97.6%)	2000	1991	9 (0.4%)
Telcom Program	Transient 1bit 0	593	125	470 (78.9%)	566	5	561 (99.1%)	593	0	593 (100%)
	Transient 1bit 1	1400	231	1169 (83.5%)	1439	2	1437 (99.8%)	1388	23	1365 (98.3%)
	Permanent 1bit 0	1318	904	414 (31.4%)	1239	23	1216 (98.1%)	1471	1471	0 (0%)
	Permanent 1bit 1	2000	1817	183 (9.1%)	2000	77	1923 (96.1%)	2000	1991	9 (0.4%)

<표 1> FT ARM 오류주입 실험결과

4. 실험 결과

표 1는 오류주입 실험결과를 보여주고 있다. 오류주입은 SystemC 모델 오류주입 환경인 SyFi(SystemC Fault Injection)을 사용하였다. 오류주입 실험은 프로세스 유형, 오류유형, 프로그램 종류에 따라 각각 2000회의 실험을 수행 하였다. 각각의 오류는 랜덤하게 시스템에 주입되었다. 표에서 보는바와 같은 분류는 Fault Active, System Failure, Fault recovery로 분류 하였다. 1) Fault Active는 시스템에 반영된 오류의 개수를 표현한다. 2)System

Failure는 시스템에 발생된 오류에 의해 잘못된 연산 결과를 Data Memory에 저장한 횟수이다. 3) Fault Recovery는 시스템에 발생된 오류가 고장감내형 메커니즘에 의해 복구 되거나, 또는 영향이 없는 오류(benign fault)로 인해 정상적인 동작을 한 경우이다. 본 연구에서는 시스템에서 발생된 오류 중 Fault Recovery 된 오류의 비율을 통해 각 프로세서의 오류에 대한 강건성을 평가 할 수 있다. 표 1에서 보는 바와 같이 Transient Fault의 경우, ARM에 비하여 TMR ARM과 Checkpoint ARM이 오류에 월등히 강인함을 볼 수 있다. 하지만 Permanent fault의 경우 TMR ARM 만이 오류에 강인하였다. Checkpoint ARM의 경우 같은 영역이 지속적으로 오류가 발생시, 계속적으로 Checkpoint 로의 rollback과 retry가 반복되어 정상적인 동작을 수행하지 못한다.

<표 2> FT ARM 수행시간

프로그램 목록	ARM		TMR ARM		Checkpoint ARM	
	Non fault	fault	Non fault	fault	Non fault	fault
JPG Program	794 cycle	794 cycle	794 cycle	794 cycle	794 cycle	879 cycle
Telcom Program	587cycle	587 cycle	587 cycle	587 cycle	587 cycle	673 cycle

5. 결론

고장감내형 프로세서를 설계하고, 이를 검증하기 위한 방법을 제시하였다. 본 연구에서는 ARM7 프로세서를 기본모델로 하여, 하드웨어의 의존성은 높지만 오류처리에 대한 지연시간이 없는 Module Level TMR ARM과 하드웨어는 더 적게 소모하지만, 오류를 처리하기 위한 시간지연이 되는 checkpoint ARM을 설계 하였다. 설계 언어로는 SystemC를 선택하여, 빠른 설계가 가능 하였다. 이와 같은 연구를 통해 프로세서 개발 전에는 알 수 없는, 오류처리 능력, 오류처리에 따른 지연시간 등의 성능 지표를 산출 할 수 있을 것으로 생각된다.

향후 연구 과제로 설계된 systemC모델을 RTL 레벨로 변환하여, 각 프로세스 모델별 사용 게이트수, 게이트 수준에서의 지연시간 등을 분석하고자 한다. 또한 본 실험으로 제기되는 각 FT_ARM 에 문제점(최적화된 Check point 주기)을 보완하는 연구를 진행할 것이다.

참고문헌

[1] David C. Black, Jack Donovan, Bill Bunton, Anna Keist "SystemC:Form The Ground Up" Springer, 2004.

[2] L.Anghel, M. Rebaudengo, M. Sonza, Reorda, M. Violante "Multi-level Fault Effects Evaluation" R. Velazco et.al. Radiation Effects on Embedded Systems, pp69-88, 2007

[3] Nicholas J. Wang, Sanjay J. Patel "ReStore: Symptom-Based Soft Error Detection in Microprocessors" IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 3, NO. 3, JULY-SEPTEMBER 2006

[4] T.M.Austin "DIVA : A Reliable Substrate for Deep Submicron Microarchitecture Design" in 32nd Annual International Symposium on Microarchitecture(MICRO), pp.196~207, 1999

[5] Francesco Abate, Luca Sterpone, and Massimo Violante "A New Mitigation Approach for Soft Errors in Embedded Processors" IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. 55, NO. 4, AUGUST 2008

[6] 나종화, 류대현, 갬대영 공역 "ARM System -on-Chip 구조" 홍릉과학출판사

[7] Shubu Mukherjee "Architecture Design For Soft Errors" Morgan Kaufmann Publishers

[8] http://www.freescale.com