

SE3208 내장형 프로세서를 위한 시뮬레이터

¹송헌철^o, ²오형철

¹고려대학교 대학원 전자정보공학과

²고려대학교(서창) 공학부

ohyeong@korea.ac.kr

A Simulator for SE3208 Embedded Processor

¹Hyunchul Song^o, ²Hyeongcheol Oh

¹Dept. of Elec. & Info. Engineering, Graduate School, Korea University

²School of Engineering, Korea University at Seo-Chang

요 약

내장형 프로세서인 SE3208의 소프트웨어 모델을 개발하고, SimpleScalar 도구 모음에 추가하여 SE3208 프로세서를 위한 시뮬레이터를 개발하였다. 소프트웨어 모델은 SimpleScalar 도구의 환경에 맞춰 함수를 재사용하는 방법으로 작성되었으며, 3단 단일 파이프라인을 갖는 SE3208을 cycle단위로 시뮬레이션 할 수 있도록 시뮬레이터를 변경하였다. 동일한 프로그램을 SE3208 소프트웨어 모델과 PC 상에서 각각 실행시켜 그 결과를 비교함으로써 개발된 시뮬레이터의 정확성을 검증하였으며, 다양한 방식과 모드의 시뮬레이션 기능을 검증하고 성능을 비교하였다.

1. 서 론

오늘날 프로세서들은 매우 복잡해지고, 평가하고 검증하기가 더욱 어려워지고 있다. 흔히 프로세서 하드웨어 개발자들은 HDL (hardware description languages)을 사용하여 하드웨어를 설계하므로, HDL수준에서의 시뮬레이션으로도 설계를 평가하고 검증할 수도 있겠으나, 이 방법은 지나치게 긴 시뮬레이션 시간으로 인하여 프로세서의 설계 과정 전반에 걸쳐서 사용하기에는 부적절하다. 이로 인하여, 실제 프로세서의 설계에는 흔히 소프트웨어 모델을 개발하여 사용한다. 개발자들은 소프트웨어 모델을, 프로세서를 설계하는 과정에서 여러 가지 설계 선택들을 빠르게 평가해보고 동작을 검증하며 설계된 프로세서 동작의 정확성에 관한 기준을 제공하는 도구로서 사용한다. 또한, 개발된 소프트웨어 모델은 후에 프로세서의 응용 단계에서 응용 프로그램의 개발을 위한 대상 모델로서 사용될 수 있다.

본 논문에서는 컴퓨터 설계자들에게 널리 사용되는 SimpleScalar 도구 모음(Tool Set)[1,2]과 함께 사용할 것을 목표로, Adchips사의 저전력 응용용 내장형 프로세서인 SE3208[3,4]의 소프트웨어 모델을 개발하였다. SimpleScalar 시뮬레이션 툴에 포팅된 SE3208의 소프트웨어 모델은 SimpleScalar 시뮬레이션 환경에 맞춰 개발되어, SimpleScalar 도구 모음의 거의 모든 시뮬레이터들과 함께 사용될 수 있다. 또한, 도구 모음에서 제공하는 옵션의 일부를 제외한 나머지는 그대로 사용할 수 있으며, 포팅 과정에서 기존 프로그램을 크게 변경하지 않고 SimpleScalar 도구 모음에서 제공하는 함수를 재사용하여 시뮬레이터의 코드 크기를 줄였다.

본 논문의 구성은 다음과 같다. 제2절에서 SimpleScalar 도구 모음과 SE3208 프로세서의 구조를 소개하고, 제3절에서 SE3208 모델의 개발과 포팅 방법에 대하여 설명한다. 제4절에서는 SE3208을 cycle단위로 시뮬

레이션 할 수 있는 sim-order 시뮬레이터의 동작에 대하여 설명하고, 제5절에서는 sim-order 시뮬레이터를 검증하고 sim-order 시뮬레이터의 2가지 모드를 비교하고 평가한다. 마지막으로 제6절에서 결론을 맺는다.

2. 배경

2.1 SimpleScalar 도구 모음

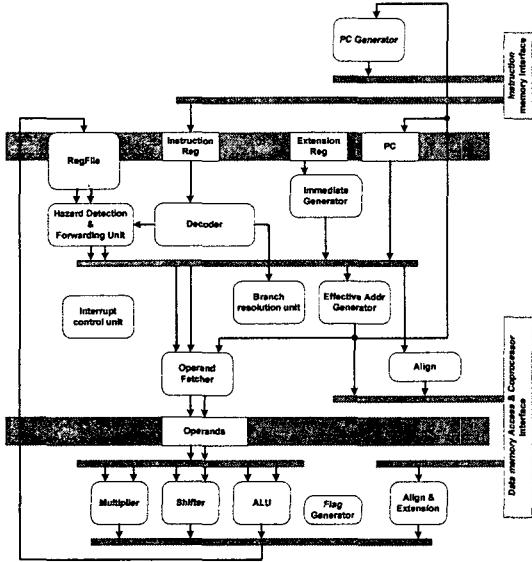
SimpleScalar 도구 모음[1,2]은 컴파일러, 어셈블러, 링커와 SimpleScalar 구조를 위한 시뮬레이터들로 구성되어 있다. 본 논문에서는 LLC[5]에서 제공하는 비영리 배포판인 SimpleScalar 도구 모음에서 SimpleScalar 시뮬레이터(simplesim-3v0d)를 사용하였으며, 3단 파이프라인 구조의 SE3208 모델을 시뮬레이션할 수 있도록, 5단 파이프라인 구조만을 지원하는 sim-outorder.c 파일을 수정한 sim-order.c 파일을 작성하였다.

2.2 SE3208 프로세서

SE3208 프로세서[3,4]는 30MIPS이하의 저비용 내장형 코어로 개발된 EISC 기반의 프로세서이다. 그림 1은 본 논문에서 개발의 대상으로 사용한 SE3208 프로세서의 마이크로 구조이다.

그림 1에 도시한 바와 같이, SE3208은 레지스터 파일을 중심으로 3개의 데이터 버스(A버스, B버스, C버스)를 지니고 있다. A버스, B버스는 연산블록으로 입력되는 버스이며, C버스는 WB(Write-Back)에 사용되는 버스이다. SE3208의 파이프라인은 명령어 패치를 하는 IF단(Instruction Fetch Stage), 명령어 해석 및 주소 생성을 수행하는 ID단(Instruction Decode Stage), 산술논리 연산 및 곱셈 연산을 수행하거나 또는 메모리 접근을 수행하는 MEM/EX 단 (MEMory access/EXecution Stage)의

3단으로 구성되어 있으며 외부 메모리 접근에는 명령어와 데이터 버스를 분리한 하바드 구조(Havard architecture)를 취하고 있다.

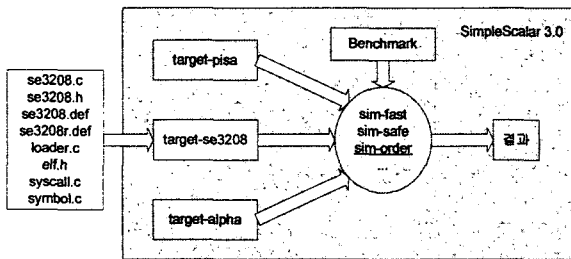


[그림 1] SE3208의 마이크로 구조[3]

SE3208은 32비트 연산을 지원하는 ALU와 배럴 쉬프트러(barrel shifter)를 지니고 있으며, 32x32=32비트 계산을 한 사이클에 수행하는 곱셈기를 가지고 있다. 또한 32비트의 범용 레지스터(General Purpose Register; GPR) 8개와 특수 목적 레지스터(Special Purpose Register; SPR) 5개를 가지고 있다.

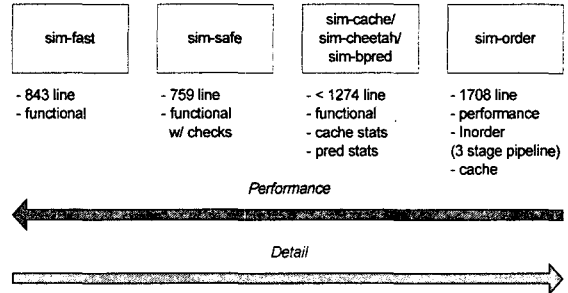
3. SE3208 poring 방법 및 구현

그림 2는 SimpleScalar 시뮬레이션 툴에 SE3208 포팅한 방법을 도시한 것으로서, 대상 프로세서로서 기존의 PISA, ALPHA 외에 SE3208을 추가하는 방식으로 구현하였다. 그림 2에서 보인 바와 같이 target-se3208은 se3208.c, se3208.h, se3208.def, se3208r.def, loader.c, elf.h, syscall.c, symbol.c의 8개 파일로 구성되었다.



[그림 2] SE3208의 porting

symbol.c 파일은 SimpleScalar 시뮬레이션 툴에 있는 파일을 그대로 사용하였고 나머지 7개의 파일은 SimpleScalar 시뮬레이션 툴 환경에 맞추어 작성되었다. 그림 2의 원안에 보인, SE3208을 시뮬레이션 할 수 있는 시뮬레이터 종류는 그림 3과 같은데, SE3208의 3단 파이프라인을 처리할 수 있도록 작성된 sim-order.c 파일이 추가 되었다.



[그림 3] 시뮬레이터의 종류

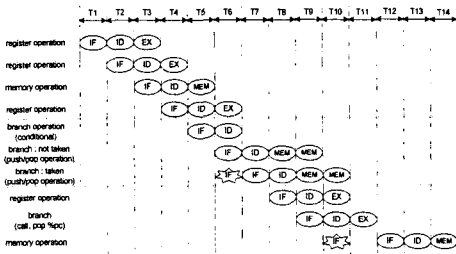
4. sim-order 동작

sim-order 시뮬레이터는 빠른 시뮬레이션 모드와 정밀 시뮬레이션 모드의 2가지 형태로 수행할 수 있다. 빠른 시뮬레이션은 기능 시뮬레이션과 같으며, 이는 -fastfwd 옵션을 사용하여 일부 명령어를 기능 시뮬레이션으로 수행시킬 수 있다.

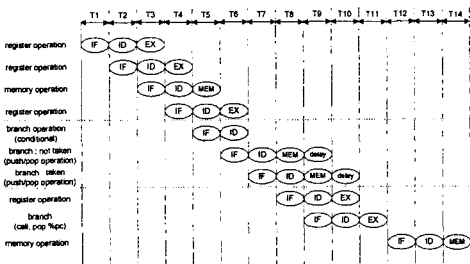
sim-order 시뮬레이터의 정밀 시뮬레이션에서는 cycle timer 모드와 fast timer모드가 있다. cycle timer 모드에서는 그림 4와 같이, push와 pop 명령의 각 마이크로 동작을 각 cycle 마다 수행 한다. 즉, 두개의 register 내용을 스택에 넣는 명령어인 "push %r0, %r1"의 경우, 첫 cycle에서는 push %r0를 수행하고 두 번째 cycle에서는 push %r1을 수행한다. 또한 분기 명령어에서 분기 되면 다음 명령어 값을 삭제(flush)하도록 하였다.

fast timer 모드에서는, 그림 5에서와 같이, 여러개의 register의 내용을 처리하는 1개의 push와 pop 명령어가 1 cycle에 모든 동작을 수행하고 나머지 cycle동안은 동작을 하지 않고 기다리는 기능이다. 이는 큰 프로그램을 수행할 때 많은 레지스터 값을 push, pop하게 되면 시간을 많이 소모하게 되므로, 모든 동작을 1 cycle에 수행함으로써 시뮬레이션 속도를 높이기 위함이다. 또한 분기명령어에서 분기된 경우에 다음 명령어를 fetch 하지 않도록 구현 하였다. 이렇게 한 이유도 fetch를 수행함으로써 생기는 시간 지연을 줄이기 위함이다.

위의 2가지 모드는 sim-order.c 파일에 #define CYCLE_TIMER로 선언되어 있어서, 정상 상태에서는 cycle timer 모드로 동작 하지만 이 부분을 주석 처리하면 fast timer 모드로 동작시킬 수 있다.



[그림 4] cycle timer 모드

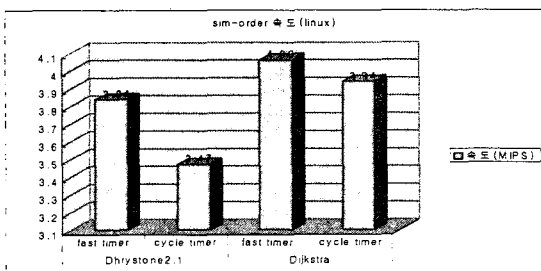


[그림 5] fast timer 모드

5. 검증 및 성능 비교

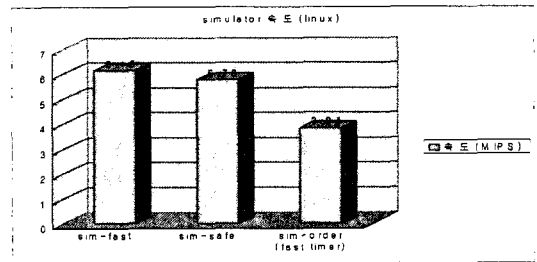
sim-order 시뮬레이터는 Dhrystone2.1과 MiBench [6]의 네트워크용 응용프로그램인 Dijkstra 프로그램을 사용하여 검증하였다. Dhrystone2.1과 Dijkstra를 시뮬레이션한 결과는, 동일 프로그램을 PC상에서 실행한 결과와 동일함을 확인하였으며, 이를 통하여 작성된 시뮬레이터의 정확성을 검증하였다. 이 과정에서 각 cycle당 레지스터의 값을 RTL 시뮬레이션에서 얻어진 레지스터 값과 비교하였다.

한편, fast timer 모드와 cycle timer 모드의 속도를 비교하고, 그 결과를 그림 6에 보였다. 결과 값은 Dhrystone2.1 벤치마크를 10000000번을 수행하여 얻은 것과 20번의 반복 주기로 최단 경로를 구하는 Dijkstra를 수행하여 얻은 것이다. 두 결과를 토대로 fast timer 모드에서 약 3%~10% 더 빠르게 시뮬레이션이 수행됨을 관찰하였다.



[그림 6] 모드에 따른 속도 비교 (단위: MIPS)

그림 7은 그림 6과 동일한 조건에서, 세 가지 시뮬레이터인 sim-fast와 sim-safe, sim-order를 이용하여 SE3208을 시뮬레이션한 속도를 비교한 것이다. 그림 7의 결과에서 기능 시뮬레이션을 수행하는 sim-fast의 수행속도가 6.15MIPS로 가장 빠르며, cycle 단위로 처리하는 sim-order가 3.84MIPS로 가장 느린 것을 알 수 있다.



[그림 7] 방식에 따른 속도 비교 (단위:MIPS)

6. 결론

본 논문에서는 3단 단일 파이프라인을 갖는, 32비트 내장형 프로세서인 SE3208 프로세서의 소프트웨어 모델을 개발하고, 이를 널리 사용되는 프로세서 시뮬레이터인 SimpleScalar 시뮬레이션 툴에 포팅하였다. 얻어진 시뮬레이터의 정확성은, 동일한 프로그램을 SE3208 소프트웨어 모델과 PC 상에서 각각 실행시켜, 그 결과를 비교함으로써 검증하였으며, 다양한 방식과 모드의 시뮬레이션 기능을 검증하고 성능을 비교하였다.

본 논문에서 개발된 SE3208 소프트웨어 모델은 EISC 프로세서를 내장하는 내장형 시스템의 개발에 사용될 수 있다.

감사의 글

본 연구는 (주)에이디칩스 및 반도체설계교육센터(IDEC)의 지원을 받아 수행되었습니다.

7. 참고 문헌

- [1] T. Austin, E. Larson, and D. Ernst. "SimpleScalar: An Infrastructure for Computer System Modeling." IEEE Computer, 35(2):59-67, Feb. 2002.
- [2] D. Burger and T. M. Austin, "The SimpleScalar Tool Set, Version 2.0, Computer Architecture News, pages 13-25, June 1997
- [3] 박정규, 이성재, 오형철. "32비트 저에너지 내장형 프로세서의 설계," 제16회 통신정보 합동학술대회 2006.
- [4] ADChips, <http://www.adc.co.kr/english/Products/IPCore/se3208.htm>
- [5] SimpleScalar LLC, <http://www.simplescalar.com>
- [6] M.R. Guthaus, J.S. Ringenberg, D. Ernst, T.M. Austin, T. Mudge, R.B. Brown. "MiBench: A free, commercially representative embedded benchmark suite," Proc. IEEE 4th Ann. Workshop on Workload Characterization, pp. 3-14, Dec. 2001.