SIMD 명령어가 추가된 VLIW ASIP 프로세서

양승준, 박상현, 허인구, 이종원, 김용주, 백윤흥
서울대학교 공과대학 전기컴퓨터 공학부
e-mail : sjyang, shpark, igheo, jwlee, yjkim@optimizer.snu.ac.kr, ypaek@snu.ac.kr

SIMD Extended VLIW ASIP architecture

Seungjun Yang, Sanghyun Park, Ingoo Heo, Jongwon Lee, Yongjoo Kim,
Yunheung Paek
School of Electrical Engineering, Seoul National University

요 약

VLIW 아키텍처는 동시에 여러 개의 명령어를 수행하면서도 상대적으로 크기가 작으며 적은 전력을 소모한다는 장점 때문에 임베디드 어플리케이션을 처리하기 위해 많이 쓰이고 있다. 본 논문에서는 SIMD 명령어를 추가한 VLIW 아키텍처를 설계함으로써 동영상 처리와 같은 미디어 어플리케이션을 효과적으로 처리할 수 있도록 하였다.

1. 서론

최근 들어 임베디드 시스템에서 어플리케이션을 처리할 때 VLIW(Very Long Instruction Word) 프로세서를 사용하는 경우가 빈번해지고 있다. VLIW 아키텍처는 동시에 여러 개의 명령어를 처리할 수 있으며, Superscalar 아키텍처와 달리 복잡한 하드웨어 구조를 필요로 하지 않기 때문에 상대적으로 작은 크기를 가지며 보다 적은 전력을 소모한다는 장점이 있다. 때문에 복잡한 연산을 정해진 시간 내에 빠르게 처리해야 하는 미디어 어플리케이션을 임베디드 시스템에서 구동시켜야 하는 경우, VLIW 아키텍처는 효과적인 답이 될 수 있다.

본 논문에는 임베디드 시스템에서 미디어 어플리케이션을 효과적으로 처리하기 위한 VLIW 아키텍처를 소개한다. 또한 병렬성을 보다 더 높이기 위해 SIMD(Single Instruction Multiple Data) 명령어를 추가하였음을 설명한다. 먼저 2 장에서는 기본 VLIW 아키텍처의 특성 및 구조를 소개하고, 3 장에서는 추가된 SIMD 명령어에 대해 설명한다. 4 장에서는 향후 연구 방향에 대해서 고찰하고, 5 장에서는 전체 내용을 요약하고 결론을 제시한다.

2. 기본 VLIW 아키텍처

본 논문에서 제안한 프로세서는 4-issue VLIW 아키텍처로써, 각 슬롯마다 32 bit 의 명령어를 읽어 들여 동시에 처리한다. 프로세서의 기본적인 특성은 다음과 같다.

- 디지털 신호 처리를 위한 DSP co-프로세서
- 32bit 데이터 및 명령어 path
- 5 단계 파이프라인 (FE, DC, MEM, EX, WB)
- 1 개의 레지스터 파일 (32 개 레지스터)
- 4 개의 내부 메모리

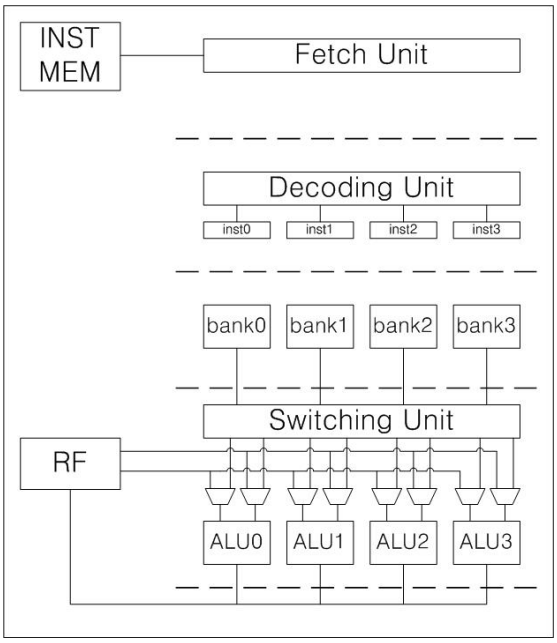

그림 1 기본 아키텍처

그림 1 은 기본 VLIW 아키텍처의 구조를 그림으로 나타낸 것이다. 각각의 이슈 슬롯은 대부분의 명령어를 실행할 수 있으며, 비교 또는 분기 명령어, 함수 호출과 같은 명령어는 첫 번째 이슈 슬롯에서만 실행될 수 있다.

3. 추가된 SIMD 명령어

본 논문에서 제안하는 프로세서는 4 개의 이슈 슬롯을 가지고 있어, 동시에 최대 4 개의 명령어를 실행할 수 있다. 이러한 각각의 이슈 슬롯에 동시에 여러

개의 데이터를 처리할 수 있는 SIMD 명령어를 추가하게 되면 어플리케이션의 병렬성을 보다 더 크게 높일 수 있다. 이 장에서는 VLIW 아키텍처에 추가된 SIMD 명령어에 대해서 설명한다.

- load16d, store16d

load16d 명령어는 메모리에서 32bit 값을 읽어올 때, 하나의 32bit 값이 아닌 연속된 2 개의 16bit 값으로 인식하여 레지스터에 저장한다. 이 때 메모리 주소가 낮은 값이 레지스터의 하위 16bit 에, 메모리 주소가 높은 값이 상위 16bit 에 저장된다. 마찬가지로 store16d 명령어는 레지스터에 저장된 두 개의 16bit 데이터 중 하위 16bit 에 저장된 값은 메모리의 낮은 주소에, 상위 16bit 에 저장된 값은 높은 주소에 기록한다.

- load8q, store8q

이 명령어들은 메모리에서 값을 읽어오거나 메모리에 값을 기록할 때, 연속된 4 개의 8bit 데이터로 인식한다. 값을 읽어올 때는 메모리 주소가 낮은 값부터 레지스터의 하위 bit 에 저장하고, 값을 쓸 때는 레지스터의 하위 bit 에 저장된 값부터 메모리의 낮은 주소에 기록한다.

- BIOP16d, BIOP8q

이 명령어들은 add, sub 와 같은 바이너리 연산을 SIMD 연산으로 수행한다. 예를 들어, “add16d r0, r1, r2”와 같은 명령어는 r1 레지스터의 하위 16bit 과 r2 레지스터의 하위 16bit 을 더해 r0 레지스터의 하위 16bit 에 저장하는 동시에, r1 레지스터의 상위 16bit 과 r2 레지스터의 하위 16bit 을 더해 r0 레지스터의 상위 16 bit 에 저장한다.

이와 같은 SIMD 명령어를 사용하면 연속적인 16bit 혹은 8bit 데이터 연산이 필요할 때 이를 효과적으로 처리할 수 있다. 예를 들어 다음과 같은 C 코드를 있다고 하자.

```
short a[10], b[10], c[10];
for ( i = 0; i <10; i++) {
    a[i] = b[i] + c[i];
}
```

만약 SIMD 명령어가 존재하지 않는다면, 위 C 코드를 처리하는 데에는 20 개의 16 bit load 명령어와 10 개의 16bit add 명령어, 10 개의 16bit store 명령어가 필요하다. 그러나 앞서 설명한 SIMD 명령어를 사용할 경우, 처리를 위해 필요한 명령어의 개수는 SIMD load 명령어 10 개, SIMD add 명령어 5 개, SIMD store 명령어 5 개로 줄어든다. 이처럼 어플리케이션 처리를 위해 필요한 명령어의 개수가 줄어들면 코드 크기가 줄어드는 이점이 있을 뿐만 아니라, 코드의 수행 시간 또한 줄어들어 전체적인 성능이 크게 향상된다.

4. 향후 연구 방향

앞서 설명한 SIMD 명령어를 효과적으로 사용하기 위해서는 컴파일러의 도움이 필수적이다. 개발자가 SIMD 명령어를 적용시킬 수 있는 부분을 직접 일이 찾는다는 것은 불가능하기 때문에, 컴파일러가 C 코드를 컴파일하는 과정에서 SIMD 명령어를 적용시킬 수 있는 부분을 찾아내고 적용시키는 알고리즘이 필수적이다. 기존의 SIMD 구조를 찾는 알고리즘은 많이 소개가 되어 있지만, 이러한 알고리즘들이 VLIW 아키텍처에서 그대로 사용될 수 있는지는 확실치 않다. 따라서 앞으로는 VLIW 아키텍처에 사용할 수 있는 SIMD 알고리즘을 구현하여 컴파일러에 적용하는 것이 필요하다.

5. 결론

VLIW 프로세서는 동시에 여러 개의 명령어를 처리하면서도 상대적으로 크기가 작고 적은 전력을 사용하기 때문에 미디어 어플리케이션처럼 고성능의 처리 능력이 요구되는 환경에 유용하게 사용될 수 있다. 본 논문에서는 이러한 VLIW 프로세서에 SIMD 명령어를 추가하여 병렬성을 높임으로써 성능 향상을 꾀할 수 있다.

참고문헌

[1] Heng Liao, A. Wolfe, "Available parallelism in video applications," Microarchitecture, IEEE/ACM International Symposium on, pp. 321, 30th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'97), 1997.

[2] TMS320C6000 CPU and Instruction Set Reference Guide, Texas Instruments Inc., Dallas, TX, 2000.

[3] Zheng Shen, Hu He, Yanjun Zhang, and Yihe Sun, “A Video Specific Instruction Set Architecture for ASIP design,” VLSI Design, vol. 2007, Article ID 58431, 7 pages, 2007. doi:10.1155/2007/58431

[4] M. Hohenauer, C. Schumacher, R. Leupers, G. Ascheid, H. Meyr, H. van Someren, “Retargetable code optimization with SIMD instructions”, Proceedings of the 4th international conference on Hardware/software codesign and system synthesis, 2006

[5] 양승준, 박상현, 허인구, 이종원, 김용주, 백윤흥, ” H.264 정수변환 및 양자화를 위한 VLIW ASIP 프로세서”, 한국정보처리학회 추계학술발표대회 제 16권 2호, 2009

Acknowledgement

본 연구는 교육과학기술부/한국과학재단 우수연구센터 육성사업(과제번호 2010-0001724), 서울시 산학연 협력사업(10560), 2010 년도 정부(교육과학기술부)의 재원으로 한국과학재단의 국가지정연구실사업(No.2010-0018465) 및 IDEC 의 지원을 받아 수행되었습니다.