멀티미디어 애플리케이션 처리를 위한 ASIP

Application Specific Instruction Set Processor for Multimedia Applications

융합기술시대의 ICT 부품 연구동향 특집

이재진 (J.J. Lee) 멀티미디어프로세서설계팀 선임연구원
박성모 (S.M. Park) 멀티미디어프로세서설계팀 팀장
엄낙웅 (N.W. Eum) 시스템반도체연구부 부장

목 차

최근 모바일 멀티미디어 기기들의 사용이 증가하면서 고성능 멀티미디어 프로세서에 대한 필요성이 높아지고 있는 추세이다. DSP 기반의 시스템은 범용성에 기인하여 다양한 응용 분야에서 사용될 수 있으나 주문형반도체 보다 높은 가격과 전력소모 그리고 낮은 성능을 가진다. ASIP는 주문형반도체의 저비용, 저전력, 고성능과 범용 프로세서의 유연성이 결합된 새로운 형태의 프로세서로서, 단일 칩 상에 H.264, VC-1, AVS, MPEG 등과 같은 다양한 멀티미디어 비디오 표준 및 OFDM과 같은 통신 시스템을 지원하고 또한 고성능의 처리율과 계산량을 요구하는 차세대 비디오 표준의 구현을 위한 효과적인 해결책으로 주목되고 있다. 본 기술 문서에서는 ASIP의 특징과 애플리케이션의 가속 방법, ASIP을 위한 컴파일러 설계 및 응용에 관하여 기술한다.

Ⅰ. 서론

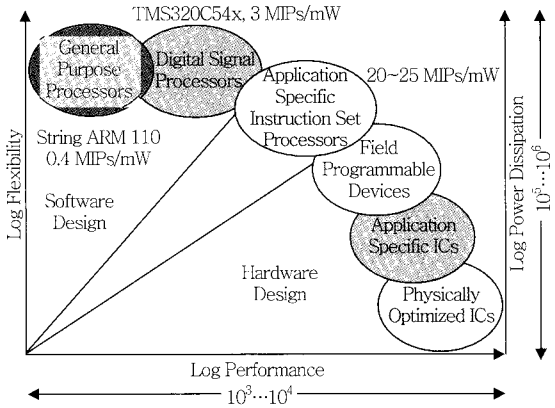
최근 모바일 멀티미디어 기기들의 사용이 증가하면서 고성능의 멀티미디어 프로세서에 대한 필요성이 높아지고 있는 추세이다. 멀티미디어 프로세서란 낮은 소비 전력으로 실시간 고성능의 멀티미디어 애플리케이션을 구현하는 프로세서이다. ARM과 같은 임베디드 프로세서나 DSP는 범용성에 기인하여 그 응용범위가 넓다는 장점이 있지만, 특정 멀티미디어 애플리케이션의 실시간 처리를 위한 성능을 만족시키지 못한다. 따라서 실시간 멀티미디어 애플리케이션의 처리를 위해 주문형반도체의 고성능과 범용프로세서(GPP)의 유연성이 결합된 ASIP가 효과적인 해결책으로 주목되고 있다[1].

현재 국내에서는 다양한 내장형 멀티미디어 프로세서들을 자체 개발하지 못하고 있는 실정이다. 이는 RTL 이하의 하부 구조 기술은 어느 정도 확보되어 있으나, 상위수준의 시스템 설계 기술이 부족하고 이를 뒷받침 해주는 툴과 상·하위 기술 간의 연계에 대한 체계적인 방법론이 결여되어 있으며 특히 고급 기능을 가진 지능형 컴파일러 개발에 큰 어려움이 있기 때문이다.

Ⅱ. ASIP

ASIP는 (그림 1)[2]에 보이는 것처럼 주문형반도체의 저비용, 저전력, 고성능과 범용 프로세서의 유연성(flexibility)이 결합된 새로운 형태의 프로세서로서, 특정 애플리케이션의 처리를 위해 설계되지만 주문형반도체와는 다르게 설계자가 원하는 특정 명령어 세트와 레지스터 파일 및 내부 인터페이스 등을 포함하고 있으며 고유의 컴파일러를 가진 프로그램 가능 가속기(accelerator)이다.

ASIP의 유용성은 단일 칩 상에 약 20~25 MIPs/mW의 성능이 요구되는 H.264, VC-1, AVS, MPEG-4, MPEG-2 등과 같은 다양한 멀티미디어 비디오 코덱 및 OFDM과 같은 통신 시스템[3]을 지원하고



<자료>: CoWare Inc.

(그림 1) Flexibility vs. Performance

또한 최근 반도체 시장의 수요 변화에 신속히 대응할 수 있으므로 고성능의 처리율과 계산량을 요구하는 차세대 비디오 표준의 구현에 그 응용이 확대될 전망이다.

ASIP의 데이터 패스는 높은 병렬성(massive parallelism)과 특정한 태스크의 처리를 위해 설계되며, 기능 유닛(functional unit)의 주문형반도체 구현과 비슷하게 병렬적으로 수행될 수 있을 뿐 아니라 내부의 커뮤니케이션을 위해 전용의 레지스터들을 사용할 수 있다. 그러나 기능 유닛의 제어(control)는 고정되어 있지 않고 프로세서처럼 명령어 디코더와 프로그램에 의해 수행된다. 이런 점은 서로 다른 응용 프로그램과 알고리듬을 처리하기 위해 기존의 ASIP가 재사용되는 것을 가능하게 한다. 플랫폼에서 ASIP의 인터페이스와 제어 방법은 하드와이어드 가속기의 설계 방법과 비슷하게 구현될 수 있다.

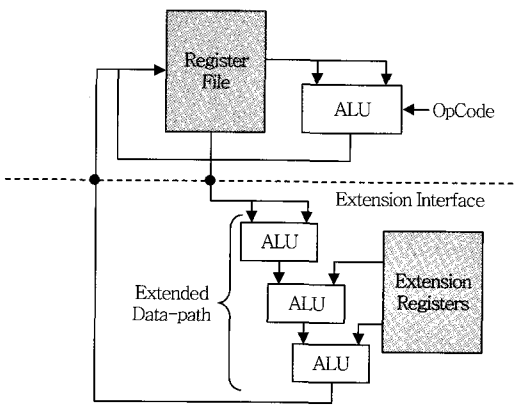
Ⅲ. 애플리케이션 가속

범용 프로세서에서는 멀티미디어 처리를 위한 특정 명령어가 널리 사용되지 않는다. 그 이유는 제공되는 특정 명령어가 다양한 응용분야에서 효율적으로 사용될 수 없기 때문이다. 그러나 애플리케이션 특정 명령어는 ASIP가 대상으로 하는 응용 분야에 적합하게 최적화 될 수 있기 때문에 효율적인 사용이 가능하다[4].

ASIP에서 멀티미디어 애플리케이션의 가속 구현은 프로세서에 멀티미디어 명령어를 추가하여 확장하는 방법, 프로세서의 전형적인 병목현상(bottleneck)인 load/store를 줄이기 위한 전용 레지스터와 데이터 수준 병렬성을 높이기 위한 SIMD 구조를 적용하는 방법, VLIW과 같은 명령어 수준 병렬성(instruction-level parallelism)을 사용하는 방법으로 구분된다.

1. 명령어 확장

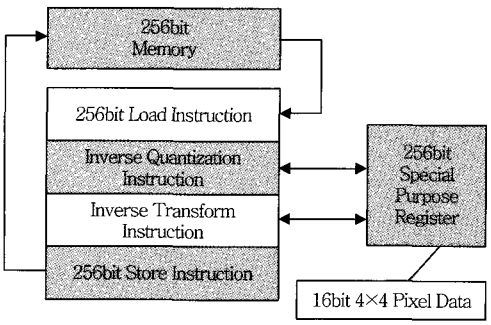
(그림 2)에 보이는 것처럼 기본 명령어 셋에 애플리케이션 전용의 복잡한 명령어 셋이 결합된 개념에 바탕을 둔 명령어 확장은 RISC와 DSP 타입의 구조(architecture)에서 수행되는 소프트웨어의 성능을 향상시킬 수 있는 강력한 수단으로 필터 연산의 경우 5배에서 10배의 프로세서 수행 사이클을 감소시킬 수 있다. 그러나 데이터의 공급과 저장, 즉 load/store가 주요한 병목현상으로 나타난다. 일반적으로 명령어 확장은 메인 프로세서의 파이프라인과 용이한 결합을 위해 메인 프로세서의 범용 레지스터 파일을 기반으로 동작된다. 따라서 피연산자(operand)는 명령어 수행 이전에 메모리로부터 레지스터 파일에 적재되며 명령어 수행 후 다음 명령어에서 결과값이 사용될 수 있도록 다시 레지스터 파일에 저장된다. 결과적으로 명령어 확장을 이용한 성능 향상은 데이터의 load/store 병목현상에 제한을 받게 된다.



(그림 2) 명령어 확장을 지원하는 전형적인 ALU 구조

2. 데이터 병렬성

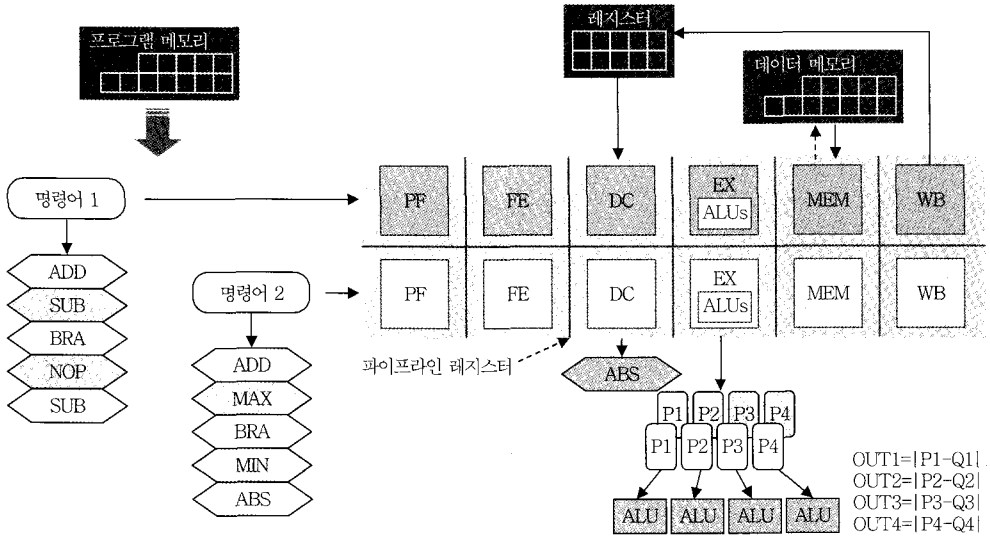
하드웨어가 소프트웨어에 비해 가지는 성능의 이점은 데이터가 병렬적으로 수행된다는 점이다. 대표적인 멀티미디어 애플리케이션인 비디오 디코더의 경우 16비트 16(4×4)개의 픽셀데이터를 하나의 벡터 타입 기능 유닛에 의해 처리할 수 있으며 프로세서 영역에서 이런 타입의 처리를 SIMD라고 부른다. (그림 3)은 H.264 디코더의 역양자화 및 역변환을 위한 SIMD 구조를 보인다. SIMD 구조에서는 데이터의 load/store에 의한 병목현상을 해결하기 위해 전용 레지스터와 결속된(tightly coupled) 지역 메모리(local memory)의 사용이 요구된다. 벡터 타입 기능 유닛은 256비트(16비트×16) 데이터를 전용 레지스터로부터 입력 받으며 연산 후 결과값을 다시 전용 레지스터에 저장한다. 지역 메모리는 전용 레지스터와 동일한 256비트의 입·출력 대역폭(bandwidth)을 가지며, 프로세서는 256비트 load/store SIMD 명령어를 통해 지역 메모리에 있는 데이터를 전용 레지스터로 적재하거나 전용 레지스터의 데이터를 지역 메모리로 저장한다.



(그림 3) SIMD 구조(예. H.264 디코딩)

3. 명령어 수준 병렬성

VLIW 구조는 대량의 데이터를 처리하는 멀티미디어 애플리케이션에 매우 적합한 구조로서, 높은 명령어 수준의 병렬 처리를 가능하게 한다. (그림 4)는 2개의 슬롯을 가지는 VLIW 구조를 예로 보인 것이다. 명령어 하나를 32비트로 가정할 때 프로세서는 프로그램 메모리로부터 32비트 명령어 2개를 읽



(그림 4) Dual Issue VLIW + SIMD 구조

어서 병렬로 수행한다. (그림 4)의 'ABS' 명령어는 4개의 8비트 데이터에 대한 각각의 절대값 연산을 병렬적으로 수행하는 SIMD 연산을 수행한다. VLIW는 병렬 연산의 동기화가 컴파일 시간에 수행되므로 프로세서의 효율이 향상되며 또한 명령어 수준 병렬처리가 가능하고 데이터 이동이 컴파일 시간에 명시되어 실시간 자원 스케줄링이 간단하다는 장점을 가진다. 그러나 효율적인 스케줄링을 위한 지능적인 컴파일러가 요구된다. 최근 도시바는 32비트 RISC와 64비트 SIMD Dual Issue VLIW가 결합된 MPE[4]를 사용하여 H.264, MPEG-4 디코더 및 ACC와 같은 멀티미디어 애플리케이션 구현 후 성능 결과를 발표하였다[5].

Ⅳ. 컴파일러 설계

컴파일러는 C와 같이 고급언어로 쓰여진 프로그램을 그와 의미적으로 동등하며 컴퓨터에서 즉시 실행될 형태의 목적 프로그램으로 번역해주는 프로그램이다. C 언어는 일반적으로 어셈블러의 상위 수준 변형(variant)으로 간주되며, ASIP을 위한 C 컴파일러 설계[6]는 아래와 같이 3개의 태스크로 구분된다.

• C 언어의 모든 데이터 타입과 연산자를 프로세서에서 지원되는 어셈블리 코드로 매핑한다.

이것은 C 언어에서 사용 가능한 기본 연산자인 +, -, × 등과 int, long, float와 같은 데이터 타입을 포함한다. 이 단계는 C 언어를 완벽히 지원하기 위해 필수 불가결한 과정이다. 컴파일러는 기본적으로 모든 가능한 연산자와 데이터의 조합을 지원해야 한다. RISC와 비슷한 구조에서 위 과정은 매우 단순한 1:1 매핑으로 간주될 수 있지만, 제한된 명령어 또는 데이터 모델을 갖는 복잡한 구조에서는 매우 복잡할 수 있다. 프로세서 구조에 없는 연산의 경우는 기본적인 연산을 이용하여 차선책(workaround)으로 구현된다. 예를 들어 C 언어의 곱셈 연산은 어셈블리 언어의 'add-loop' 구조로 에뮬레이션 될 수 있다. C의 데이터 모델에서 리소스는 무제한으로 간주된다. 프로그래밍 과정 중 프로그래머는 임의 사이즈의 변수를 임의 개 선언할 수 있다.

그러나 프로세서는 각각 서로 다른(액세스 속도, 사이즈) 제한된 메모리와 레지스터를 가지므로 컴파일러는 C의 변수에 리소스를 할당할 때 이런 점들을 고려해야 한다. 이런 할당(allocation)을 최적화하는 것이 컴파일러 설계의 가장 어려운 점이다.

- Re-entrant 함수 호출 지원
- MAC, SIMD 명령어와 같이 ANSI-C에서 제공되지 않지만 프로세서에서 지원되는 특수 명령어를 어셈블리 코드로 매핑한다.

  이런 특정 명령어를 지원하기 위해서 컴파일러의 매핑 룰을 추가하는 방법과 CKF[1] 및 인라인 어셈블리 코드를 사용하는 방법이 있으며, 매핑의 용이성과 최적화 성능 측면에서 이점을 가지는 CKF가 널리 이용된다.

V. 결론

고성능의 처리율과 계산량을 요구하는 멀티미디어 애플리케이션의 효율적인 구현을 위해 주문형반도체의 고성능과 범용 프로세서의 유연성이 결합된 ASIP가 아래와 같은 응용 분야에서 널리 사용되고 있으며, 현재 재구성 가능한 ASIP에 대한 연구가 활발히 진행되고 있다.

- "Baseband Architecture for Multi-Standard Cell Phones," Infineon
- "Multi-Standard Video Decoder Deblocking Filter IP," Philips
- "RISC Processor for Control and Multiple SIMD Engines for Video Streaming," Zoran
- "MACC-Processor for Wireless Systems in SW Radio," Intel
- "Xtensa Customizable Processor," Tensilica

● 용 어 해 설 ●

멀티미디어 프로세서: PC나 휴대용 기기에서 비디오, 오디오, 데이터 등과 같은 멀티미디어 처리를 일괄적으로 실행할 수 있는 전용 프로세서로써, 범용 프로세서에 파이프라인 확장이나 멀티미디어 명령어 추가, 분기 예측 버퍼 탑재 등을 통해 멀티미디어 기능이 강화됨

주문형반도체: 사용자가 특정용도의 반도체를 주문하면 반도체 업체가 이에 맞춰 설계 및 제작해 주는 반도체로서, 주로 정보통신기기에 사용되어 왔으나 최근 들어 가전 및 모바일 제품 등으로 용도가 급속하게 확대되고 있음

약어 정리

ASIP	Application Specific Instruction Set Processor
CKF	Compiler Known Function
DSP	Digital Signal Processor
GPP	General Purpose Processor
MAC	Multiply and Accumulate
OFDM	Orthogonal Frequency Division Multiplexing
RTL	Register Transfer Level
SIMD	Single Instruction Multiple Data
VLIW	Very Long Instruction Word

참 고 문 헌

[1] "Processor and Compiler Designer Reference Manual," *CoWare*, 2009.

[2] Achim Nohl, "Designing with the Flexibility of SW and the Efficiency of HW," *CoWare Technical Presentation*, 2009.

[3] Jae H. Baek, Sung D. Kim, and Myung H. Sunwoo, "SPOCS: Application Specific Signal Processor for OFDM Communication Systems," *Journal of Signal Proc. Systems*, Vol.53, No.3, Dec. 2008, pp.383-397.

[4] W. Zhao and C.A. Papachristou, "An Evolution Programming Approach on Multiple Behaviors for the Design of Application Specific Programmable Processors," *European Design & Test Conf.*, 1996, pp.144-150.

[5] Shouou Nomuar et al., "A 9.7mW AAC-Decoding, 620mW H.264 720p 60fps Decoding, 8-Core Media Processor with Embedded Forward-Body-Biasing and Power-Gating Circuit in 65nm CMOS Technology," *ISCC 2008*, Feb. 2008, pp.262-263.

[6] J. Teich and R. Weper, "A Joined Architecture/Compiler Design Environment for ASIPs," *In Proc. of the Conf. on Compilers, Architectures and Synthesis for Embedded Systems (CASES)*, Nov. 2000.