1999年度 大韓電子工學會 秋季綜合學術大會 論文集 第 22 卷 第2號 99/11

ARM 프로세서용 부동 소수점 보조 프로세서 개발

김 태 민(金 泰 珉),신 명 철(申 明 澈), 박 인 철(朴 仁 哲)
한국과학기술원 전기 및 전자공학과
전화 : (042)869-4031 / 팩스 : (042) 869-4040

Development of a Floating Point Co-Processor for ARM Processor

Taemin Kim, Myung-cheol Shin and In-cheol Park
Department of Electrical Engineering
Korea Advanced Institute of Science and Technology
E-mail : (tmkim,mcshin,icpark)@vslab.kaist.ac.kr

Abstract

In this paper, we present a coprocessor that can operate with ARM microprocessors.

The coprocessor supports IEEE 754 standard single- and double-precision binary floating point arithmetic operations. The design objective is to achieve minimum-area, low-power and acceleration of processing power of ARM microprocessors. The instruction set is compatible with ARM7500FE. The coprocessor is written in verilog HDL and synthesized by the SYNOPSYS Design Compiler. The gate count is 38,115 and critical path delay is 9.52ns.

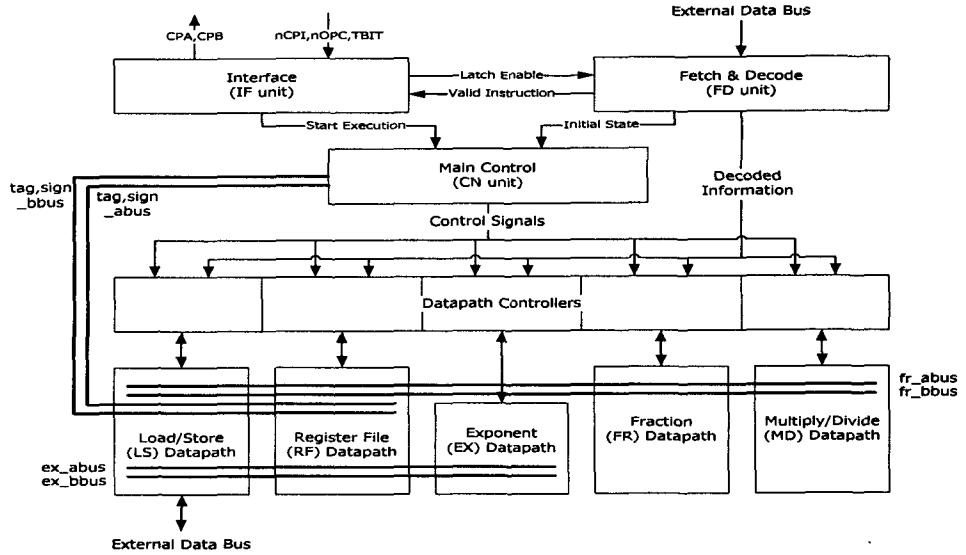
I. 서론

최근의 자바 머신, 네트워크 컴퓨터, 셋탑 박스 등의 멀티미디어 시스템, 무선 통신과 같은 휴대용 시스템, 그리고 여러 내장 콘트롤러 등에 대한 요구가 급증하면서 파워, 속도, 가격의 모든 관점에서 우수한 특성을 가지는 콘트롤러의 필요성이 증대되었다. 이러한 제반 조건을 충분히 만족시키는 콘트롤러로서 ARM프로세서를 들 수 있다. 그러나 정수 연산 능력으로 한정된 ARM프로세서 자체만으로는 고도의 정확도를 요구하는 콘트롤러나 신호 처리 분야, 3D-Graphics 그리고 화상처리와 같은 최근에 들어 관심의 초점이 되고 있는 멀티미디어 분야에 대한 사용자들의 요구를 처리 속도의 측면에서는 만족시켜 줄 수 없다.

정수 처리 능력의 제한은 일반적으로 빈번한 오버플로우의 조사, 데이터의 사전, 사후 지수 조정, 상대적으로 약한 신호 대 잡음비 등에 기인한다고 볼 수 있다. 적용 대상의 측면에서 제어용 콘트롤러나 가상현실 시스템의 좌표 변환 등 현재는 물론 앞으로도 그 요구가 계속적으로 증가할 것으로 예상되는 분야는 대부분 이러한 제약점 등에 시달리고 있으며, 이를 해결하기 위해서는 ARM과 같은 콘트롤러에 대한 전용 부동 소수점 연산 프로세서의 등장이 요구된다.

이러한 목적의 보조 프로세서의 개발에 있어, 기존의 수치 보조프로세서등이 대부분 범용 마이크로프로세서와 함께 독립적인 컴퓨터 시스템에 응용될 수 있도록 개발된 데 반해, ARM 프로세서의 개발 철학을 유지하면서 그 부동 소수점 처리 능력을 효과적으로 부가하기 위해서는 전용 부동 소수점 보조 프로세서의 설계에 있어 연산 속도는 물론 전력 소비와 칩 면적에서 충분히 고려된 설계가 요구된다.

II장에서는 개발하고 있는 부동 소수점 보조 프로세서의 구조에 대해서 설명하고, III장에서는 설계된 프로세서의 합성결과를 보이겠다. 그리고 V장에서는 결론을 맺는다.



<그림-1>부동 소수점 보조 프로세서의 전체 구조

II. 프로세서 구조

<그림-1>은 설계된 프로세서의 전체 구조를 나타낸 그림이다. 그림의 상단 오른쪽의 FD Unit은 명령어의 펫치와 디코드를 하고 왼쪽의 IF Unit은 ARM 프로세서와의 Interface를 담당한다. 그림 중앙의 CN Unit은 전체적인 제어를 하고 각 기능 블록의 세부 제어는 CN Unit 아래 부분의 Datapath Controller가 담당하게 된다. 아래에 보조프로세서의 구조적인 특징들에 대해서 설명한다.

A. Data Format

설계된 프로세서는 Data Format 은 IEEE754-1985 Standard를 따르고 있다. 그리고, Single Precision과 Double Precision을 모두 지원한다.

프로세서 외부에서의 Data Format은 IEEE754-Standard를 따르고 프로세서 내부에서는 예외경우들을 다루기 위하여 표준과는 약간 다른 형태의 자료 구조를 택하고 있다. 물론 부동 소수점 연산을 수행할 때의 자료 구조는 표준과 같고, Register File에 저장될 때의 자료 구조에 4-bit의 Tag가 붙게 된다. 내부에 저장되는 자료구조를 <그림-2>에 나타냈다.

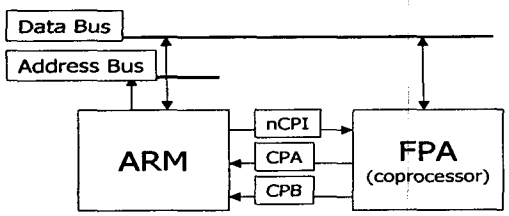
<그림-2>Register File에 저장되는 Data Format

한편, Single Precision과 Double Precision 모두를 지원하기 위하여 내부 자료 처리 과정은 모두 Double Precision으로 수행한다. 그리고 Single Precision의 경우는 처리가 끝난 후에 내부 저장은 Double Precision으로 하고 외부로 전송 될 때는 Single로 변형하게 된다.

B. ARM 프로세서와의 Interface

ARM 프로세서와의 통신은 ARM 프로세서에 규정된 내용을 따르게 된다. 설계된 부동 소수점 프로세서는 스스로 메모리 주소를 생성하지 않는다. 명령어를 메모리로부터 가져 올 때 메모리의 주소는 ARM 프로세서가 생성하고 보조 프로세서는 버스에 실려 있는 명령어를 가져오기만 한다. 물론 그 명령어를 ARM 프로세서도 가져가게 되고 명령어 디코딩을 수행한다. 만약에 가져온 명령어가 ARM 프로세서의 명령어가 아니라면 ARM 프로세서는 주위의 보조 프로세서들에게 ARM 프로세서의 명령이 아니라는 신호를 보내게 된다. 그 신호가 <그림-3>에 나타나 있는 nCPI이다. 이것이 Low Level로 떨어지게 되면 보조 프로세서 명령이란 뜻이 된다. 보조 프로세서는 nCPI가 떨어진 것을 확인하고 자신의 상태를 ARM 프로세서에게 알려주게 된다. 보조 프로세서가 자신의 상태를 알려주는 신호는 2개가 있다. 하나는 CPA이고 나머지는 CPB신호이다. CPA는 보조 프로세서가 명령어를 디코딩해서 그것이 자신의 명령어이면 CPA를 Low로 떨어뜨리게 된다. 자신의 명령어가 아니면 CPA는 High를 유지하

게 된다. CPB는 보조 프로세서가 명령어를 수행중인지 아닌지를 나타낸다. 만약에 보조 프로세서가 어떤 명령을 수행하고 있으면 CPB는 그 명령어 수행이 끝나는 Cycle까지 High를 유지하게 된다. 그 명령어 수행이 끝나면 명령어 수행의 마지막 Cycle에서 CPB가 Low로 떨어지게 된다. CPA가 Low이고 CPB가 High일 때는 ARM 프로세서는 어떠한 일도 수행하지 않고 기다리게 된다. 그러나 ARM 프로세서에 인터럽트가 발생하면 보조프로세서와의 Interface과정을 벗어나서 인터럽트를 처리하게 된다.



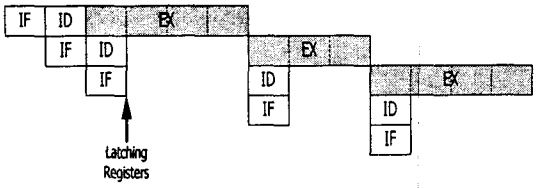
<그림-3>ARM과 보조프로세서의 Interface

C. Pipeline

보조 프로세서 명령어가 나타났을 경우 바르게 응답하기 위해서 보조 프로세서는 명령어의 복사 본을 가지고 있어야 한다. 즉, ARM 프로세서의 Pipeline에서 IF(Instruction Fetch)와 ID(Instruction Decode)단까지는 ARM 프로세서를 그대로 따라가도록 한다.

부동 소수점 연산 보조 프로세서의 Pipeline은 ARM 프로세서의 Pipeline과 같은 3단계 Pipeline을 가진다. 즉, IF(Instruction Fetch), ID(Instruction Decoding), EX(Execution)단이 Pipeline되어 있다.

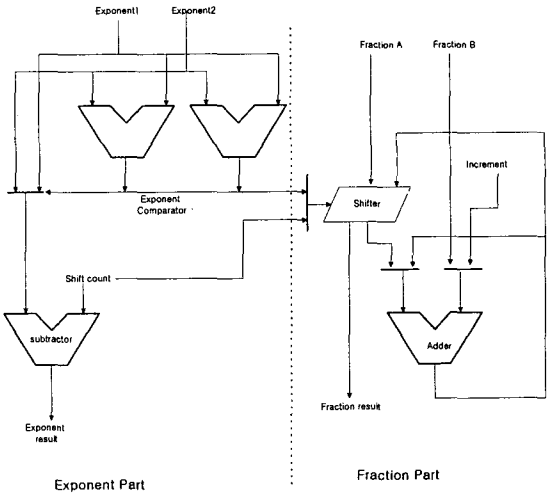
한편, 부동 소수점 연산의 경우는 1-Cycle에 명령어의 수행이 끝나는 경우가 매우 드물다. 즉, 대부분의 명령어는 수행하는데 여러 Cycle이 걸리게 된다. 그런 경우에 Pipeline에서 명령어의 디코딩과 펫치를 더 이상하지 않고 이전에 디코딩하고 펫치한 정보들을 유지하게 된다. 그리고 현재 EX단에 있는 명령어의 수행이 끝나면 Pipeline이 다시 진행되게 된다. 그 과정을 나타낸 그림이 <그림-4>이다.



<그림-4>보조 프로세서의 Pipeline

D. ADD/SUB 기능 블록의 구조

ADD/SUB 연산의 경우 Pre-Alignment->ADD/SUB->ROUND->Post-Alignment의 순서로 연산을 수행한다. 만약에 ADD/SUB가 Pipeline 구조로 되어 있으면 Pre-Alignment와 Post-Alignment에 필요한 시프터가 2개가 있어야 한다. 또한 ADD/SUB와 ROUND에 가산기가 필요한데 파이프라인 구조이면 가산기 2개가 필요하다. 그러나 이 설계에서는 EX단이 Pipeline되어 있지 않으므로 시프터와 가산기를 공유해서 쓸 수가 있다. 즉, 1개씩의 시프터와 가산기를 줄일 수 있다. <그림-5>에 설계된 ADD/SUB 블록의 구조를 나타냈다.



<그림-5>설계된 ADD/SUB 기능 블록의 구조

E. MUL/DIV 기능 블록의 구조

Mul/Div 데이터패스는 부동 소수점 곱셈과 나눗셈의 소수 부분의 연산을 수행한다. 곱셈과 나눗셈은 각각 Modified-Booth 알고리즘과 Radix-4 SRT 알고리즘을 사용한다. Booth 알고리즘을 이용하여 부분 합을 더하는 CSA(Carry Save Adder) 4단이 연결되어 있어서 7사이클만에 소수의 곱셈을 마칠 수 있다.

면적을 줄이기 위해서 곱셈기의 CSA를 공유하여 나눗셈의 중간 합을 구하고, 곱셈기의 가드 자리들을 구하는 데 쓰이는 8-bit 가산기를 공유하여 나눗셈의 중간 합의 상위 7 비트를 구한다.

F. 컨트롤 블록의 설계

부동소수점 연산 보조 프로세서에서는 일반적으로 데이터 프로세싱의 흐름이 일정하기 때문에 데이터패스 컨트롤 신호들이 명령어에 따라 사이클 별로 규칙적으로 발생된다. 따라서, 로컬 컨트롤을 구현할 때 가장 쉽고 효율적인 방법이 명령어 별로 FSM(Finite State Machine)을 만들어 그 상태에 따라서 컨트롤 신호들을 내 주는 것이다. 그런데 HDL로 컨트롤러를 설

계할 때 각 상태와 컨트롤 신호와의 관계를 직관적으로 기술하게 되면, 오류가 발생했을 때 오류원인을 발견하기도 어려울 뿐만 아니라, FSM을 최적화 하기도 어렵다. 따라서 본 설계에서는 컨트롤러들을 기술할 때 모든 컨트롤 신호들의 값을 FSM의 상태와 입력신호들의 모든 경우에 대해 표를 만드는 방법을 채택하였다. 이렇게 하면 모든 신호의 값을 모든 순간에 대해 바로 알아볼 수 있기 때문에 오류원인을 찾기가 용이하다. 그리고 이 표를 PLA 형식으로 기술하면 Espresso를 이용하여 FSM과 함께 최적화하기도 쉽다. 또한 합성된 결과를 예측하기도 용이하다.

III. 합성 결과

보조 프로세서의 설계는 Verilog로 RTL 기술을 하고 Synopsys의 Design Compiler를 이용하여 합성하였다. Design Library는 0.35um 3.3V공정을 이용했다.

각 블록들의 합성결과를 <표-1>에 나타냈다. Gate Count의 경우 2-input NAND를 Gate 1개로 환산해서 나타낸 값이다.

Unit Name	Count	Unit Name	Count
CNunit (main control)	654	IFunit (interface)	45
EXunit (exponent)	1,387	LSunit (load/store)	1,505
FDunit (fetch/decode)	1,266	MDunit (mul/div)	10,616
FRunit (fraction)	15,298	RFunit (register file)	7,344

<표-1>합성 후의 Gate Count

총 Gate Count는 38,115이다. 그리고, Critical Path의 Delay는 9.52ns로서 목표치인 80MHz로 동작하는 데에는 무리가 없을 것이다.

IV. 결론 및 추후 연구

멀티미디어 응용 분야에 많이 쓰일 것으로 예상되는 부동 소수점 연산 보조 프로세서를 설계하였다.

칩의 면적을 줄이기 위하여 곱셈과 나눗셈의 기능 블록에서는 CSA(Carry Save Adder), Mux, 8-bit 가산기를 공유하도록 설계하였다. 그리고 덧셈과 뺄셈 기능 블록에서는 시프터와 가산기를 각각 1개씩 줄일 수 있는 구조로 설계하였다.

Verilog로 RTL 기술을 하였고 0.35um, 3.3V공정 Library로 합성하였다.

추후에 더 연구를 할 것은 저전력을 위한 구조를 제안하고 효율적인 검증 방법을 제안하도록 한다.

References

[1] I. Koren, "COMPUTER ARITHMETIC ALGORITHMS", Prentice Hall International, 1993

[2] R.V.K. Pillai, et al.,"A Low Power Approach to Floating Point Adder Design", ICCD, pp178-185, 1997

[3] Stuart F. Oberman and Michael J. Flynn, "A VARIABLE LATENCY PIPELINED FLOATING-POINT ADDER", Stanford University Technical Report, CSL-TR-96-689

[4] Hiroki Suzuki, et al., "Leading-Zero Anticipatory Logic for High-Speed Floating Point Addition", IEEE JSSC, pp1157-1164, Aug. 1996.

[5] Jerome T. Coonen, "An Implementation Guide to a Proposed Standard for Floating-Point Arithmetic", IEEE Computer, Jan. 1980

[6] J. Hennessy and D. A. Patterson, "Computer architecture a quantitative approach", Morgan Kaufmann Publishers, inc.,A.1-A.53,1990

[7] T.Han, "On Chip Floating Point Unit Design for K486 Microprocessor", Master's Thesis, Dept. of EE, KAIST, Korea, 1994

[8] S.Chun, "A Design of Floating-Point Arithmetic Unit Satisfying IEEE Standard 754", Master's Thesis, Dept. of EE, KAIST, Korea, 1996

[9] C.H.Ryu, "A Design of Floating-Point Unit For VLIW Processor", Master's Thesis, Dept. of EE, KAIST, Korea, 1992

[10] J.Kim,"A Design of Floating-Point Unit for Low-power Processor", Master's Thesis, Dept. of EE, KAIST, Korea, 1998

[11] Y.Kim,"A Design fo Low Power Floating-Point Arithmetic Units", Master's Thesis, Dept. of EE, KAIST, Korea, 1999

[12] S.Tak,"A Design of Pipelined Floating-Point Arithmetic Unit", Master's Thesis, Dept. of EE, KAIST, Korea, 1994