VLIW 프로세서를 위한 소프트에러 검출 및 수정 기법

이운영, 이종원, 허인구, 권용인, 이경우, 백윤흥
서울대학교 공과대학 전기정보공학부
연세대학교 공과대학 컴퓨터과학부
e-mail : wylee, jwlee, igheo, yikwon@sor.snu.ac.kr, kyoungwoo.lee@yonsei.ac.kr, ypaek@snu.ac.kr

Soft Error Detection & Correction for VLIW Architecture

Yunrong Li, Jongwon Lee, Ingoo Heo, Yongin Kwon, Kyoungwoo Lee, Yunheung Paek
Dept. of Electrical and Computer Engineering, Seoul National University
Dept. of Computer Science, Yonsei University

요 약

임베디드 시스템에서 저전력 공급, 칩사이즈 축소, 낮은 노이즈 마진 등 설계기법이 날로 향상됨에 따라 소프트에러가 기하급수적으로 늘어나고 있다. 본 논문에서는 VLIW 아키텍처에서 치명적인 오류를 일으키는 이런 소프트에러들을 검출하고 수정하는 기법을 제안하고자 한다.

1. 서론

임베디드 시스템 설계에 있어서 신뢰성에 대한 요구가 갈수록 더 높아지고 있다. 하지만 저전압 공급, 칩사이즈 축소, 낮은 노이즈 마진 등 공정기술의 발전으로 말미암아 소프트에러는 기하급수적으로 늘어나고 이러한 잠시적인 오류들로 인하여 결국 잘못된 결과값을 얻게 되는 심각한 결과를 초래하게 된다. 소프트에러란 하드에러의 반대개념으로서 시스템에서 일시적인 데이터오류로 인하여 일으키는 에러를 말한다.

본 논문에서는 4-way VLIW 아키텍처를 Reconfigure 하여 적은 Cost Overhead 로 소프트에러를 찾아주고 수정해주는 메카니즘을 설명하고자 한다.

VLIW Instruction set 들을 보게 되면 많은 NOP 들이 있다. Compile time 에 우리는 이런 빈 NOP 에 Operation 들의 복사본을 넣어주고 Run time 에서 원본 Operation 의 결과값과 복사본 Operation 의 결과값을 비교하여 같으면 최종적으로 결과값을 레지스터에 쓰거나 메모리를 참조하며 다르면 수정하여 정확한 값을 보이도록 한다.

2 장에서 먼저 기본적인 VLIW 아키텍처와 소프트에러를 검출하기 위하여 제안한 Reconfigure VLIW 아키텍처의 구조에 대해서 소개하고, 이어서 3 장에서는 VLIW datapath 에서 소프트에러를 수정하는 솔루션에 대하여 설명하겠고 4 장에서 실험결과를 보여주고 마지막 5 장에서 전체 내용을 요약하고 결론매듭을 지으려 한다.

2. 소프트에러 검출을 위한 VLIW 아키텍처

기본적인 32bit 4-way VLIW 아키텍처의 구조는 다음과 같다. 두 개의 Integer ALU, 하나의 Integer Multiplier, 하나의 Load/Store Unit 및 하나의 Branch Unit 을 갖고 있으며 16 개 General Purpose Register, 8Mb 의 Data Memory 와 8Mb Instruction Memory 를 포함한다. 이 VLIW 아키텍처는 또한 FE, DC, EX, MEM, WB 5 Stage 의 Pipeline 을 갖는다.

위에 언급한 VLIW 아키텍처를 바탕으로 본 논문에서 제안한 새로운 VLIW 아키텍처의 구조는 Figure 1 과 같다.

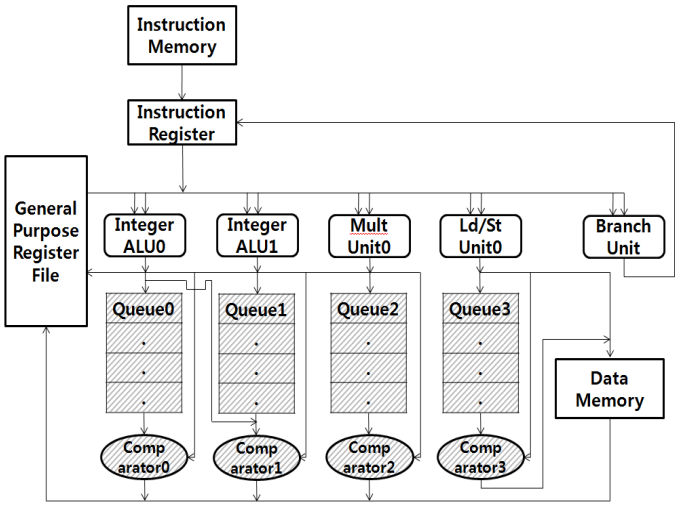

Figure 1. Proposed VLIW Architecture

Figure 1 에서 빗금 친 부분은 새로 추가 된 하드웨어 유닛들이다. Queue 는 각 슬롯마다 하나씩 갖고 있는데 Queue0~2 는 원본의 연산결과값을, Queue3 는 메모리 주소값을 임시 저장하는데 사용된다. 이렇게 저장된 임시 값들은 후에 출현되는 복사본 Operation 에서 얻은 결과값과 Comparator 에서 비교하여 같으면 최종적으로 레지스터에 쓰이거나 메모리를 참조한다. 다를 경우 복사본의 Operation 을 한번 더 돌려 수정작업을 거치는데 아래 3 장에서 설명하도록 하겠다.

3. 소프트에러 수정을 위한 Pipeline configuration

원본 operation 에서 얻은 결과값을 복사본 operation 의 결과값과 비교하여 보는데 이 단계는 Figure 2 와 같이 EX Stage 에서 진행 된다.

Cycle	0	1	2	3	4	5	6	7	8
	FE	DC	EX	MEM	WB	FE	DC	EX	MEM
		FE	DC	EX	MEM	WB	FE	DC	EX
			FE	DC	EX	MEM	WB	FE	DC
				FE	DC	EX	MEM	WB	FE
					FE	DC	EX	MEM	WB

Figure 2. Error Detected in EX stage

만약 에러가 검출 되면 Figure 3 에서처럼 다른 Stage 들은 중지시키고 EX Stage 만 다시 실행시켜, 즉 복사본 Operation 을 다시 돌려 그 결과값을 다시 원본 Operation 의 결과값과 비교한다. 결과적으로 우리는 3 개의 결과값을 얻어서 비교하게 되는데 여기에서 TMR Voting Mechanism 을 이용하여 정확한 결과값을 얻을 수 있다.

Cycle	0	1	2	3	4	5	6	7	8	9
	FE	DC	EX	MEM	WB		FE	DC	EX	MEM
		FE	DC	EX	MEM		WB	FE	DC	EX
			FE	DC	EX	EX	MEM	WB	FE	DC
				FE	DC		EX	MEM	WB	FE
					FE		DC	EX	MEM	WB

Figure 3. Redo EX stage & Stall other stages

위의 솔루션에서 알 수 있다시피 우리는 소프트에러를 찾았을 때 단지 1 cycle 의 overhead 로 소프트에러를 찾고 수정하여 시스템의 신뢰성을 높이는데 성공하였다.

4. 실험결과

제안한 새로운 32bit 4-way VLIW 아키텍처를 Synopsys Design Compiler 및 CACTI 를 이용하여 130nm 공정을 가정으로 Area 를 측정해 본 결과는 아래 Table 1 과 같다.

Memory (μm^2)	Base Archi. (μm^2)	Our Archi. (μm^2)	Overhead (%)
13606315	513616	751540	1.69%

Table 1. Area Estimation

Overhead = (Our Archi. – Base Archi.) / (Base Archi. + Memory) 를 이용하여 계산 한 결과 약 1.69%의 하드웨어 Overhead 만 생겼다.

제안한 아키텍처를 우리 SOR 연구실에서 개발한 Soargen Compiler 와 Synopsys Processor Designer 의 Simulator 를 이용하여 DSPstone 을 Benchmark 로 실행하여 아키텍처의 신뢰성을 검증하였다. Figure 4 에서는 Branch, Move Operation 을 제외한 다른 모든 Operation 들의 복사본을 만들어 적절한 VLIW Scheduling 기법을 이용하여 Schedule 한 코드를 제안한 아키텍처의 Simulator 에서 실행한 결과를 보여주고 있다.

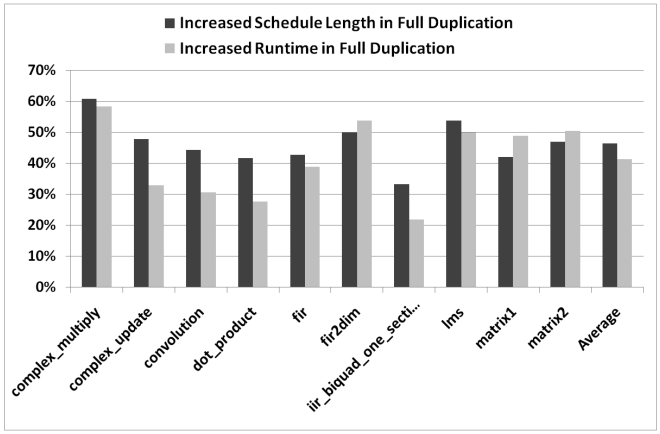

Figure 4. Increased Schedule Length & Runtime

실험결과 모든 Operation 들의 복사본을 만들어 시스템의 신뢰성을 높이는데 평균 40~50%의 Performance 손실이 있었다.

5. 결론

임베디드 시스템의 신뢰성을 향상시키기 위하여 많은 연구들이 진행되어 왔었는데 이런 소프트에러를 검출하고 수정하는 비용은 매우 비쌌다. 우리는 본 연구에서 TMR Voting Mechanism 을 이용하여 1cycle 의 overhead 로 소프트에러를 수정하였고 또한 하드웨어 cost overhead 도 1.69%로 최소화하여 시스템의 신뢰성을 높였다.

참고문헌

[1] P. Hazucha and C. Svensson. “Impact of cmos technology scaling on the atmospheric neutron soft error rate.” IEEE Trans. on Nuclear Science, 47(6):2586–2594, 2000.

[2] M. Ahn and Y. Paek. “Transactions on high-performance embedded architectures and compilers” ii. chapter Fast Code Generation for Embedded Processors with Aliased Heterogeneous Registers, pages 149–172. 2009.

[3] J. Hu, F. Li, V. Degalahal, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin. “Compiler-assisted soft error detection under performance and energy constraints in embedded systems.” ACM Transactions on Embedded Computing Systems, 8:27:1–27:30, July 2009.

[4] D. K. Pradhan. “Fault-Tolerant Computer System Design.” Prentice Hall, 1996. ISBN 0-1305-7887-8.

Acknowledgement

본 연구는 교육과학기술부/한국과학재단 우수연구센터육성사업(과제번호 2011-0000975), 2011 년도 정부(교육과학기술부)의 재원으로 한국과학재단의 국가지정연구실사업(No.2011-0018609) 및 IDEC 의 지원을 받아 수행되었습니다.