Block level parallelism 을 위한 수정된 SIMD Architecture 설계

윤종희, 김대호, 안민욱, 최영규, 김호균, 양승준, 백윤흥
서울대학교 공과대학 전기컴퓨터 공학부
e-mail : jhyoun, dhkim, mwahn, ykchoi, hkkim, sjyang, ypaek@compiler.snu.ac.kr

A Design of Modified SIMD Architecture for block level parallelism

Jonghee Youn, Daeho Kim, Minwook Ahn, Youngkyu Choi, Hokyun Kim, Seungjun Yang, Yunheung Paek
School of Electrical Engineering, Seoul National University

요 약

미디어 어플리케이션, 특히 비디오 어플리케이션의 경우 커널 코드를 얼마나 효과적으로 처리하느냐에 따라 전체적인 성능에 큰 차이가 생긴다. 이러한 커널 코드를 효과적으로 처리하기 위해, 일반적인 DSP co-processor 에 SIMD 구조를 추가한 아키텍처를 설계하여 비디오 어플리케이션의 전체적인 성능을 향상할 수 있도록 하였다.

1. 서론

최근 들어, 임베디드 시스템에서 미디어 어플리케이션을 구동하는 경우가 점점 빈번해지고 있다. 이러한 시스템은 하나 이상의 일반적인 프로세서(General Purpose Processors, GPPs)와 특정 어플리케이션에 특화된 디지털 신호처리 프로세서(Digital Signal Processors), 내부/외부 메모리 및 시스템 버스에 연결되는 주변 장치들을 포함한다.

본 논문에서는 먼저 미디어 어플리케이션, 특히 H.264 코덱과 같은 비디오 어플리케이션을 효과적으로 처리하기 위한 이종 멀티코어 시스템 온 칩 아키텍처를 소개한다. 또한 비디오 어플리케이션의 커널 코드를 보다 더 효과적으로 처리하기 위해, 설계한 아키텍처에 SIMD 구조를 추가하였음을 설명한다.

2. 기본 아키텍처 모델

비디오 어플리케이션 처리를 위해, 먼저 기본적인 아키텍처 모델을 설계하였다. 설계된 아키텍처는 DSP co-프로세서로써, 16 bit 의 데이터 및 명령어 구조를 통해 바이너리 코드의 크기 및 전력 소모를 줄일 수 있으며 비디오 어플리케이션의 커널 코드를 처리하기에 충분한 성능을 가지고 있다.

이 아키텍처의 파이프라인 구조 및 레지스터 파일, 메모리/버스 구조와 ISA(Instruction Set Architecture)는 LISA(Language for Instruction Set Architecture)[1]로 기술되었다. 이렇게 LISA 로 기술된 아키텍처의 명세를 LISA 프로세서 설계 플랫폼(LPDP)에 제공하면, 아키텍처의 검증 및 어플리케이션 개발을 위해 필요한 소프트웨어 도구가 생성된다[2]. 여기에 좀 더 자세한 명세를 덧붙일 경우, LDPD 는 게이트 수준의 합성 가능한 HDL 코드를 생성한다. LPDP 가 제공하는 도구들은 다음과 같다[3].

- Language debugger
- Assembler & Disassembler, Linker
- ISA simulator & debugger

위와 같은 도구를 이용하여 설계한 기본 아키텍처의 특성은 다음과 같다.

- 디지털 신호 처리를 위한 DSP co-프로세서
- 16bit 데이터 및 명령어 path
- 4 단계 파이프라인 (FE, DC, EX, WB)
- 1 개의 레지스터 파일 및 데이터 메모리
- BUS 가 없는 내부 메모리
- DMA 및 OS 지원 제외

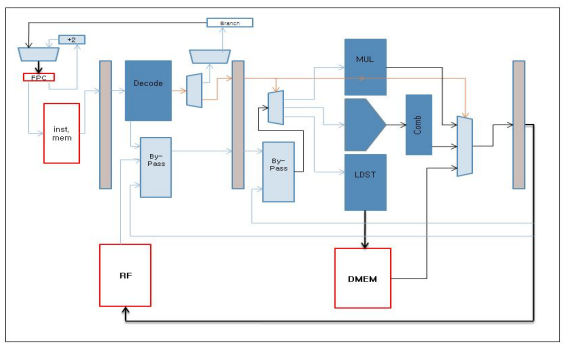

그림 1 파이프라인 단계 및 데이터 패스

3. Simple SIMD 아키텍처 모델

성능 향상을 위해 기본 아키텍처에 수정된 SIMD 구조를 추가하였다.

SIMD(Single Instruction, Multiple Data) 구조는 1960 년대 초에 제안되었다. 그 당시의 SIMD 는 여러 개의 프로세서를 이용하여 성능 및 유용성을 동시에 향상시키고자 한 아이디어 중 하나였다[4]. 오늘날, 각각의 functional unit 을 위한 작은 규모의 SIMD 명령어는 개인용 컴퓨터 하드웨어에 널리 쓰이고 있다.

SIMD 는 다중 프로세서에서 실행되는 동일한 명령어가 다른 데이터 스트림을 가지도록 한다. 이는 SoC 시스템이 하나 이상의 프로세서를 포함하든 그렇지 않든 간에 각각의 functional unit 이 고유한 메모리 공간을 가지며, 분리된 레지스터 파일 영역을 가질 수도 있다는 것을 뜻한다.

SIMD 구조는 미디어 어플리케이션을 다루는 데 있어서 두 가지 장점을 가지고 있다[5]. 첫째로, 데이터를 하나의 블락 안에 있는 것으로 볼 수 있고, 동시에 여러 개의 데이터를 읽어 들이거나 계산해 낼 수 있다. 이를 통해 프로세서가 functional unit 에 데이터를 전달하는 시간을 줄일 수 있다. 둘째로, SIMD 는 보통 하나의 명령어 내에 포함된 여러 데이터에 동시에 적용될 수 있는 명령어들을 포함한다. 만약 SIMD 시스템이 네 개의 데이터 짝을 동시에 읽어 들였을 경우, 논리적/산술적 명령어들은 각각의 짝에 독립적으로 적용될 수 있다. 이 네 개의 명령어 결과는 한 번에 계산되며, 결과적으로 블록 수준의 성능을 향상시킨다.

1980 년대에 들어서면서 SIMD 구조는 단일 코어 프로세서에서의 명령어 또는 블록 수준 병렬 처리(ILP/BLP)를 위해 수정되었다. SIMD 시스템은 여러 개의 데이터를 한 번의 연산으로 처리하는 특수한 명령어를 정의함으로써 구현된다. 이러한 방법은 서로 다른 데이터 집합들과 레지스터 배정, 스케쥴링과 코드 생성과 같은 컴파일러 작업들과 병렬 연산 간의 이해득실 측정을 필요로 한다. 대부분의 경우, SIMD 명령어는 보이지 않는 규약이나 데이터 접근에 대한 제한을 필요로 하며, 이는 소프트웨어 엔지니어로 하여금 소프트웨어 개발 도구(Software Development Toolkit, SDK)를 개발하는 것을 어렵게 한다.

우리의 SIMD 구조는 컴파일러에서의 레지스터 배정 및 코드 생성에서의 어려움을 줄일 수 있도록 수정되었다. 또한 어떠한 보이지 않는 규약이나 제한이 존재하지 않는다. 수정된 SIMD 구조에서, SIMD 명령어는 특정 상태 레지스터가 설정되었을 경우에만 활성화된다. 명령어가 활성화된 후 각각의 명령어 슬롯은 각자의 레지스터 파일과 데이터 메모리에 묶이게 되고, 이로 인해 보통의 명령어를 실행시킬 때처럼 저장 공간 및 functional unit 을 사용하게 된다. 소프트웨어 엔지니어는 컴파일러의 레지스터 배정과 코드 생성 부분을 SIMD 구조로 바꿀 필요가 없지만, SIMD 연산을 이용하여 병렬 처리할 수 있는 블록을 어떻게 찾을 수 있을지에 대해서는 고려해야 한다.

수정된 SIMD 아키텍처는 다음과 같은 점에서 기본 아키텍처와 차이가 있다.

- 2 개의 레지스터 파일
- 2 개의 데이터 메모리

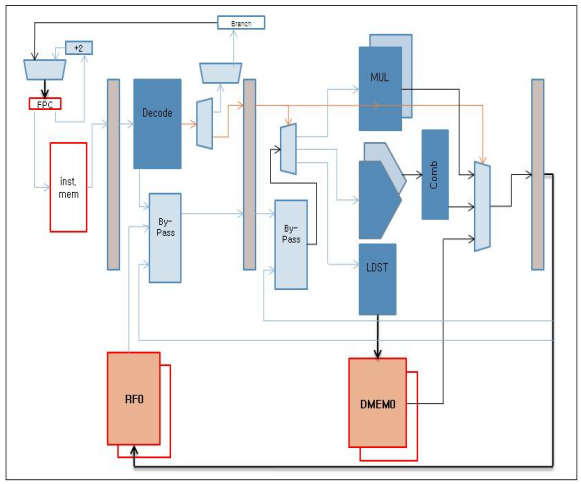

그림 2 Simple SIMD architecture

4. 결론

미디어 어플리케이션의 커널 코드를 효과적으로 처리하기 위해서는, 일반적인 DSP co-프로세서 아키텍처에 SIMD 구조를 추가하는 것이 효과적이다. SIMD 구조가 추가될 경우 여러 가지 제약이 있을 수 있지만, SIMD 구조를 보다 더 간단하도록 수정하면 이러한 제약을 줄일 수 있다.

참고문헌

[1] **LISA – Machine Description Language for Cycle-Accurate Models of Programmable DSP Architectures**. S. Pees, A. Hoffmann, V. Zivojnovic, and H. Meyr. In *Proc. of the Design Automation Conference (DAC)*, New Orleans, June 1999.

[2] **Architecture Implementation Using the Machine Description Language LISA,** Oliver Schliebusch, Andreas Hoffmann, Achim Nohl, Gunnar Braun and Heinrich Meyr, Design Automation Conference, 2002. Proceedings of **7th Asia and South Pacific and the 15th International Conference on VLSI Design. Proceedings. (ASP-DAC 2002)**

[3] **A Novel Methodology for the Design of Application Specific Instruction Set Processors (ASIP) Using a Machine Description Language.** A. Hoffmann, A. Nohl, G. Braun, O. Schliebusch, T. Kogel, and H. Meyr. *IEEE Transactions on Computers-Aided Design(TCAD)*, Nov. 2001.

[4] **Computer Architecture: A Quantitative Approach**, John H. Hennessy, David A. Patterson, MORGAN CAUFMANN, 3rd Edition.

[5] **Wikepedia**, URL http://wikipedia.org/wiki

Acknowledge

본 연구는 교육과학기술부/한국과학재단 우수연구센터육성사업(R11-2008-007-01001-0), 지식경제부 출연금으로 ETRI, SoC 산업진흥센터에서 수행한 ITSoC 핵심설계인력양성사업, 서울시 산학연 협력사업, 2008 년도 정부(교육과학기술부)의 재원으로 한국과학재단의 국가지정연구실사업(R0A-2008-000-20110-0), 지식경제부 및 정보통신연구진흥원의 대학 IT 연구센터 지원사업(IITA-2008-C1090-0801-0020), 지식경제부 및 정보통신연구진흥원의 IT 원천기술개발사업[과제관리번호: 2006-S-006-02, 과제명: 유비쿼터스 단말용 부품/모듈]의 지원을 받아 수행되었습니다.