특집

VLIW DSP 프로세서 아키텍처의 설계 기술

이상정 (순천향대학교 공과대학 정보기술공학부)

I. 서 론

DSP(digital signal processor) 아키텍처는 FFT, 디지털 필터링 등과 같은 특정의 규칙적인 계산 지향 태스크를 염두에 두고 설계된 아키텍처이다. 초기 DSP 프로세서 아키텍처들은 MAC (multiply-accumulate) 유닛, ALU, 시프터 및 특정 타겟 알고리즘에 적합한 하드웨어로 구성되었고, 한 명령 워드에 다수의 오퍼레이션들을 기술하고 수행하는 복잡한 명령 형식을 취하였다. 그러나 1990년대 중반 이후 더욱 향상된 성능과 빠른 프로그램 개발 시간을 요하는 멀티미디어 및 이동통신 등의 새로운 응용들이 대두됨에 따라 사이클 당 다수의 오퍼레이션을 병렬처리하여 성능을 향상시키면서 기존의 어셈블리 언어가 아닌 고급 언어의 사용이 필요해졌다. 따라서 컴파일러의 코드 생성이 용이하도록 단순한 명령으로 구성된 DSP 프로세서 아키텍처가 요구되었다. 이러한 DSP 응용의 변화에 부응하기 위해 대두된 DSP 프로세서 아키텍처가 VLIW(very long instruction word) 아키텍처이고, 1996년 TI(Texas Instruments) 사의 TMS320X62xx 아키텍처 이후 많은 DSP 프로세서가 VLIW 아키텍처로 개발되어 사용되고 있다[1].

VLIW 아키텍처는 하나의 긴 명령에 다수의 오퍼레이션들을 기술하여 명령수준에서 병렬처리(instruction level parallelism, ILP)를 수행하는 아키텍처이다. 명령수준 병렬성(이하 ILP)을 이용하여 성능을 향상시키는 측면에서 현재 대부분의 범용 프로세서 아키텍처에서 채택되고 있는 슈퍼스칼라(superscalar) 프로세서 아키텍처와 유사하지만, 슈퍼스칼라 프로세서는 하드웨어가 동적으로 ILP를 검출하여 명령들을 스케쥴링하는 반면 VLIW 아키텍처에서는 컴파일러가 정적으로 ILP 검출을 위한 스케쥴링을 한다는 점이 크게 다르다[2][3]. VLIW 아키텍처는 단순한 하드웨어 구조를 가지고 사이클 당 다수의 오퍼레이션을 수행할 수 있어서 DSP 프로세서로 널리 사용되고 있다.

또한 최근에는 VLIW 아키텍처에 SMT (simultaneous multithresding) 기능을 결합하여 성능을 극대화하고자 하는 연구가 시도되고 있다[5][6]. SMT는 각 사이클에 다수의 쓰레드로부터 가능한 한 많은 명령을 이슈하여 최대의 병렬성을 추출함으로써 프로세서 효율을 개선하는 방식이다[4]. 따라서 VLIW와 SMT를 결합하여

서로 독립적인 DSP 응용들이나 프로그래머나 컴파일러에 의해 독립적인 쓰레드로 병렬화될 수 있는 DSP 응용들로부터 높은 ILP를 추출하여 성능향상을 시도하고 있다.

본 고에서는 VLIW DSP 프로세서 아키텍처의 설계 기법 및 동향에 대하여 살펴본다. 또한 VLIW DSP 프로세서 아키텍처에 더 많은 ILP를 제공하여 성능을 향상키고자 시도되는 최근의 SMT VLIW DSP 프로세서 아키텍처의 연구도 소개한다.

II. 아키텍처 관련기술

이 장에서는 아키텍처의 관점에서 본 DSP 응용들의 특징과 전형적인 VLIW 아키텍처와 SMT 아키텍처의 특성을 살펴본다.

1. 아키텍처 관점에서의 DSP 응용

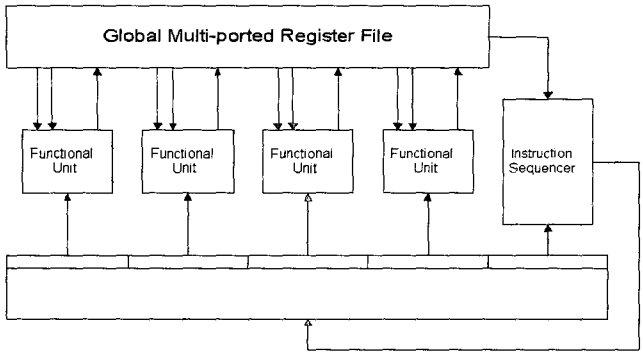
범용 프로세서는 SPEC 벤치마크들과 같은 범용의 응용 프로그램들을 타겟으로 개발된다. 이들 전통적인 프로그램들은 비교적 제한된 ILP를 가지면서 컴파일러가 예측하기 어려운 많은 분기명령으로 프로그램이 구성된다. 또한 공간 및 시간 국부성(spacial and temporal locality)을 가지고 특정 영역의 메모리를 반복적인 패턴으로 액세스하여 전통적인 다단계 캐시 메모리 구조에 적합한 메모리 액세스 패턴을 가지고 있다. 반면에 DSP나 멀티미디어 응용들은 이미지의 픽셀 등과 같은 대용량 데이터 상에서 수행되는 서로 독립적인 오퍼레이션의 반복으로 구성되어서 높은 수준의 ILP를 갖는다. 또한 대부분의 수행시간을 작은 내부루프(innermost loop)에서 수행되는 규칙적인 제어흐름을 가져서 컴파일러가 휴리스틱이나 프로파일링 등을 통하여 분기의 방향을 예측할 수 있다. 그리고 대부분의 DSP 응용들이 코드 및 데이터의 인접된 부분을 반복 사용하는 공간 및 시간 국부성을 갖지 않고 인터리브(interleave)하게 메모리를 액세스하는 경향이 있기 때문에 전통적인 캐시 계층에 맞지 않는 메모리 액세스 패턴을 갖는다.

DSP 프로세서는 특정의 규칙적인 계산지향 태스크를 지원하도록 설계되어서 다음과 같은 점에서 범용 프로세서와 차이가 난다. 첫째, DSP 프로세서들은 자주 사용되는 오퍼레이션들을 단일 사이클에 실행할 수 있도록 특수목적의 복합 명령을 갖는다. 가장 일반적인 오퍼레이션들은 대부분의 필터 커널들에서 사용되는 a=a+b*c 형태의 곱의 누적합인 MAC 연산이다. 둘째, DSP 프로세서는 단일 사이클에 동시에 다수의 메모리의 병렬 액세스를 지원한다. 이때 생성되는 주소는 후치연산(post-incrtement), 비트반전(bit-reversed) 및 모듈로(modulo) 방식으로 생성되어 메모리를 액세스한다. 병렬 메모리 액세스를 지원하기 위해서 메모리는 분할되거나 인터리브(interleaved)하게 구성된다. 셋째, 응용 프로그램의 대부분이 작고 반복적인 루프 상에서 수행되므로 이를 지원하는 명령들(loop, decrement-and- branch instruction 등)을 제공한다.

이상과 같은 전통적인 DSP 프로세서의 기능을 포함하면서 일반 범용 프로세서의 유연성를 수용하고, ILP를 활용하여 성능의 향상시키고자 VLIW DSP 프로세서가 대두 되었다.

2. VLIW 프로세서 아키텍처

ILP를 이용하여 한 클럭 사이클에 다수의 명령을 실행하는 아키텍처 설계 방식으로는 슈퍼



〈그림 1〉 이상적인 VLIW 아키텍처 모델

스칼라 아키텍처와 VLIW 아키텍처가 있다. 슈퍼스칼라 아키텍처는 다수의 기능장치를 가지고 한 클럭 사이클에 여러 명령들을 이슈하여 병렬처리하는 아키텍처이다[3]. 슈퍼스칼라 프로세서는 컴파일 시가 아닌 실행 시 하드웨어에 의해 직렬의 명령어 열로부터 병렬로 수행될 수 있는 명령어들을 이슈하여 동적으로 스케쥴하기 때문에 기존의 아키텍처와의 목적코드 호환성이 가능하다는 장점이 있어 오늘날 범용 프로세서의 설계 방식으로 채택되고 있다. 그러나 명령의 스케쥴을 전적으로 하드웨어에 의존하므로 명령어 디코딩, 이슈, 분기예측 및 인터록(interlock)을 위한 복잡한 하드웨어 장치가 필요하다. VLIW 아키텍처는 종래 CISC(complex instruction set computer)의 수평 마이크로코드 프로세서의 개념을 확장 발전시킨 기법으로 다수의 중복된 기능 장치들의 동작이 하나의 긴 명령어의 고정된 필드에 의해 기술되어 병렬처리되는 아키텍처이다[2]. 이상적인 VLIW 아키텍처는 한 사이클 당 서로 독립적인 여러 오퍼레이션들의 병렬처리를 위하여 <그림 1>과 같이 광역 레지스터 파일에 연결된 다수의 기능장치로 구성된다. 광역, 다중 포트의 레지스터 파일에 연결된 기능장치는 명령 필드의 독립된 세트에 의해 동작이 제어된다. <그림 1>의 모델이 전형적인 VLIW 아키텍처의 형태이지만 하나의 레지스터 파일에 많은 기능장치를 연결하기 위한 다중 포트의 설계가 현실적으로 어려워서 실제 상업용과 연구용으로 개발된 많은 VLIW 아키텍처에서는 변형이 많이 되었다.

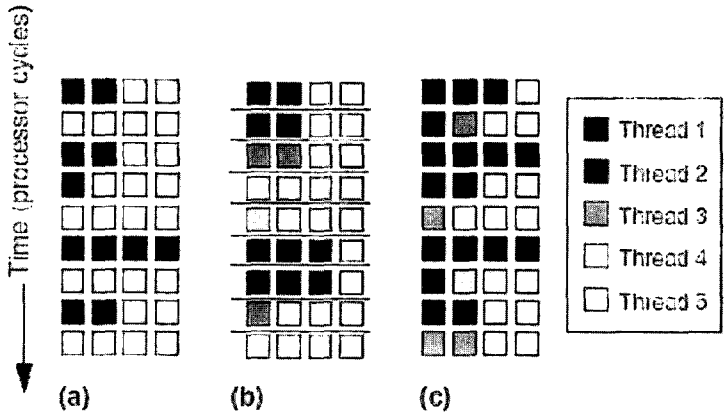
VLIW 아키텍처에서는 하나의 긴 명령어의 여러 오퍼레이션들이 다수의 기능장치를 이용하여 병렬처리 되고, 각 명령 간에는 파이프라인 처리된다. 다수의 기능장치를 사용하여 오퍼레이션들 간의 병렬처리를 위하여 최적화 컴파일러가 병렬처리되는 오퍼레이션의 스케쥴링 및 동기화를 전적으로 책임진다. 따라서 명령의 병렬수행을 위한 하드웨어 로직과 명령어 간의 인터록 하드웨어를 요구하지 않아 아키텍처의 구성이 규칙적이고 간단하다. 또한 명령어 수행 시에 거의 부작용을 갖지 않고, 각 명령의 타이밍을 포함하여 모든 프로세서의 동작은 광역 클럭에 동기되므로 명령의 실행 효과가 결정적이다. 그러므로 컴파일러가 프로그램의 실행 시 발생

되는 많은 정보를 정확히 예측할 수 있어서 실행 시에 동기화를 하지 않아도 병렬처리 되는 오퍼레이션들의 효율적인 스케쥴링이 가능하다. 그러나 VLIW 아키텍처의 성능 향상을 위해서는 병렬처리 되는 명령어 간의 데이터 종속관계, 자원 상충관계 등을 실행하기 전에 컴파일러가 정적 스케쥴해야 하므로 보다 정교한 최적화 컴파일러가 필수적인 요소가 된다. 이러한 최적화 컴파일러에 적용되는 대표적인 스케쥴 기법으로는 기본블럭을 넘어 특정 실행 경로 상에서 코드를 스케쥴링하는 트레이스 스케쥴링(trace scheduling) 등이 있다[7]. 그리고 소프트웨어 파이프라이닝(software pipelining) 기법 등을 사용하여 프로그램 루프 내에서 루프반복 간에 병렬처리될 수 있는 오퍼레이션의 패턴을 검출하여 스케쥴링한다[8].

일반적으로 DSP 응용은 규칙적으로 반복되는 연산과 데이터 흐름을 가지고 있어서 VLIW 아키텍처와 컴파일러에 적합한 구조를 갖는다. 일반 범용 응용에서 VLIW의 정적 스케쥴링은 프로그램의 동적 실행 시 발생하는 상황에 대처하지 못하지만, DSP와 멀티미디어 응용들은 상대적으로 정적흐름을 가지고 있어서 VLIW가 이상적인 선택이 된다. 또한 VLIW 아키텍처는 다수의 기능장치 제어 및 동기화를 위한 부가적인 하드웨어 없이 전형적인 DSP 코드에서 발견되는 풍부한 ILP를 활용하여 성능을 향상시킬 수 있다. 그리고 개발자가 기존의 어셈블리 코드에서 C 언어 같은 고급언어를 사용할 수 있게 되었고 그 결과 개발시간과 노력이 크게 단축하는 이점도 있다. 다만 VLIW의 가장 큰 단점은 목적코드의 호환성인데 재 컴파일과 각기 독특한 하드웨어 구조가 일반화된 DSP 분야에서 이는 큰 장애요인이 되지 못한다.

3. SMT 프로세서 아키텍처

SMT 아키텍처는 다중이슈 슈퍼스칼 프로세서와 멀티쓰레드 프로세서(multithreaded processor)의 특징을 결합한 방식이다. 슈퍼스칼라 프로세서에서 한 사이클에 다수의 명령들을 이슈할 수 있는 기능과 멀티쓰레드 프로세서에서 다수의 프로그램이나 쓰레드의 하드웨어 상태를 유지 관리하는 기능을 결합한 아키텍처이다[4]. 슈퍼스칼라 프로세서 상에서 일반 응용 프로그램들의 실행 시 낮은 ILP로 인하여 한 두개의 명령만이 병렬 수행되어 프로세서의 하드웨어를 충분히 활용하지 못한다는 문제점이 있다. 따라서 프로세서가 명령수준의 병렬성 뿐만 아니라 쓰레드수준 병렬성(thread-level parallelism) 병렬성도 활용하여 프로세서의 이용을 최대화하고 성능을 향상시키고자 SMT 프로세서 아키텍처가 제안되었다. SMT 프로세서에서 쓰레드 수준 병렬성은 멀티쓰레드를 갖는 병렬 프로그램이나 멀티프로그래밍 워크로드(multiprogramming workload) 및 독립된 개별 프로그램들로 추출되고, 명령수준 병렬성은 단일 프로그램이나 쓰레드로부터 추출된다. 즉, 각 사이클에 다수의 쓰레드로부터 다수의 명령들을 이슈하여 다양한 워크로드에서 향상된 성능을 달성할 수 있게 된다. 멀티프로그래밍과 같이 독립된 프로그램들의 혼합에서는 머신의 전체 효율이 향상되고 컴파일러나 프로그래머에 의해 병렬화된 프로그램에서는 프로그램의 성능이 향상된다. 또한 단일 쓰레드 프로그램 하나만이 수행되는 경우에도 기존의 슈퍼스칼라 프로세서에서 수행되는 것과 같은 정도의 성능이 유지된다.



〈그림 2〉 실행 시 명령의 이슈슬롯과 기능장치의 분할:
(a) 슈퍼스칼라 프로세서 (b) 멀티쓰레드 프로세서 (c) SMT 프로세서

<그림 2>는 슈퍼스칼라, 멀티쓰레딩, SMT 간의 차이를 보여주기 위해 실행 시 명령의 이슈슬롯과 기능장치의 분할을 나타내는 그림이다. 각 사각형은 하나의 명령을 표현하고 색깔은 서로 다른 쓰레드를 구분한다. <그림 2>에서 빈 사각형은 이슈슬롯 및 기능장치가 사용되지 않고 낭비됨을 표시한다. 수평공간의 낭비는 명령수준 병렬성 부족으로 각 사이클에 이슈 슬롯을 채우지 못한 것을 의미하며, 수직공간의 낭비는 긴 지연 명령(메모리 참조 등)으로 인한 명령 이슈의 지연을 의미한다. <그림 2>의 (a)는 단일 프로그램이나 단일 쓰레드를 수행하는 기존의 슈퍼스칼라 프로세서를 보여주며 수평과 수직 공간 모두에서 낭비가 발생함을 알 수 있다. (b)는 멀티쓰레드(fine-grained multitherad) 프로세서 상에서 수행을 보여주는 그림으로 한 사이클에는 동일 쓰레드로부터의 명령들을 수행하고, 다른 사이클에는 다른 쓰레드로부터의 명령들이 수행됨을 나타낸다. 즉, 멀티쓰레드 프로세서의 주요 이점은 긴 지연의 명령들에 대해서 효과적으로 대처하여 수직낭비를 제거할 수 있다는 점이다. 그러나 각 사이클에 단일 쓰레드의 명령들만 수행되어 슈퍼스칼라 프로세서와 마찬가지로 단일 쓰레드 상의 명령수준 병렬성의 부족으로 수평낭비가 존재한다. (c)는 SMT 프로세서를 보여주며 각 사이클에 다수의 쓰레드로 부터 다수의 명령들을 이슈할 수 있어서 수평 및 수직 낭비 모두를 줄일 수 있음을 보여주는 그림이다.

이와같이 SMT 프로세서는 각 사이클에 다수의 쓰레드로부터 가능한 한 많은 명령을 이슈하여 최대의 병렬성을 추출함으로써 프로세서 효율을 개선한다. 따라서 서로 독립적인 응용들이 프로그래머나 컴파일러에 의해 독립적인 쓰레드로 병렬화될 수 있는 워크로드에 대해서 성능이 향상된다. 따라서 최근에는 VLIW DSP 프로세서 상에서도 이와 같은 장점을 활용하고자 SMT를 적용하려는 연구가 시도되고 있다. 그러나 SMT를 적용하려면 각 쓰레드에 대해 프로

세서의 컨텍스트(레지스터 파일,프로그램 카운터 등)는 복사되어져야 하고 기능장치들의 집합은 쓰레드 간 공유되어야 한다.

III. VLIW DSP 프로세서 아키텍처

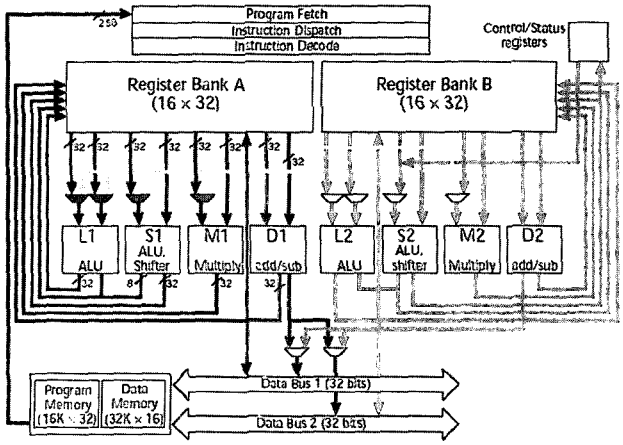
앞서 소개된 VLIW 아키텍처의 간소한 하드웨어와 컴파일러 추출에 의한 명령수준 병렬성을 이용하는 특성이 다수의 데이터 상에서 서로 독립적인 오퍼레이션을 반복 처리하는 DSP 응용에 적합하다. 이 장에서는 명령집합 구조(instruction set architecture, ISA)에 의한 2가지의 VLIW 프로세서 분류방식을 소개하고, 각 방식의 대표적인 VLIW DSP 아키텍처인 TI(Texas Instrument)의 'C6x DSP 아키텍처[9]와 Lucent/Motorolar의 StarCore SC140 VLIW DSP 프로세서 아키텍처[10]를 살펴본다.

1. 명령집합 구조에 의한 VLIW 아키텍처 분류

VLIW 오퍼레이션들은 각기 다른 지연을 갖는 경우 명령집합 구조에 따라 UAL(Unit Assumed Latencies) ISA와 NUAL(Non-Unit Assumed Latencies) ISA 두가지 방식으로 분류된다. UAL ISA는 한 명령 내의 모든 오퍼레이션들이 다음 명령이 이슈되기 전에 완료되어야 한다. 따라서 오퍼레이션들의 실제 지연은 컴파일러나 프로그래머에게 노출되지 않는다. 반면 NUAL ISA는 지연 L을 갖는 오퍼레이션에 대해 이 오퍼레이션이 완료되기 전에 다음 L-1 개의 오퍼레이션들이 이슈된다. 각 오퍼레이션의 지연은 컴파일러와 프로그래머에게 노출되어 있다. 따라서 NUAL의 VLIW 아키텍처는 각 오퍼레이션들의 지연을 알고 있어야 한다.

UAL ISA 상에서는 컴파일러나 프로그래머가 쉽게 명령을 스케쥴링 할 수 있다. 모든 오퍼레이션들이 한 사이클의 지연을 가지면 하드웨어는 매 사이클에 명령을 이슈할 수 있다. 그러나 일부 오퍼레이션이라도 한 사이클 보다 큰 지연을 가지면 하드웨어가 인터록 및 종속관계 체크 기능을 가져야만 매 사이클 명령 패킷을 이슈할 수 있다. 예를 들어 StarCore SC140은 UAL VLIW로 이전 명령 패킷에 속한 모든 오퍼레이션이 수행 완료되어야만 명령 패킷이 실행을 위해 이슈된다. 또 다른 UAL VLIW로 Analog Devices의 TigerSharc이 있다. 이 프로세서는 종속관계가 검출되면 해당 데이터가 가용할 때까지 프로그램을 정지하는 인터록 레지스터 파일들을 가지고 있다. 한 명령이 다수의 오퍼레이션으로 구성된 VLIW 아키텍처에서는 이 같은 종속관계 체크 메커니즘의 비용은 커질 수 있다. 또한 파이프라인이 깊어지면 정지로 인하여 성능손실이 커진다. 그러나 TigerSharc과 SC140 같은 UAL VLIW 프로세서들은 짧은 1 또는 2-스테이지의 실행장치를 가지고 있어 이점이 있다.

NUAL ISA의 이점은 기능유닛의 지연이 컴파일러에 노출되어 있어서 컴파일 시에 종속관계 체크와 지연을 고려한 명령 스케쥴링이 가능하다는 것이다. 또한 하드웨어에 의한 정지 메카니즘이 없어서 파이프라인을 깊게 할 수 있어 클럭 속도를 높일 수 있다. 따라서 NUAL VLIW는 UAL VLIW와 같은 복잡한 하드웨어 메커니즘 없이도 더 좋은 성능을 보인다. 그러나 컴파일러에 오퍼레이션의 지연이 노출되어 NUAL ISA와 하드웨어 구현 간 강한 결합이 요구된다. 따라서 지연이나 기능장치의 변경 등으로 인해 ISA나 하드웨어가 변경되면 코드의 호환성을 상실하는 문제점이 있다. TI의 'C6x DSP 아키텍처가



〈그림 3〉 TI 'C6x 아키텍처 프로세서 코어 구성

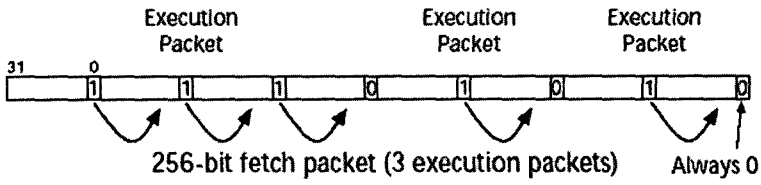
대표적인 NUAL VLIW 아키텍처이다.

2. 'C6x DSP 프로세서 아키텍처

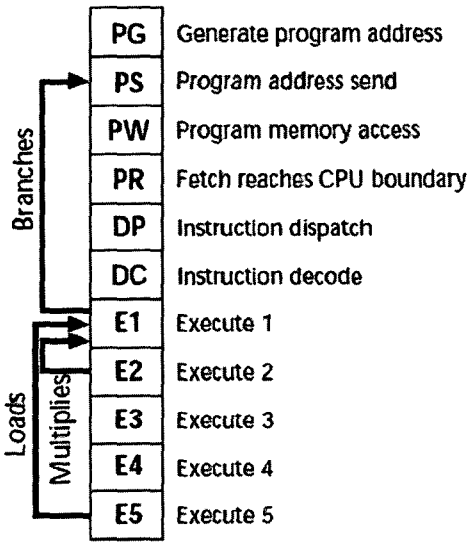
TI의 'C6x DSP 프로세서 아키텍처는 11-스테이지 파이프라인을 가진 8-웨이 VLIW DSP이고, 8개의 기능장치가 4개씩 2개의 세트로 나뉘어진 클러스터 아키텍처이다. 1997년에 발표된 TMS320C6201은 'C6x DSP 프로세서 아키텍처에 근거하여 첫 번째로 개발된 프로세서이다. 이후 2000년에 TMS320C62x 패밀리에 SIMD (single-instruction multiple-data) 처리 기능과 이동통신 관련 특수 명령어 등을 부가한 TMS320C64x 패밀리 발표되었고 2003년에 720MHz 1.4V에 동작하는 TMS320C6416을 발표하였다.

'C6x DSP 아키텍처는 <그림 3>에 나타난 바와 같이 각각 16개의 레지스터로 구성된 2개의 레지스터 뱅크 A,B가 각 기능장치 세트에 연결되어 있다. 각 기능장치는 40비트 정수 ALU(L-유닛), 40비트 시프터(S-유닛), 16비트 곱셈기(M-유닛)와 주소를 생성하는 32비트 가산기(D유닛)로 구성된다. L-유닛과 S-유닛은 1 사이클의 실행지연, M-유닛은 2 사이클 그리고 D-유닛은 5 사이클의 실행 지연을 갖는다.

'C6x DSP 프로세서는 한번에 최대 8개의 32비트 오퍼레이션들을 256 비트의 명령 버스로부터 페치한다. 이 8개의 오퍼레이션으로 구성된 긴 명령어를 TI에서는 페치 패킷(fetch packet)이라 하였다. 페치 패킷 내의 8개 오퍼레이션들은 기존의 VLIW와는 달리 위치에 관계없이 컴파일러나 프로그래머에 의해 특정 기능장치를 지정하여 인코딩된다. 페치 패킷 내의 모든 오퍼레이션들은 모두 동시에 실행될 필요는 없고 각 오퍼레이션에서의 최하위 비트(LSB)에 의해 동시에 실행되는 오퍼레이션들을 - 이들을 실행 패킷(execute packet)이라 한다 - 표시한다. 즉 각 오퍼레이션의 LSB가 세트되었으면 다음 오퍼레이션이 동시에 실행됨을 표시한다. <그림 4>는 3개의 실행 패킷으로 구성된 긴 명령을 보여주는 예이다.



〈그림 4〉 3개의 실행 패킷으로 구성된 예



〈그림 5〉 'C6x DSP 프로세서의 11개 파이프라인 스테이지

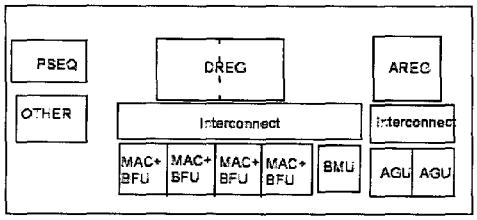
<그림 5>는 'C6x DSP 프로세서의 11개 파이프라인 스테이지를 보여주는 그림이다. 페치 단계는 프로그램 주소 생성(PG), 프로그램 주소 전송(PS), 프로그램 메모리 액세스 대기(PW)와 프로그램 페치 패킷 수신(PR) 등 4개의 스테이지로 구성된다.

디코드 단계는 명령 디스패치(DP)와 명령 디코드(DC)의 2개의 스테이지로 구성된다. DP 스테이지에서는 페치 패킷에서 실행 패킷의 오퍼레이션들이 추출되어 각 기능장치에 할당된다. 페치 패킷이 하나 이상의 실행 패킷들로 구성된 경우 페치 패킷 내의 모든 실행 패킷들이 디스패치될 때까지 페치 단계는 정지(stall)된다. DC 스테이지에서는 각 기능장치에서 오퍼레이션의 실행을 위해 소스 레지스터, 데스티네이션 레지스터 및 관련 경로들이 디코드된다. 실행 단계에서는 각 기능장치의 실행 지연에 따라 스테이지 수가 결정된다. 분기 오퍼레이션들은 5개의 지연슬롯(delay slot)을 갖는 지연분기(delayed branch)로 처리된다. 분기주소는 S-유닛에서 결정되어(E1 스테이지) PS 스테이지로 공급된다.

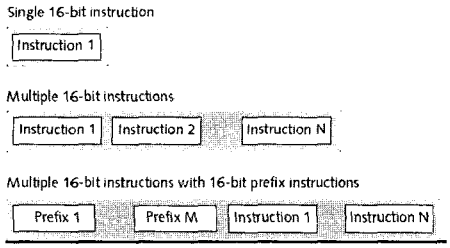
3. StarCore SC140 아키텍처

SC140 아키텍처는 Lucent/Motorola에서 1999년에 발표한 16 비트 고정소수점 VLIW DSP 코어 아키텍처이다. SC140은 이동통신, IP 전화(IP telephony), 모뎀 등 저전력 응용에 타겟을 두고 발표된 첫 번째 VLIW DSP 아키텍처이다. SC140에 기반하여 만들어진 첫 번째 프로세서는 Motorola의 MSC8101 DSP 프로세서이다. 이후 SC140에 기반을 두고 합성가능한 DSP 코어 SC1200과 SC1400이 개발되었다.

<그림 6>은 SC140 프로세서 코어의 구성을 보여주는 그림이다. <그림 6>에 나타난 바와 같이



〈그림 6〉 SC140 아키텍처 프로세서 코어 구성



〈그림 7〉 SC140 코어 아키텍처의 명령 구성 예

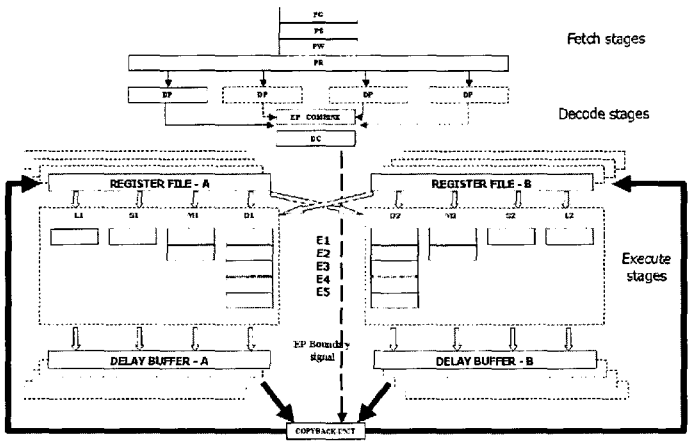
SC140은 ALU/MAC/BFU(bit-field unit)가 결합된 4개의 16 비트 데이터패스를 갖는다. 모든 데이터패스는 동일한 구조를 가지며 16개의 40 비트 데이터 레지스터(DREG)에 연결된다. 40비트보다 작은 오퍼랜드는 부호확장(sign extension)을 거쳐 40 비트로 확장된다. 데이터패스는 16 비트 곱셈을 한 사이클에 수행한다. 주소생성 유닛(address generation unit, AGU)은 두 개의 주소연산 유닛(address arithmetic unit, AAU)과 한 개의 비트마스크 유닛(bit mask unit, BMU)으로 구성되고 AGU는 16개의 주소 레지스터(address register, AREG)에 연결된다. 이 외에도 프로그램 시퀀스 로직(PSEQ)와 디버깅 로직(OTHER)으로 구성된다.

SC140은 VLIW 아키텍처로 컴파일러나 프로그래머가 명령의 병렬수행을 위해 사이클 당 6개의 명령을 스케줄한다. 각 명령 그룹은 4개의 데이터패스 오퍼레이션, 두개의 AGU (address generation unit) 오퍼레이션들로 구성된다. 명령은 16 비트 프리픽스 명령(prefix instruction)을 사용하여 기본 명령의 기능을 확장 - 예를 들어 다수의 레지스터 액세스, 명령의 조건실행(predicated execution) 등 - 할 수 있다. <그림 7>은 SC140의 명령 구성 예를 보여주는 그림이다. SC140의 명령 파이프라인은 선페치(pre-fetch), 페치, 디스패치, 주소생성, 실행 등 5개의 파이프라인 스테이지를 갖는다. 앞의 3 스테이지는 프로그램 시퀀스에 관련된 스테이지고 나머지 두 개는 데이터패스 상에서 실행되는 스테이지이다.

IV. SMT VLIW DSP 아키텍처

SMT는 각 사이클에 다수의 쓰레드로부터 가능한 한 많은 명령을 이슈하여 최대의 병렬성을 추출하여 성능을 향상시키는 기법으로 서로 독립적인 응용들이나, 프로그래머나 컴파일러에 의해 독립적인 쓰레드로 병렬화될 수 있는 응용에 적합하다. 특히 핸드폰 등과 같이 다수의 멀티미디어 응용들이 동시에 수행되는 응용에서는 큰 성능향상을 기대할 수 있다. SMT를 VLIW DSP에 적용하려면 각 쓰레드에 대한 레지스터 파일과 프로그램 카운터 등 프로세서의 컨텍스트를 지원되는 쓰레드 수만큼 중복 구성해야 하고 기능장치들의 세트는 쓰레드 간 공유되어야 한다.

이 장에서는 VLIW DSP 프로세서 아키텍처에 SMT를 적용한 연구를 소개한다. 먼저 Iyer 등이 TI의 TMS320C6201 VLIW DSP 아키텍처를 기본으로 SMT 기능을 추가한 XSI(extended spilit-issue) SMT VLIW 아키텍처[5]를 소개한다. 그리



〈그림 8〉 XSI SMT VLIW DSP 아키텍처 구성도

고 Kaxiras 등이 StarCore의 SC140 코어 아키텍처에 SMT를 적용한 StarCore SMT VLIW DSP 아키텍처 [6]를 살펴본다.

1. XSI SMT VLIW DSP 아키텍처

NUAL VLIW 아키텍처는 컴파일러가 지연을 고려한 명령 스케쥴링이 가능하고 깊은 파이프라인 처리로 높은 클럭속도를 구현할 수 있는 장점이 있는 반면, VLIW 명령들과 하드웨어 구현이 서로 밀접하게 결합되어서 ISA나 하드웨어가 일부만 변경되어도 코드의 호환성을 상실하는 문제점이 있었다. 이와 같은 문제점의 해결을 위해 Maryland 대학의 Iyer 등은 기존의 Rau에 의해 제안된 분리이슈(spilt-issue) 방식을 확장하여 하드웨어 구현과 ISA 설계를 분리할 수 있는 XSI(sXtended Split-Issue) 방식을 제안하였다. 그리고 XSI 방식에 SMT를 결합하여 제안된 방식이 다양한 하드웨어로 구성되어 성능을 향상시킬 수 있음을 보였다. XSI 방식은 N+1 사이클의 지연을 갖는 오퍼레이션들의 실행 후 결과값을 바로 레지스터로 전송하지 않고 지연버퍼(delay-buffer)에 임시로 저장한 후 N 사이클 후에 아키텍처 레지스터로 전송하는 제한된 동적 스케쥴링 기법을 제안하였다. Rau의 방식이 명령의 필드 내에 오퍼레이션의 지연값을 명시하여 코드의 호환성을 갖지 못하는 반면, XSI는 머신 사이클이 아닌 동시에 수행되는 오퍼레이션들의 집합인 명령 패킷들 수의 상대값으로 지연을 계산하여 코드의 호환성을 가질 수 있다. XSI의 이슈 하드웨어는 명령 패킷 내의 개개의 오퍼레이션을 임의의 순서로 분리하여 이슈한 후 한 명령 패킷 내의 모든 오퍼레이션들이 이슈를 완료할 때마다 버퍼의 내용을 레지스터에 전송하도록 시그널을 보낸다. XSI의 이슈 하드웨어는 한 명령 패킷 내의 오퍼레이션들만 스케쥴하고 다른 명령 패킷 간의 비순차(out-of-order) 스케쥴링은 수행하지 않는다는 점에서 슈퍼스칼라 프로세서와 구분된다.

<그림 8>은 4개의 쓰레드까지 지원하는 XSI

SMT VLIW DSP 아키텍처를 보여주는 그림이다. TI의 'C6x DSP 아키텍처를 기본 프로세서로 하고 4개의 쓰레드를 지원하기 위해 4개의 프로세서 컨텍스트 - 레지스터 파일, 프로그램 카운터 등 - 를 추가하였다. 기능장치는 'C6x 아키텍처와 동일하고 분리이슈를 지원하기 위해 각 기능장치의 출력에 4개의 지연버퍼가 추가되었다. 복사유닛(copyback unit)은 DC유닛으로부터 시그널을 받아서 지연버퍼에서 레지스터로 데이터들을 전송하는 오퍼레이션들을 스케쥴한다.

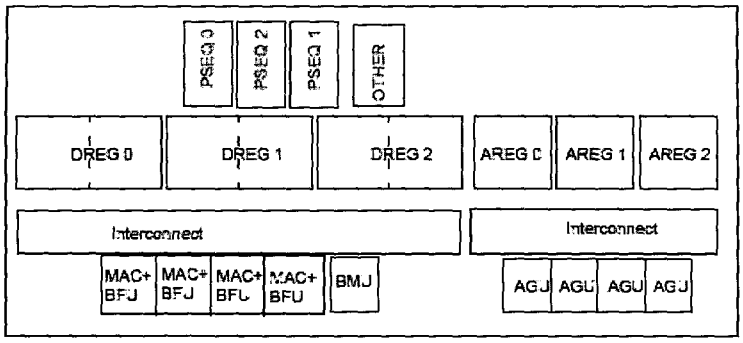
파이프라인 처리는 기본적으로 <그림 5>의 'C6x DSP 아키텍처의 파이프라인 스테이지와 같고 실행패킷-결합(EP-combine) 스테이지가 추가되었다. 명령페치 단계의 PR 스테이지에서 4개 쓰레드의 명령들(페치 패킷들)이 페치되고, 디코드 단계에서 동적 스케쥴링과 멀티쓰레딩이 수행된다. DP 스테이지에서 4개 쓰레드의 모든 명령들(실행 패킷들)이 디스패치된다. EP-결합 스테이지에서는 각 쓰레드들의 실행 패킷들로부터의 오퍼레이션들을 동적으로 스케쥴링한다. DC 스테이지에서는 각 쓰레드를 구분하는 태그가 명시된 오퍼레이션들을 받아서 디코드한 후 각 기능장치들로 보내져서 실행된다. 각 오퍼레이션들의 소스 오퍼랜드들은 명시된 태그를 참조하여 해당 쓰레드의 레지스터 파일로부터 읽혀진다. DC 스테이지에서 디코드할 때 해당 오퍼레이션이 실행패킷의 마지막에 위치한 경계 오퍼레이션인 경우 EP-경계 시그널을 복사유닛으로 보낸다. 복사유닛은 이 시그널을 받아서 지연버퍼의 내용들을 해당 쓰레드의 레지스터 파일로 전송한다.

Iyer 등은 실험 결과 XSI SMT VLIW DSP 아키텍처가 VLIW DSP 프로세서에 SMT를 결합함으로써 기능장치의 활용도를 높여서 4개의 쓰레드를 지원하는 경우 기본 'C6x 아키텍처와 비교하여 120% 정도 성능이 향상됨을 보였다.

2. StarCore SMT VLIW DSP 아키텍처

Kaxiras 등은 StarCore(SC140) VLIW DSP 상에서 차세대 무선 핸드셋에서의 동적 전력소모를 줄이는 주파수/전압의 스케일링(scaling)을 위해 SMT를 결합 적용하는 방식을 연구하였다. 즉 멀티쓰레딩으로 ILP를 증가시켜서 동일한 성능 요구 사항을 만족하면서 일반 DSP 프로세서보다도 낮은 클럭 주파수를 사용하여 공급 전압을 감소시킴으로써 전력소모를 줄이는 방식을 제안하였다. 그러나 SMT를 적용하면 프로세서 컨텍스트 등이 중복되어 칩면적이 증가되고 이는 더 높은 로드 커패시턴스(load capacitance)를 유발하여 결과적으로 동적 전력소모가 증가된다. Kaxiras 등은 SMT의 주파수/전압 스케일링으로 인한 전력소모의 감소가 칩 면적 증가로 인한 전력소모 증가보다도 커서 전체적으로 전력소모가 감소됨을 보였다.

<그림 9>는 SC140 프로세서 코어를 기반으로 하여 3개의 쓰레드를 지원하는 SMT VLIW DSP 아키텍처 구성도이다. 각 쓰레드의 상태 보존을 위하여 프로그램 시퀀스 로직(PSEQ)과 프로세서 컨텍스트(DREG, AREG 레지스터 파일) 등이 지원되는 쓰레드의 수 만큼 복사되었다. 또한 더 높은 메모리 대역폭을 지원하기 위해 주소생성 유닛(AGU)도 2배로 늘려서 구성하였다. SMT의 증가된 레지스터 파일로 인한 지연을 반영하도록 파이프라인 스테이지가 추가되었다. 부가된 파이프라인 스테이지에서는 각 쓰레드



〈그림 9〉 StarCore SMT VLIW DSP 아키텍처 구성도

가 이슈된 기능장치를 결정하였다.

Kaxiras 등은 차세대 무선 핸드셋을 타겟으로 음성과 비디오 등의 멀티미디어 응용 벤치마크에 대하여 성능을 측정하였다. 이들 응용들 - 특히 음성 - 에서는 실시간 처리가 중요하여 이들 응용들의 처리성능이 실시간 처리를 요하지 않는 다른 응용들에 영향을 받지 말아야 한다. 따라서 성능 측정 시에 실시간 제약 - 한 음성 프레임에 20ms 의 제약 - 을 지키면서 여러 멀티미디어 응용들을 동시에 수행하여 성능과 전력소모를 측정하였다. 성능 측정 결과 5개의 하드웨어 컨텍스트를 갖는 SMT VLIW DSP 아키텍처인 경우 일반 VLIW DSP 아키텍처와 비교해서 성능은 그대로 유지하면서 약 4배 정도 전력소모를 줄일 수 있음을 보였다.

V. 결 론

본 고에서는 VLIW DSP 프로세서 아키텍처의 설계 기법을 살펴보고, 대표적인 VLIW DSP 아키텍처인 TI의 'C6x DSP 아키텍처와 Lucent/Motorolar의 StarCore SC140 VLIW DSP 프로세서 아키텍처를 소개하였다. VLIW DSP 프로세서는 간소한 하드웨어와 컴파일러 추출에 의한 명령수준 병렬성을 이용하는 특성이 다수의 데이터 상에서 서로 독립적인 오퍼레이션을 반복 처리하는 DSP 응용에 적합하다.

또한 최근에는 VLIW 아키텍처에 SMT 기능을 결합하여 성능을 극대화하고자 하는 새로운 시도의 연구도 소개하였다. VLIW와 SMT를 결합하여 서로 독립적인 DSP 응용들이나, 프로그래머나 컴파일러에 의해 독립적인 쓰레드로 병렬화될 수 있는 DSP 응용들로부터 높은 명령수준 병렬성을 추출하여 성능향상이나 전력소모의 감소를 시도한 최근의 연구를 소개하였다. 이와 같이 향 후 SMT나 모험적 실행(speculative execution) 등과 같은 범용 고성능 프로세서의 아키텍처 설계 기법들을 적용하여 성능을 향상 시키면서 전력소모를 줄이는 DSP 프로세서들의 활발한 연구개발이 전망된다.

참고문헌

[1] P.Faraboschi, G. Desoli, and J.A. Fisher, "The Latest Word in Digital and Media Processing", IEEE Signal Processing Magazine, p.59-85, Mar. 1998.

[2] J.Ellis, Bulldog: A Compiler for VLIW Architecture, The MIT Press, 1986

[3] M.Johnson, Superscalar Microprocessor Design, Prentice Hall, 1991

[4] S.Eggers, J.Emer, H.Levy, J.Lo, R.Stamm, and D.Tullsen, "Simultaneous Multithreading: A Platform for Next-generation Processors", IEEE Micro, p.12-18, Sept. 1997

[5] B.Iyer, S.Srinivasan, and B.Jacob, "Extended Split-Issue: Enabling flexibility in the hardware implementation of NUAL VLIW DSPs", Proceedings of 31st International Symposium on Computer Architecture(ISCA' 04), p.364-375., June 2004.

[6] S.Kaxiras, G.Narlikar, A.Berenbaum, and Z.Hu, "Comparing Power Consumption of an SMT and a CMP DSP for Mobile Phone Workloads", Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES), p. 211-220, Dec. 2001.

[7] J.Fisher, "Trace Scheduling: A Technique for Global Microcode Compaction", IEEE Transactions on Computers, Vol.30 No.7, p.478-490, July 1981

[8] M.Lam, "Software Pipelining: An Effective Scheduling Technique for VLIW Machines", ACM Proceedings of SIGPLAN '88 Conference on Programming Language Design and Implementation, p.318-328, June 1988.

[9] J.Turley and H.Hakkarainen, "TI's New 'C6x DSP Screams at 1,600 MIPS", Microprocessor Report, Vol.11 No.2, Feb. 1997

[10] O.Wolf and J.Bier, "StarCore Launches First Architectur", Microprocessor Report, Vol.12 No.14, Oct. 1998

저자소개

이 상 정

1983년 한양대학교 전자공학과 학사
1985년 한양대학교 전자공학과 석사
1988년 한양대학교 전자공학과 박사
1999년-2000년 University of Minnesota 방문교수
1988년-현 재 순천향대학교 정보기술공학부 교수
주관심분야 고성능 마이크로프로세서 설계, 네트워크 응용