(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.[6]
G06F 9/30

(45) 공고일자 2005년06월17일
(11) 등록번호 10-0464101
(24) 등록일자 2004년12월20일

(21) 출원번호	10-1998-0705419	(65) 공개번호	10-2000-0064518
(22) 출원일자	1998년07월15일	(43) 공개일자	2000년11월06일
번역문 제출일자	1998년07월15일		
(86) 국제출원번호	PCT/US1997/001067	(87) 국제공개번호	
국제출원일자	1997년01월23일	국제공개일자	

(81) 지정국

국내특허 : 아일랜드, 일본,

EP 유럽특허 : 오스트리아, 벨기에, 스위스, 독일, 덴마크, 스페인, 프랑스, 영국, 그리스, 이탈리아, 룩셈부르크, 모나코, 네덜란드, 포르투칼, 스웨덴,

(30) 우선권주장 08/618,243 1996년03월18일 미국(US)

(73) 특허권자
어드밴스드 마이크로 디바이시즈,인코포레이티드
미국 텍사스 78741 오스틴 메일 스톱 562 이스트 벤 화이트 불러바드 5204

(72) 발명자
아스가 사프
미국 텍사스 78750 오스틴 퀼 리프 코브 7010

아이레튼 마크
미국 텍사스 78739 오스틴 록스버리 레인 6005

바트코위악 존
미국 텍사스 78746 오스틴 와일더니스 코브 1203

(74) 대리인
박장원

심사관 : 성경아

(54) DSP펑션을수행하도록의도된명령시퀀스들을DSP펑션식별자로변환하는DSP펑션프리프로세서를갖는CPU

요약

X86 코어와 같은 범용 CPU구성 요소 및 DSP 코어를 포함하는 CPU 또는 마이크로프로세서가 개시된다. 상기 CPU는 또한 지능적인 DSP 펑션 디코더 또는 프리프로세서를 포함하는바, 이는 X86 연산 코드 시퀀스들을 검사하고, DSP 펑션이 실행되고 있는 지의 여부를 결정한다. DSP 펑션 디코더가 DSP 펑션이 실행되고 있다고 결정하면, 이 DSP 펑션 디코더는 상기 연산 코드들을 DSP 매크로 명령(이는 DSP 코어로 제공된다)으로 변환 또는 맵핑한다. DSP 코어는 1개 이상의 DSP 명령들을 실행함으로써, 매크로 명령에 응답하여 필요한 DSP 펑션을 구현한다. DSP 코어는 보다 적은 수의 명령들 및 클럭 사이클들을 이용하여 DSP 펑션을 구현 또는 수행함으로써, 시스템의 성능을 향상시킨다. 명령 캐시 또는 명령 메모리의 X86 연산 코드들이 DSP 타입 펑션을 수행할 것을 나타내지 않거나 또는 DSP 타입 펑션을 수행하도록 의도되지 않는다면, 종래 기술의 컴퓨터 시스템들에서와 같이 연산 코드들은 X86 코어에 제공된다. X86 코어 및 DSP 코어는 서로 연결되어, 동기화를 위해 데이터 및 타이밍 신호들을 교환한다. 따라서, DSP 코어는 X86 코어로부터 이러한 수학적인 펑션들을 오프로드함으로써, 시스템의 성능을 향상시킨다. DSP 코어는 또한 X86 코어와 동시에 동작하기 때문에, 성능에 있어서 그 이상의 이득을 제공한다. 따라서, 본 발명의 CPU는 어떠한 부가적인 X86 연산 코드도 요구하지 않으면서 X86 논리 보다 효율적으로 DSP 펑션들을 구현한다. 본 발명은 또한 X86 코어 및 DSP 코어를 포함하는 본 발명에 따른 CPU 또는 X86만의 CPU에서 투명하게 동작하는 코드를 생성한다. 따라서, 본 발명은 기존의 소프트웨어와 백워드 호환성을 갖는다.

명세서

관련 출원

하기의 출원들은 본 발명에 관련되며, 이들은 본원의 참조로서 인용된다.

1996년 3월 18일 출원었으며 그 명칭이 "X86 및 DSP 펑션 유닛들을 갖는 중앙 처리 장치"인 미국 특허 출원 제08/618,000호,

1996년 3월 18일 출원었으며 그 명칭이 "DSP 펑션들을 위해 명령 시퀀스들을 스캔하는 DSP 펑션 프리프로세서를 포함하는 중앙 처리 장치"인 미국 특허 출원 제 08/618,240호,

1996년 3월 18일 출원었으며 그 명칭이 "DSP 펑션들을 수행하는 명령 시퀀스들을 검출하는 패턴 인식 검출기를 갖는 DSP 펑션 프리프로세서를 포함하는 중앙 처리 장치"인 미국 특허 출원 제08/618,242호, 및

1996년 3월 18일 출원었으며 그 명칭이 "DSP 펑션들을 수행하는 명령 시퀀스들을 검출하는 룩업 테이블 장치를 갖는 DSP 펑션 프리프로세서를 포함하는 중앙 처리 장치"인 미국 특허 출원 제08/618,241호.

이들 모두는 어드밴스드 마이크로 디바이시즈 인코포레이티드에 양도되었다.

기술분야

본 발명은 범용 코어 및 디지털 신호 처리(DSP) 코어를 포함하는 컴퓨터 시스템 CPU 또는 마이크로프로세서에 관한 것으로서, 이 CPU는, DSP 타입 펑션(function)들을 수행하도록 의도된 범용 연산 코드 시퀀스들을 검출한 다음 이 연산 코드들을 DSP 코어가 실행할 수 있도록 대응하는 DSP 매크로(macro)들로 변환하는 DSP 펑션 디코더를 포함한다.

배경기술

퍼스널 컴퓨터 시스템들 및 범용 마이크로프로세서들은 원래 워드프로세싱 및 스프레드시트와 같은 비즈니스 어플리케이션들을 위해 개발되었다. 그러나, 컴퓨터 시스템은 현재 수많은 실시간 DSP 관련 어플케이션들을 처리하는 데에 이용되는바, 이러한 어플리케이션들로는 특히 비디오 및 오디로 구성요소들을 갖는 멀티미디어 어플리케이션들, 비디오 캡춰 및 재생, 전화 어플리케이션들, 음성 재인식 및 합성, 그리고 통신 어플리케이션들이 있다. 이러한 실시간 또는 DSP 타입 어플리케이션들은 전형적으로 높은 CPU부동 소수점 성능을 필요로 한다.

여기에서 발생하는 하나의 문제는, 원래 비즈니스 어플리케이션들을 위해 설계된 범용 마이크로프로세서가 멀티미디어 어플리케이션들 및 통신 어플리케이션들과 같은 현대의 DSP 관련 어플리케이션들의 실시간 요건들 및 수학적인 계산 요건들에 적합하지 않다는 것이다. 예를 들어, 인텔사의 X86 패밀리의 마이크로프로세서들은 정수 기반 계산들 및 메모리 관리 동작들을 지향하고 있어, DSP 타입 펑션들은 그리 잘 수행하지 못한다.

퍼스널 컴퓨터 시스템들이 보다 실시간적이고 멀티미디어가 가능한 시스템으로 발전함에 따라, 범용 CPU는 보다 수학 집중적인 DSP타입 펑션들을 수행할 필요가 있게 되었다. 따라서, 현재 많은 컴퓨터 시스템들은 이러한 복잡한 수학적 펑션들에 전용되는 1개 이상의 디지털 신호 처리기들을 포함한다.

컴퓨터 시스템 아키텍쳐들의 현재의 추세는 원시 신호 처리(NSP)쪽으로 이동하고 있다. 처음에, 이러한 원시 신호 처리(NSP)는 DSP로부터 특정 펑션들을 오프로드(offload)한 다음 이러한 펑션들을 메인 또는 범용 CPU 내에서 수행하기 위한 방책으로서 인텔사에 의해 도입되었다. 이러한 방책은, 범용 CPU의 성능 및 클럭 속도가 증가하면, 이전에 전용 DSP에 의해 수행되었던 많은 펑션들을 범용 CPU가 수행할 수 있는 것으로 가정한다. 따라서, 마이크로프로세서 산업의 한 추세는, 보다 강력한 부동 소수점 유닛과 같은 DSP 타입 성능을 강화하고 보다 속도가 빠른 CPU를 설계하고자 하는 노력이다. 마이크로프로세서 산업의 다른 추세는 DSP 제조업자들이, 높은 속도에서 동작할 뿐 아니라 메모리 관리 기능들과 같은 CPU 타입성능들을 에뮬레이트할 수 있는 DSP를 제공한다는 것이다.

디지털 신호 처리기는 본질적으로 범용 마이크로프로세서인바, 이는 일반적으로 마이크로프로세서와 관련이 없는 속도 및 효율로 수학 펑션들을 실행하는 특별한 하드웨어를 포함한다. 현재의 컴퓨터 시스템 아키텍쳐들에서, DSP들은 코프로세서들로서 이용되며, 시스템 내의 범용 CPU들과 함께 동작한다. 예를 들어, 현재의 컴퓨터 시스템들은 메인 CPU로서 범용 CPU를 포함하며, 그리고 전용 DSP들을 포함하는 1개 이상의 멀티미디어 또는 통신 확장 카드들을 포함한다. CPU는 수학 펑션들을 디지털 신호 처리기에 오프로드함으로써, 시스템의 효율성을 높인다.

디지털 신호 처리기들은 실행 유닛들을 포함하는바, 이 실행 유닛들은 파이프라인 방식으로 복잡한 수학 알고리즘들을 구현하는 하드웨어 곱셈기에 결합된 1개 이상의 산술 논리 유닛(ALU)들을 구비한다. 명령 세트는 주로 DSP 타입 명령들을 포함하며, 비 DSP 펑션을 갖는 소수의 명령들을 또한 포함한다.

DSP는 전형적으로, 다양한 연산들중에서, 상관 관계, 컨벌루션, 유한 임펄스 응답(FIR) 필터들, 무한 임펄스 응답(IIR) 필터들, 고속 퓨리에 변환(FFr), 행렬계산 및 내적(inner product)과 같은 수학 알고리즘들에 대해 최적화된다. 이러한 수학 알고리즘들을 구현하는 데에는 일반적으로 조직적인 산술/곱셈 연산들의 긴 시퀀스를 필요로 한다. 이러한 연산들은 결정형 커맨드(decision type command)들에 의해 다양한 경우들에서 인터럽트된다. 일반적으로, DSP 시퀀스들은 70% 내지 90%의 시간으로 실행되는 매우 작은 명령 세트의 반복이다. 나머지 10%내지 30%의 시간으로 실행되는 명령들은 주로 부울(Boolean)/결정 연산들(또는 일반적인 데이터 처리)이다.

범용 CPU는 실행 유닛, 메모리 관리 유닛 및 부동 소수점 유닛 뿐 아니라 다른 논리로 구성된다. 범용 CPU의 작업은 코드를 실행하고 컴퓨터 메모리의 데이터에 대해 연산들을 수행하여, 계산 플랫폼을 관리하는 것이다. 일반적으로, 범용 CPU 아키텍쳐는 주로 부울/관리/데이터 조작 결정 연산들을 수행하도록 설계된다. 범용 CPU에 의해 실행되는 명령들 또는 연산 코드들은 기본적인 수학 평션들을 포함한다. 그러나, 이러한 수학 평션들은 복잡한 DSP 타입 수학 연산들에는 적합하지 않다. 따라서, 범용 CPU는 기본적인 DSP 평션을 수행하기 위해서는 많은 수의 연산 코드들 또는 명령들을 실행해야 한다.

따라서, 범용 CPU를 포함하고 높은 성능으로 DSP타입 수학 평션들을 수행하는 컴퓨터 시스템 및 CPU 아키텍쳐가 필요하다. 또한, 범용 CPU가 모든 수학 계산들을 수행하고 있다고 가정하는 기존의 소프트웨어 어플리케이션들과 백워드 호환성(backward-compatible)을 갖는 CPU 아키텍쳐가 필요하다. 또한, 기존의 소프트웨어 어플리케이션들에 대해 보다 높은 수학적 성능을 제공하는 새로운 CPU 아키텍쳐가 필요하다.

퍼스널 컴퓨터 시스템에서 이용되는 하나의 대중적인 마이크로프로세서는 X86 패밀리의 마이크로프로세서이다. 이 X86 패밀리의 마이크로프로세서에는 인텔사의 8088, 8086, 80186, 80286, 80386, i486, 펜티엄 및 P6 마이크로프로세서를 포함한다. X86 패밀리의 마이크로프로세서는 또한, 어드밴스드 마이크로 디바이시즈 인코포레이티드의 4486 및 K5 프로세서들, 사이릭스(Cyrix)의 M1 프로세서, 넥스트젠사(NextGen Corporation)의 NextGen 5X86 및 6X86프로세서와 같은 X86 호환성 프로세서들을 포함한다. 이러한 X86 패밀리의 마이크로프로세서는 원래 비즈니스 어플리케이션용으로 설계되고 개발된 것이다. 일반적으로, X86 패밀리의 마이크로프로세서의 명령 세트는 오늘날의 멀티미디어 및 통신 어플리케이션들에 대해 충분한 수학적 또는 DSP 평션을 갖추지 못하고 있다. 따라서, 현재의 X86 프로세서보다 효율적으로 DSP 평션들을 실시하면서도 이 X86 프로세서에 대해 추가적인 연산 코드들을 필요로 하지 않는 새로운 X86 CPU 아키텍쳐가 필요하다.

발명의 상세한 설명

따라서, 본 발명은, X86 코어와 같은 범용 CPU구성 요소를 포함하고 DSP코어를 또한 포함하는 CPU 또는 마이크로프로세서를 제공한다. 이 CPU는, 명령들 또는 연산 코드들(X86 연산 코드들)의 시퀀스를 검사하여 DSP 평션이 실행되고 있는지의 여부를 결정하는 지능적인(intelligent) DSP 디코더 또는 프리프로세서를 포함한다. DSP 평션이 수행되고 있다고 결정하면, DSP 디코더는 명령 시퀀스를, DSP 코어에 제공되는 DSP 매크로 명령 또는 평션 식별자로 변환 또는 맵핑한다. DSP 코어는 1개 이상의 DSP 명령들을 실행하여, DSP 매크로 또는 평션 식별자에 의해 지시되는 바람직한 DSP 평션을 구현한다. DSP 코어는 범용 CPU 코어에 의해 수행되는 다른 연산들과 동시에 DSP 평션을 수행한다. 또한, DSP 코어는 보다 적은 수의 명령들 및 보다 적은 수의 클럭 사이클을 이용하여 DSP 평션을 수행함으로써, 시스템의 성능을 향상시킨다.

바람직한 실시예에서, 본 발명의 CPU는 명령 메모리 또는 명령 캐시를 포함하는바, 이는 시스템 메모리로부터 마이크로프로세서 명령들 또는 연산 코드들을 수신한 다음, 이 연산 코드들을 CPU에 의해 이용될 수 있도록 저장한다. CPU는 또한 명령 시퀀스 프리프로세서라고도 불려지는 DSP 평션 디코더 또는 프리프로세서를 포함하는바, 이는 명령 캐시의 명령 시퀀스들을 분석한 다음, DPS 타입 평션이 언제 명령 시퀀스에 의해 실시 또는 표현되는 지를 지능적으로 결정한다. 이 평션 프리프로세서는 DSP 평션을 실시하는 명령 캐시 내의 명령 시퀀스들을 사전에 스캔한다.

일 실시예에서, 평션 프리프로세서는, DSP 평션을 실행하는 명령 시퀀스들을 나타내는 다수의 비트 패턴들을 저장하는 패턴 인식 검출기를 포함한다. 이 패턴인식 검출기는 각 패턴을 명령 시퀀스들과 비교하여, 패턴들중 하나가 명령 시퀀스와 실질적으로 일치하는 지를 결정한다. 일 실시예에서는, 패턴이 명령 캐시와 90% 이상 일치할 때에 실질적인 일치가 이루어진다. 다른 실시예에서는, 평션 프리프로세서가 룩업 테이블(look-up table)을 포함하는바, 이 룩업 테이블은 DSP 평션들을 실행하는 명령 시퀀스들을 나타내는 다수의 비트 패턴 엔트리들을 저장한다. 평션 프리프로세서는 각 패턴 엔트리를 명령 시퀀스와 비교하여, 엔트리들중 하나가 명령 시퀀스와 정확히 일치하는 지의 여부를 결정한다. 다른 실시예들은 룩업 테이블 및 패턴 인식 검출기에 의한 2단계의 결정을 포함한다.

바람직한 실시예에서, 평션 프리프로세서는 컨벌루션, 상관 관계, 고속 퓨리에 변환(FFR), 유한 임펄스 응답(FIR) 필터들, 무한 임펄스 응답(IIR) 필터들, 내적 및 행렬 조작 연산들과 같은 DSP타입 평션들을 수행하도록 의도된 X86 명령 시퀀스들을 검출한다.

명령 캐시 또는 명령 메모리 내의 명령들이 DSP 타입 평션을 실행하지 않는 다면, 종래 기술에 따른 현재의 컴퓨터 시스템들에서도 이루어지는 바와 같이, 범용 또는 X86 코어, 또는 1개 이상의 X86 실행 유닛들에게 명령들이 제공된다. 따라서, X86 코어가, DSP 평션들을 나타내지 않는 범용 X86 명령들을 실행한다.

DSP 평션을 수행하는, 즉 DSP 타입 평션들을 수행하도록 의도된 X86 명령들의 시퀀스들을 검출했을 때, 평션 프리프로세서는 X86 명령들의 시퀀스들을 디코드하여, 이 X86 명령들의 시퀀스에 의해 지시되는 평션을 나타내는 단일 매크로 또는 평션 식별자를 생성한다. 평션 프리프로세서는 또한 X86 명령 시퀀스의 정보를 검사하여, DSP 타입 연산에 이용되고 있는 데이터 값들을 나타내는 0개 또는 그 이상의 파라미터들을 생성한다. 이후, 평션 프리프로세서는 평션 식별자 및 필요한 많은 파라미터들을 DSP 코어 또는 1개 이상의 DSP 실행 유닛에 제공한다.

DSP 코어는 매크로 또는 평션 식별자 및 각각의 파라미터들을 수신한 다음, 이 매크로를, 지시된 DSP 평션을 실행하는 DSP 마이크로코드 시퀀스로 인덱스하는 데에 이용한다. DSP 코어는 또한 DSP 평션을 수행하는 데 있어 각각의 파라미터들을 이용한다. DSP 코어는 이러한 DSP 타입 수학 연산들에 대해 최적화되기 때문에, 일반적으로 적은 수의 명령들 및 클럭 사이클들을 이용하여 요구되는 평션을 실행할 수 있다.

DSP 코어는 범용 CPU 코어와 동시에 실행된다. 따라서, 데이터의 독립성이 존재한다는 가정하에서, X86(비 DSP) 연산 코드들은 DSP 펑션들과 동시에, 범용 CPU 코어 또는 X86 코어에 의해 실행될 수도 있다. 범용 코어와 DSP 코어는 서로 결합되어, 동기화를 위해 데이터 및 타이밍 신호들을 교환한다. 일 실시예에서는, 범용 코어와 DSP 코어 사이에 캐시 또는 버퍼가 제공되어, 이러한 2개의 유닛들 사이에 정보를 전송한다.

따라서, 범용 CPU 부분이 종래의 시스템들에서와 같이 X86 명령들을 실행한다.

그러나, DSP 타입 펑션들을 수행하도록 의도된 명령 시퀀스들에 대해, 펑션 프리프로세서는 이러한 시퀀스들을 지능적으로 검출하여, 대응하는 매크로 및 파라미터들을 DSP 코어에 제공한다. 따라서, DSP 코어는 범용 코어로부터 이러한 수학적인 펑션들을 오프로드함으로써, 시스템의 성능을 향상시킨다. DSP 코어는 또한 범용 코어와 동시에 동작하기 때문에, 성능에 있어서 그 이상의 이득을 제공한다.

따라서, 본 발명은 DSP연산을 수행하는 DSP코어를 갖는 범용 CPU로 이루어진다. 이 CPU는 지능적인 DSP 펑션 디코더 또는 프리프로세서를 포함하는바, 이는 명령 시퀀스들을 검사한 다음, DSP 펑션들을 수행하는 시퀀스들을 DSP 매크로 명령으로 변환 또는 맵핑하여 DSP 코어에 의해 실행될 수 있게 한다. DSP 코어는 이러한 DSP 매크로 명령을 이용하여, 요구되는 DSP 펑션을 실행한다. DSP 코어는 보다 적은 수의 명령들 및 클럭 사이클들을 이용하여 DSP 펑션을 실행 또는 수행함으로써, 시스템의 성능을 향상시킨다. 따라서, 본 발명의 CPU는 어떠한 부가적인 X86 연산 코드도 필요로 하지 않으면서, X86 논리 보다 효율적으로 DSP 펑션들을 실행한다. 본 발명의 CPU는 또한 X86 만의 CPU에서 동작하는 코드를 실행함으로써, 기존의 소프트웨어와 백워드 호환성을 제공한다. 또한, 본 발명의 CPU에 대해 기록된 코드는 X86만의 CPU상에서도 적절하게 동작한다.

본 발명은 도면과 함께 설명되는 하기의 바람직한 실시예에 대한 상세한 설명을 통해 보다 명확히 이해될 것이다.

도면의 간단한 설명

도 1은 본 발명에 따른, 범용 CPU코어 및 DSP코어를 갖는 CPU를 포함하는 컴퓨터 시스템의 블록도이다.

도 2는 본 발명에 따른, 범용 CPU코어, DSP코어 및 DSP 펑션 프리프로세서를 포함하는 도 1의 CPU의 블록도이다.

도 3은 본 발명의 동작을 나타낸 흐름도이다.

도 4는 도 1의 CPU의 상세 블록도이다.

도 5는 도 4의 명령 디코드 유닛의 블록도이다.

도 6은 본 발명의 일 실시예에 따른 펑션 프리프로세서의 블록도이다.

도 7은 본 발명의 일 실시예에 따른 패턴 인식 검출기를 포함하는 펑션 프리프로세서의 블록도이다.

도 8은 도 7의 패턴 인식 검출기의 동작을 나타낸 도면이다.

도 9는 본 발명의 일 실시예에 따른 룩업 테이블을 포함하는 펑션 프리프로세서의 블록도이다.

도 10은 도 9의 룩업 테이블의 동작을 나타낸 도면이다.

도 11은 본 발명의 일 실시예에 따른 패턴 인식 검출기 및 룩업 테이블을 포함하는 펑션 프리프로세서의 블록도이다.

실시예

컴퓨터 시스템 블록도

도 1은 본 발명에 따른 중앙 처리 장치(CPU) 또는 마이크로프로세서(102)를 포함하는 컴퓨터 시스템의 블록도이다. 도 1의 컴퓨터 시스템은 단지 예시적인 것으로서, 본 발명의 CPU(102)는 다양한 타입의 컴퓨터 시스템들중 어느 것에라도 포함될 수 있다.

도시된 바와 같이, CPU(102)는 범용 CPU코어(212) 및 DSP코어(214)를 포함한다. 하기 설명되는 바와 같이, 이 범용 CPU 코어(212)는 범용(비 DSP) 연산 코드들을 실행하고, DSP 코어(214)는 DSP 타입 펑션들을 실행한다. 본 발명에 따른 실시예에서, 범용 CPU 코어(212)는 X86 코어이다. 즉, X86 패밀리의 마이크로프로세서와 호환성이다. 하지만, 이 범용 CPU코어(212)는, PowerPC 패밀리, DEC 알파 및 SunSparc 패밀리 프로세서들을 포함하는 다양한 타입의 CPU들중 어느 것이라도 될 수 있다. 하기의 설명에서, 범용 CPU 코어(212)는 편의상 X86 코어라 부른다. 이 범용 CPU 코어(212)는 1개 이상의 범용 실행 유닛들을 포함하고, DSP 코어(214)는 1개 이상의 디지털 신호 처리 실행 유닛들을 포함한다.

도시된 바와 같이, CPU(102)는 CPU 논리 버스(104)를 통해 호스트/PCI/캐시브리지 또는 칩셋(chipset)(106)에 연결된다. 이 칩셋(106)은 바람직하게는 인텔사의 트리톤(Triton) 칩셋과 유사하다. 필요한 경우, 제 2 레벨 또는 L2 캐시 메모리(미도시)가 칩셋 내의 캐시 메모리에 연결될 수 있다. 또한, 어떠한 프로세서들에 대해서는, 외부 캐시가 L1 또는 제 1 레벨

캐시가 될 수 있다. 브리지 또는 칩셋(106)은 메모리 버스(108)를 통해 메인 메모리(110)에 연결된다. 바람직하게는, 이 메인 메모리(110)는 필요에 따라 DRAM(다이내믹 랜덤 액세스 메모리) 또는 EDO(확장 데이터 출력) 메모리 또는 다른 타입의 메모리들이 될 수 있다.

상기 칩셋(106)은 인터럽트 시스템, 실시간 클럭(RTC) 및 타이머들, 직접 메모리 액세스(DMA) 시스템, ROM/플래시 메모리, 통신 포트들, 진단 포트들, 커맨드/상태 레지스터들 및 비휘발성 정적 랜덤 액세스 메모리(NVSRAM)(이들 모두 미도시)를 포함하는 다양한 주변 장치들을 포함한다.

호스트/PCI/캐시 브리지 또는 칩셋(106)은 주변 구성요소 상호 연결(PCI) 버스(120)와 인터페이스된다. 바람직한 실시예에서는, PCI 로컬 버스가 이용된다. 그러나, 주목할 사항으로서, VESA(비디오 전자 표준화 위원회) VL 버스와 같은 다른 로컬 버스들이 이용될 수 있다. PCI 버스(120)에는 다양한 타입의 장치들이 연결될 수 있다. 도 1에 도시된 실시예에서는, 비디오/그래픽 제어기 또는 어댑터(170) 및 네트워크 인터페이스 제어기(140)가 PCI 버스(120)에 연결된다. 비디오 어댑터(170)는 비디오 모니터(172)에 연결되고, 네트워크 인터페이스 제어기(140)는 LAN(142)에 연결된다. 또한, 도시된 바와 같이, SCSI(소형 컴퓨터 시스템 인터페이스) 어댑터(122)가 PCI 버스(120)에 연결된다. 이 SCSI 어댑터(122)는 필요에 따라 CD-ROM 드라이브 및 테이프 드라이브와 같은 다양한 SCSI 장치들에 연결될 수 있다. 당업계에 널리 알려져 있는 바와 같이, 다양한 다른 장치들이 PCI 버스(120)에 연결될 수 있다.

확장 버스 브리지 논리(150)가 또한 PCI 버스(120)에 연결된다. 이 확장 버스 브리지 논리(150)는 확장 버스(152)와 인터페이스된다. 이 확장 버스(152)는 AT 버스라고도 불리는 산업 표준 아키텍쳐(ISA) 버스, 확장된 산업 표준 아키텍쳐(EISA) 버스 또는 마이크로채널 아키텍쳐(MCA) 버스를 포함하는 다양한 타입의 버스들중 어느 것이라도 될 수 있다. 확장 버스(152)에는 확장 버스 메모리(154) 및 모뎀(156)과 같은 다양한 장치들이 결합될 수 있다.

CPU 블록도

도 2는 도 1의 CPU(102) 내의 특정한 구성요소들을 나타낸 고레벨의 블록도이다. 도시된 바와 같이, CPU(102)는 시스템 메모리(110)로부터 명령들 또는 연산 코드들을 수신하는 명령 캐시 또는 명령 메모리(202)를 포함한다. 펑션 프리프로세서(204)가 명령 메모리(202)에 연결되어, 이 명령 메모리(202)의 명령 시퀀스들 또는 연산 코드 시퀀스들을 검사한다. 펑션 프리프로세서(204)는 또한 X86코어(212) 및 DSP 코어(214)에 연결된다. 도시된 바와 같이, 펑션 프리프로세서(204)는 X86 코어(212)에 명령들 또는 연산 코드들을 제공하고, DSP 코어(214)에 정보를 제공한다.

X86 코어(212) 및 DSP 코어(214)는 서로 연결되어 데이터 및 타이밍 신호들을 교환한다. 일 실시예에서, CPU(102)는 X86 코어(212)와 DSP 코어(214) 간의 데이터 전송을 용이하게 하기 위해 이들 사이를 인터페이스하는 1개 이상의 버퍼들(미도시)을 포함한다.

도 3 - 흐름도

도 3은 본 발명의 동작을 나타낸 흐름도이다. 주목할 사항으로서, 도 3의 단계들중 2개 이상의 단계들이 동시에 수행될 수 있고, 본 발명의 동작은 편의상 흐름도의 형태로 도시하였다. 도시된 바와 같이, 단계(302)에서, 명령 메모리(202)는 다수의 X86 명령들을 받아 저장한다. 이러한 다수의 X86 명령들은 DSP 펑션을 수행하는 1개 이상의 명령 시퀀스들을 포함한다. 단계(304)에서, 펑션 프리프로세서(204)는 명령 메모리(202) 내의 연산 코드들, 즉 명령 시퀀스를 분석한다. 그리고, 단계(306)에서, 펑션 프리프로세서(204)는 명령들의 시퀀스가 DSP 타입 펑션을 수행하도록 설계 또는 의도되었는 지의 여부, 즉 명령 시퀀스가 DSP 타입 펑션을 실행하는 지의 여부를 지능적으로 결정한다. 본 개시에서, DSP 타입 펑션은 상관 관계, 컨벌루션, 고속 퓨리에 변환, 유한 임펄스 응답 필터, 무한 임펄스 응답 필터, 내적 및 행렬 조작을 포함하는 수학 펑션들중에서 1개 이상을 포함한다. 펑션 프리프로세서(204)의 동작은 도 4와 관련된 설명에서 보다 상세히 설명한다.

명령 캐시(202)에 저장된 명령들 또는 연산 코드들이 DSP 타입 펑션과 대응하지 않는 다면, 단계(308)에서 명령들은 X86 코어(212)에 제공된다. 따라서, 종래 기술의 X86 호환성 CPU에서와 같이, 이러한 명령들 또는 연산 코드들은 명령 캐시(202)로부터 X86 코어(212)로 직접 제공되어 실행된다. 연산 코드들이 X86 코어(212)로 전달되면, 단계(310)에서 X86 코어(212)가 명령들을 실행한다.

단계(306)에서 펑션 프리프로세서(204)가 DSP타입 펑션에 대응하거나 이를 실행하는 명령들의 시퀀스를 검출하면, 단계(312)에서 펑션 프리프로세서(204)는 명령들의 시퀀스를 분석하여, 실행되는 각각의 DSP 타입 펑션을 결정한다. 단계(312)에서, 펑션 프리프로세서(204)는 명령들의 시퀀스를 각각의 DSP 매크로 식별자(이는 펑션 식별자라고도 한다)로 맵핑한다. 단계(312)에서, 상기 펑션 프리프로세서(204)는 연산 코드들의 시퀀스 내의 정보를 분석하여, 펑션 식별자를 실행하는데에 있어서 DSP 코어 또는 가속기(accelerator)(214)가 이용하기 위한 0개 또는 그 이상의 파라미터들을 생성한다. 도시된 바와 같이, 단계(314)에서, 펑션 프리프로세서(204)는 DSP코어(214)에 펑션 식별자 및 파라미터들을 제공한다.

DSP 코어(214)는 펑션 프리프로세서(204)로부터 펑션 식별자 및 관련된 파라미터들을 받아, 단계(316)에서 각각의 DSP 펑션을 수행한다. 바람직한 실시예에서, DSP 코어(214)는 펑션 식별자를 DSP 마이크로코드 RAM 또는 ROM으로 인덱스하는 데에 이용하여, DSP 명령들 또는 연산 코드들의 시퀀스를 실행한다. DSP 명령들에 의해, DSP는 요구되는 DSP 타입 펑션을 실행한다. DSP 코어(214)는 또한 DSP 펑션을 실행하는 데에 각각의 파라미터들을 이용한다.

상기 설명한 바와 같이, X86 코어(212) 및 DSP 코어(214)는 서로 연결되어 데이터 및 타이밍 신호를 교환한다. 바람직한 실시예에서, X86 코어(212)와 DSP 코어(214)는 실질적으로 병렬로, 즉 동시에 동작한다. 따라서, X86 코어(212)가 연산 코드들의 한 시퀀스를 실행하는 동안, DSP 가속기(214)는 연산 코드들의 다른 시퀀스에 대응하는 1개 이상의 DSP 펑션들을 실행할 수 있다. 따라서, DSP코어(214)는 슬레이브 또는 코프로세서로서 동작하는 것이 아니라, 독립적인 실행 유닛 또는 파이프라인으로서 동작한다. DSP 코어(214) 및 X86 코어(212)는 서로 데이터 및 타이밍 신호들을 교환함으로써, 동작 상태를 표시하고, 생성되는 어떠한 데이터 출력들을 제공할 뿐 아니라, 데이터 코히런시/독립성을 보장한다.

동작예

이하, 본 발명에 따라 어떻게 X86 연산 코드들의 스트링 또는 시퀀스가 펑션 식별자로 변환된 다음 DSP 코어 또는 가속기(214)에 의해 실행되는 지의 예를 설명한다. 하기의 설명에서, X86 연산 코드의 시퀀스는 단순한 내적 계산을 수행하는 바, 여기서 상기 내적은 20개의 값들을 포함하는 벡터에 대해 평균화된다.

X86 Code
(Simple inner product)

```
1   Mov ECX num_samples;          {Set up parameters for macro}
1   Mov ESI address_1;
1   Mov EDI, address_2;
1   Mov EAX, 0;                   {Initialize vector indices}
1   Mov EBX, 0;
4   FLdZ;                         {Initialize sum of products}
    Again:
                                      {Update counter}
4   Fld dword ptr [ESI+EAX*4];        {Get vector elements and}
1   Inc EAX;                          {update indices}
4   Fld dword ptr [EDI+EBX*4];
1   Inc EBX;
13  FMulP St(1), St;              {Compute product term}
7   FAddP St(1), St;              {Add term to sum}
1   LOOP Again;                   {Continue if more terms}
```

도시된 바와 같이, 단순 내적을 위한 X86 연산 코드 명령들은 다수의 이동(move) 명령들 및 이에 계속되는 F-load 펑션을 포함하며, 이러한 시퀀스는 다수회 반복된다. 이러한 X86 연산 코드의 시퀀스가 X86 코어(212)에 의해 실행된다면, 이러한 내적 계산을 위한 실행 시간은 709 사이클(9+ 20x35)을 필요로 한다. 이는 i486 타이밍, 부동 소수점 연산들의 동시 실행, 및 내적 계산에 필요한 모든 명령들 및 데이터의 캐시 히트(cache hit)를 가정한다. 펑션 프리프로세서(204)는 연산 코드들의 시퀀스를 분석하여, 이 연산 코드들이 내적 계산을 수행하고 있는 지를 검출한다. 이후, 펑션 프리프로세서(204)는 X86 연산 코드들의 시퀀스 전체를 단일 매크로 또는 펑션 식별자 및 1개 이상의 파라미터들로 변환한다. 상기에서 보인 X86 연산 코드 시퀀스에 기초하여 생성되는 매크로 또는 펑션 식별자의 예를 들면 다음과 같다:

Example Macro
(as it appears in assembler)

```
Inner_product_simple (
        address_1,          {Data vector}
        address_2,          {Data vector}
        num_samples);{Length of vector}
```

이러한 펑션 식별자 및 1개 이상의 파라미터들은 DSP코어(214)에 제공된다. 이 DSP 코어(214)는 펑션 프리프로세서(204)로부터 제공되는 매크로를 이용하여, DSP 펑션을 실행하는 1개 이상의 DSP 연산 코드들 또는 명령들을 적재한다. 바람직한 실시예에서, DSP 코어는 상기 매크로를, DSP 펑션을 실행하는 데에 이용되는 명령들을 포함하는 ROM으로 인덱스하는 데에 이용한다. 이 예에서, 상기 설명한 매크로의 수신에 응답하여 DSP 코어(214)에 의해 실행되는 DSP 코어 또는 명령들은 다음과 같다:

DSP Code
(Simple inner product)

```
1   Cntr  =     num_samples;  {Set up parameters from macro}
1   ptr1  =     address_1
1   ptr2  =     address_2
1   MAC   =     0;            {Initialize sum of products}
1   reg1  =     *ptr1++,      {Pre-load multiplier input registers}
    reg2  =    *ptr2++;
1   Do LOOP until ce;         {Specify loop parameters}
1   MAC  += reg1  *reg2,      {Form sum of products}
         reg1 =   *ptr1++,
         reg2 =   *ptr2++;
    LOOP:                     {Continue if more terms}
```

이 예에서, DSP 코어(214)는 20개의 값들을 포함하는 벡터에 대해 평균화되는 이러한 내적을 수행하여, 총 26 사이클(6+ 20×1)을 소모한다. 이는, 명령들의 단일 사이클 연산, 제로 오버헤드 루핑(looping), 및 모든 명령들 및 데이터에 대한 캐시 히트를 포함하는 전형적인 DSP 타이밍을 가정한다. 따라서, DSP코어(214)는 X86 코어(212)가 이러한 DSP 펑션을 실행하는 경우에 비해 28배가 넘는 성능의 향상을 가져온다.

도 4 - CPU 블록도

도 4는 본 발명에 따른 CPU(102)의 내부 구성요소들을 상세히 도시한 블록도이다. 설명을 단순화하기 위해, 본 발명을 이해하는 데에 필요없는 CPU(102)의 구성요소들에 대해서는 설명하지 않는다. 도시된 바와 같이, 바람직한 실시예에서, CPU(102)는 버스 인터페이스 유닛(440), 명령 캐시(202), 데이터 캐시(444), 명령 디코드 유닛(402), 다수의 실행 유닛들(448), 적재/저장 유닛(450), 재배열 버퍼(452), 레지스터 파일(454), DSP실행 유닛(214)을 포함한다.

도시된 바와 같이, CPU(102)는 버스 인터페이스 유닛(440)을 포함하는바, 이는 CPU 버스(104)를 통해 통신을 수행하는 회로를 포함한다. 이 버스 인터페이스 유닛(440)은 데이터 캐시(444) 및 명령 캐시(202)와 인터페이스 한다. 명령 캐시(202)는 시스템 메모리(110)로부터 명령들을 프리페치(prefetch)한 다음, CPU(102)가 이용할 수 있도록 이 명령들을 저장한다. 명령 디코드 유닛(402)이 명령 캐시(202)에 연결되어, 이 명령 캐시(202)로부터 명령들을 받는다. 명령 디코드 유닛(402)은 도시된 바와 같이 펑션 프리프로세서(204)를 포함한다. 명령 디코드 유닛(202) 내의 펑션 프리프로세서(204)는 명령 캐시(202)에 연결된다. 명령 디코드 유닛(402)은 또한 명령 정렬 유닛 및 다른 논리를 포함한다.

명령 디코드 유닛(402)은 다수의 실행 유닛들(448), 재배열 버퍼(452) 및 적재/저장 유닛(450)에 연결된다. 다수의 실행 유닛들은 본원에서 집합적으로 실행 유닛들(448)로 불려진다. 재배열 버퍼(452), 실행 유닛들(448) 및 적재/저장 유닛(450)은 각각 실행 결과들을 전송하기 위해 전송 버스(458)에 연결된다. 적재/저장 유닛(450)은 데이터 캐시(444)에 연결된다. DSP 실행 유닛(214)은 DSP 디스패치 버스(456)를 통해 명령 디코드 유닛(402)에 직접 연결된다. 주목할 사항으로서, 1개 이상의 DSP실행 유닛들(214)이 명령 디코드 유닛(402)에 연결될 수 있다.

버스 인터페이스 유닛(440)은 마이크로프로세서(102)와 시스템 버스(104)에 연결된 장치들 간에 통신을 수행하도록 구성된다. 명령 캐시(202)에서 미스(miss)된 명령 페치들은 버스 인터페이스 유닛(440)에 의해 메인 메모리(110)로부터 전송된다. 유사하게, 적재/저장 유닛(450)에 의해 수행되는 데이터 요구들(이들은 데이터 캐시(444)에 없다)은 버스 인터페이스 유닛(440)에 의해 메인 메모리(110)로부터 전송된다. 또한, 데이터 캐시(444)는 마이크로프로세서(102)에 의해 변경된 데이터의 캐시 라인을 폐기한다. 버스 인터페이스 유닛(440)은 변경된 라인을 메인 메모리(110)로 전송한다.

명령 캐시(202)는 바람직하게는 명령들을 저장하는 고속의 캐시 메모리이다. 주목할 사항으로서, 이 명령 캐시(202)는 세트 어소시에이티브 구성(set-associative configuration) 또는 직접 맵핑 구성(direct mapped configuration)으로 이루어질 수 있다. 이 명령 캐시(202)는 또한 분기 명령들을 테이큰(taken) 또는 낫테이큰(not taken)으로서 예측하는 분기 예측 메커니즘을 포함한다. "테이큰" 분기 명령에 의해, 명령의 페치 및 실행은 분기 명령의 타겟 어드레스에서 계속된다. "낫테이큰" 분기 명령에 의해, 명령의 페치 및 실행은 이 분기 명령 다음의 명령에서 계속된다. 명령들은 명령 캐시(202)로부터 페치되어, 명령 디코드 유닛(402)으로 전달되어 디코드된 다음, 실행 유닛으로 디스패치된다. 명령 캐시(202)는 또한 매크로 명령들을 예측하여 적절한 동작을 취하는 매크로 예측 메커니즘을 포함한다.

명령 디코드 유닛(402)은 명령 캐시(202)로부터 수신한 명령들을 디코드한 다음, 이 디코드된 명령들을 실행 유닛들(448), 적재/저장 유닛(450) 또는 DSP 실행 유닛(214)에 제공한다. 명령 디코드 유닛(402)은 바람직하게는 명령을 1개 이상의 실행 유닛(448)에 디스패치하도록 구성된다.

명령 디코드 유닛(402)은 펑션 프리프로세서(204)를 포함한다. 본 발명에 따르면, 명령 디코드 유닛(402)의 펑션 프리프로세서(204)는 DSP 펑션들에 대응하거나 또는 이를 수행하는 명령 캐시(202) 내의 X86 명령 시퀀스들을 검출하도록 구성된다. 이러한 명령 시퀀가 검출되면, 펑션 프리프로세서(204)는 대응하는 매크로 및 파라미터들을 생성한 다음, 이 대응하는 DSP 매크로 및 파라미터들을 DSP 디스패치 버스(456)를 통해 DSP 실행 유닛(214)에 전송한다. DSP 실행 유닛(214)은 명령 디코드 유닛(402)으로부터 DSP 펑션 매크로 및 파라미터 정보를 수신한 다음, 지시된 DSP 펑션을 수행한다. 또한, DSP 실행 유닛(214)은 바람직하게는 데이터 오퍼랜드들을 위해 데이터 캐시(444)를 액세스하도록 구성된다. 이러한 데이터 오퍼랜드들은 보다 빠른 액세스를 위해 DSP 실행 유닛(214) 내의 메모리에 저장되거나, 또는 필요할 때 데이터 캐시(444)로부터 직접 액세스될 수 있다. 펑션 프리프로세서(204)는 명령 캐시(202)에 피드백을 제공함으로써, 매크로 검색을 위한 충분한 룩어헤드 명령들을 얻을 수 있게 한다.

명령 캐시(202)의 X86 명령들이 DSP 펑션을 수행하도록 의도되지 않는 다면, 명령 디코드 유닛(402)은 명령 캐시(202)로부터 페치된 명령들을 디코드한 다음, 이 명령들을 실행 유닛들(448) 그리고/또는 적재/저장 유닛(450)으로 디스패치한다. 명령 디코드 유닛(402)은 또한 명령에 의해 이용되는 레지스터 오퍼랜드들을 검출하고, 재배열 버퍼(452) 및 레지스터 파일(454)로부터 이러한 오퍼랜드들을 요구한다. 실행 유닛들(448)은 X86 명령들을 실행하는바, 이는 당업계에 알려져있다.

또한, DSP실행 유닛(214)이 CPU(102)에 포함되지 않거나 또는 소프트웨어를 통해 디스에이블된다면, 명령 디코드 유닛(402)은 모든 X86 명령들을 실행 유닛들(448)로 디스패치한다. 이 실행 유닛들(448)은 종래 기술에서와 같이 X86 명령들을 실행한다. 이러한 방식으로, DSP 실행 유닛(214)이 디스에이블되면, DSP 펑션들을 수행하는 명령들을 포함하는 X86 코드가, 현재 기술의 X86 마이크로프로세서로 행해지는 바와 같이, X86 코어에 의해 실행된다. 따라서, DSP 실행 유닛(214)이 디스에이블되면, 프로그램은 DSP 실행 유닛(214)의 대응하는 루틴의 실행에 비해 동작의 효율은 떨어지지만 정확하게 실행한다. 유익하게는, CPU(102)의 DSP 코어(214)의 인에이블 또는 디스에이블, 또는 존재 또는 부재는 프로그램의 정확한 동작에 영향을 미치지 않는다.

일 실시예에서, 실행 유닛들(448)은, 각각 마이크로프로세서(102)에 의해 이용되는 명령 세트를 실행하도록 구성된 대칭적인 실행 유닛들이다. 다른 실시예에서, 실행 유닛들(448)은 다른 명령 서브 세트들을 실행하도록 구성된 비대칭적인 실행 유닛들이다. 예를 들어, 실행 유닛들(448)은 분기 명령들을 실행하는 분기 실행 유닛, 산술 및 논리 명령들을 실행하는 1개 이상의 산술/논리 유닛들 및 부동 소수점 명령들을 실행하는 1개 이상의 부동 소수점 유닛들을 포함할 수 있다. 명령 디코드 유닛(402)은 명령을 실행 유닛(448) 또는 적재/저장 유닛(450)으로 디스패치하는바, 이들은 상기 명령을 실행하도록 구성된다.

적재/저장 유닛(450)은 실행 유닛(448)과 데이터 캐시(444) 간에 인터페이스를 제공한다. 적재 및 저장 메모리 동작들은 적재/저장 유닛(450)에 의해 데이터 캐시(444)에 대해 수행된다. 또한, 적재 및 저장 메모리 동작들 간의 메모리 의존성이 적재/저장 유닛(450)에 의해 검출되어 처리된다.

실행 유닛들(448) 및 적재/저장 유닛(들)(450)은, 오퍼랜드들이 아직 제공되지 않은 명령들을 저장하는 1개 이상의 예약 스테이션(reservation station)들을 포함한다. (1) 명령의 오퍼랜드들이 제공되고, (2) 선택된 명령 이전의 명령들이 아직 오퍼랜드들을 수신하지 않았다면, 예약 스테이션들에 저장된 명령들로부터 명령이 선택되어 실행된다. 주목할 사항으로

서, 개별적인 예약 스테이션들 대신 집중(centralized) 예약 스테이션이 포함될 수 있다. 이 집중 예약 스테이션은 명령 디코드 유닛(402), 실행 유닛들(448) 및 적재/저장 유닛(450) 사이에 연결된다. 이러한 실시예는 집중 예약 스테이션 내에서 디스패치 평션을 수행할 수 있다.

CPU(102)는 바람직하게는 순서를 벗어난(out of order) 실행을 지원하며, 그리고 재배열 버퍼(452)를 이용하여, 추론적으로 실행되는 명령들의 실행 결과들을 저장하고, 이러한 결과들을 프로그램 순서로 레지스터 파일(454)에 저장함으로써, 의존성 체크 및 레지스터 재명명(renaming)을 수행하며, 오예측된 분기 및 예외 복구를 제공한다. 명령 디코드 유닛(402)이 명령을 디코드하면, 레지스터 오퍼랜드들이 재배열 버퍼(452) 및 레지스터 파일(454)로 전달된다. 레지스터 오퍼랜드 요구들에 응답하여, 다음의 3개의 값들, 즉 (1) 값이 추론적으로 생성되는 경우에는, 재배열 버퍼(452)에 저장된 값, (2) 값이 추론적으로 생성되지 않는 경우에는, 그 결과를 저장하게 될 재배열 버퍼(452) 내의 위치를 식별하는 태그(tag), 및 (3) 재배열 버퍼(452) 내의 어떠한 명령들도 레지스터를 수정하지 않는 경우에는, 레지스터 파일(454) 내의 레지스터에 저장된 값중 하나가, 명령을 수신한 실행 유닛(448) 그리고/또는 적재/저장 유닛(450)으로 전송된다. 또한, 명령 디코드 유닛(402)에 의해 디코드되는 명령의 결과들을 저장하기 위해, 재배열 버퍼(452) 내에 저장 위치가 할당된다. 이 저장 위치는 태그에 의해 식별되는바, 이 태그는 명령을 수신하는 유닛으로 전달된다. 주목할 사항으로서, 1개 이상의 재배열 버퍼의 저장 위치가 특정한 레지스터에 대응하는 값들을 저장하도록 할당되면, 프로그램 순서의 마지막 값에 대응하는 값 또는 태그가 그 특정한 레지스터에 대한 레지스터 오퍼랜드 요구에 응답하여 전달된다.

실행 유닛들(448) 또는 적재/저장 유닛(450)이 명령을 실행할 때, 재배열 버퍼(452)에 의해 명령에 할당된 태그가 명령의 결과와 함께 결과 버스(458) 상으로 전달된다. 재배열 버퍼(452)는 지시된 저장 위치에 이 결과를 저장한다. 또한, 실행 유닛들(448) 및 적재/저장 유닛(450)은 그 내에 저장된 명령들에 대한 오퍼랜드들의 태그들을 결과 버스(458) 상으로 전달된 태그들과 비교한다. 일치하면, 유닛은 결과 버스(458)로부터 결과를 캡쳐한 다음 이를 대응하는 명령과 함께 저장한다. 이러한 방식으로, 명령은 동작하도록 의도되는 오퍼랜드들을 받을 수 있다. 명령들이 이용할 수 있도록 결과 버스(458)로부터 결과들을 캡쳐하는 것을 "결과 포워딩(result forwarding)"이라 한다.

명령 결과들은 재배열 버퍼(452)에 의해 프로그램 순서로 레지스터 파일(454)에 저장된다. 명령의 결과들을 저장한 다음 재배열 버퍼(452)로부터 이 명령을 삭제하는 것을 명령의 "퇴거(retiring)"라고 한다. 명령들을 프로그램 순서로 퇴거함으로써, 부정확한 추론적 실행에 대한 복구가 수행될 수 있다. 예를 들어, 어떠한 명령이, 테이큰/낫테이큰 예측이 부정확한 분기 명령 뒤에 온다면, 이 명령은 부정확하게 실행될 수 있다. 오예측된 분기 명령 또는 예외를 일으키는 명령이 검출되면, 재배열 버퍼(452)는 이러한 오예측된 분기 명령들 이후의 명령들을 폐기한다. 이렇게 폐기되는 명령들은 또한 실행 유닛들(448), 적재/저장 유닛(450) 및 명령 디코드 유닛(402)으로부터도 소거(flush)된다.

레지스터 파일(454)은, 마이크로프로세서(102)가 이용하는 마이크로프로세서 아키텍쳐에 의해 정의되는 각 레지스터를 위한 저장 위치들을 포함한다. 예를 들어, CPU(102)가 X86 마이크로프로세서 아키택쳐를 포함하는 바람직한 실시예에서, 레지스터 파일(454)은 EAX, EBX, ECX, EDX, ESI, EDI, ESP 및 EBP 레지스터 값들을 저장하는 위치들을 포함한다.

데이터 캐시(444)는 마이크로프로세서(102)에 의해 조작되는 데이터를 저장하도록 구성된 고속의 캐시 메모리이다. 주목할 사항으로서, 이 데이터 캐시(444)는 세트 어소시에이티브 또는 집적 맵핑 구성으로 이루어질 수 있다.

도 5 - 명령 디코드 유닛

도 5는 명령 디코드 유닛(402)의 일 실시예를 도시한다. 이 명령 디코드 유닛(402)은 명령 정렬 유닛(460), 다수의 디코더 회로들(462) 및 DSP 펑션 프리프로세서(204)를 포함한다. 명령 정렬 유닛(460)은 명령 캐시(202)로부터 페치된 명령들을 수신한 다음, 이 명령들을 디코더 회로들(462)로 정렬시키도록 결합된다.

명령 정렬 유닛(460)은 명령들을 디코더 회로들(462)로 루팅(routing)한다. 일 실시예에서, 명령 정렬 유닛(460)은 명령 캐시(202)로부터 페치된 명령 바이트들이 큐(queue)되는 바이트 큐를 포함한다. 명령 정렬 유닛(460)은 바이트 큐에서 유효한 명령들을 찾은 다음, 이 명령들을 각각의 디코더 회로들(462)로 디스패치한다. 다른 실시예에서, 명령 캐시(202)는 프리디코드 회로(predecode circuit)를 포함하는바, 이는 명령 바이트들이 명령 캐시(202)에 저장될 때 이 명령 바이트들을 프리디코드한다. 명령들의 시작 및 끝을 나타내는 시작 및 끝 바이트 정보가 생성되어 명령 캐시(202)에 저장된다. 프리디코드 데이터가 명령들과 함께 명령 정렬 유닛(460)으로 전송되면, 이 명령 정렬 유닛(460)은 프리디코드 정보에 따라 상기 명령들을 디코더 회로들(462)로 전송한다.

펑션 프리프로세서(204)가 또한 명령 캐시(202)에 연결되어, 명령 캐시(202)내의, DSP 명령들을 수행하는 명령 시퀀스들을 검출한다. 디코더 회로(462) 및 펑션 프리프로세서(204)는 명령 정렬 유닛(460)으로부터 X86 명령들을 수신한다. 펑션 프리프로세서(204)는 DSP 버스를 통해 명령 디스에이블 신호를 디코더 회로들(462) 각각에 제공한다.

각 디코더 회로(462)는 명령 정렬 유닛(460)으로부터 수신된 명령을 디코드하여, 그 명령에 의해 조작되는 레지스터 오퍼랜드들 및 그 명령을 수신하는 유닛을 결정한다. 명령을 수신하는 유닛의 표시 및 명령 그 자체가 다수의 디스페치 버스들(468)을 통해 실행 유닛들(448) 및 적재/저장 유닛(450)으로 전달된다. 미도시된 다른 버스들이 재배열 버퍼(452) 및 레지스터 파일(454)로부터 레지스터 오퍼랜드들을 요구하는 데에 이용된다.

펑션 프리프로세서(204)는 명령 캐시(202)로부터의 X86 명령들의 스트림들 또는 시퀀스들을 분석하여, DSP 펑션이 실행되고 있는 지의 여부를 결정한다. DSP 펑션이 수행되고 있으면, 펑션 프리프로세서(204)는 X86 명령 스트림을 DSP 매크로 및 0개 또는 그 이상의 파라미터들로 맵핑한 다음, 이러한 정보를 1개 이상의 DSP 실행 유닛들(214)에 제공한다. 일 실시예에서, 각 명령 시퀀스가 디코더 회로(462)에 도달하면, 펑션 프리프로세서(204)는 디코더 회로들(462) 각각에게 디스에이블 신호를 표명(assert)하여, 검출된 명령 시퀀스에 대한 디코더 회로들(462)의 동작을 디스에이블시킨다. 디코더 회로(462)가 펑션 프리프로세서(204)로부터 디스에이블 신호를 검출하면, 이 디코더 회로(462)는 디스에이블 신호가 해제될 때 까지 디코딩 동작을 중단한다. DSP 펑션에 대응하는 명령 시퀀스가 명령 캐시(202)를 나온 후, 펑션 프리프로세서

(204)는 각 디코더 회로(462)에 대한 디스에이블 신호를 제거한다. 즉, 펑션 프리프로세서(204)가 X86 명령 시퀀스의 끝을 검출하면, 이 펑션 프리프로세서(204)는 각 디코더 회로(462)에 대한 디스에이블 신호를 제거하며, 이에 따라 디코더는 동작을 재개한다.

디코더 회로들(462) 각각은 명령, 및 이 명령을 수신하는 유닛 또는 유닛들의 표시를 디스패치 버스들(468)중 하나를 통해 전달하도록 구성된다. 일 실시예에서, 상기 표시 내에는, 실행 유닛들(448) 및 적재/저장 유닛(450) 각각에 대한 비트가 포함된다. 특정 비트가 세트되면, 대응하는 유닛이 그 명령을 실행한다. 특정 명령이 1개 이상의 유닛에 의해 실행된다면, 상기 표시 내에는 1개 이상의 비트가 세트될 수 있다.

펑션 프리프로세서

도 6은 본 발명의 일 실시예에 따른 펑션 프리프로세서(204)의 블록도이다. 도시된 바와 같이, 이 실시예에서, 펑션 프리프로세서(204)는 명령 메모리 또는 명령 캐시(202)의 명령들의 시퀀스들을 검사 또는 스캔하는 스캔 어헤드 회로(scan-ahead circuit)(502)를 포함한다. 일 실시예에서, 스캔 어헤드 회로 또는 수단(502)은 스캔되는 각 명령 시퀀스를 포함하는 명령들을 디코딩할 때, 명령 디코더(402)가 동작하기 이전에, 명령 메모리(202)에 저장된 명령들의 시퀀스들을 검사한다. 따라서, 이 스켄 어헤드 회로(502)는, 각 명령이 명령 디코더(402)에 제공되기 전에, 명령 캐시(202)의 명령 시퀀스들을 앞서서 검사한다.

펑션 프리프로세서(204)는 또한 명령 메모리(202)의 명령들의 시퀀스가 디지털 신호 처리 펑션을 구현하는 지의 여부를 결정하는 명령 시퀀스 결정 회로(504)를 포함한다. 이러한 결정은 하기 설명되는 다양한 방식으로 수행될 수 있다.

펑션 프리프로세서(204)는 또한 디지털 신호 처리 펑션을 구현하는 명령 메모리(202)의 명령들의 시퀀스를 디지털 신호 처리 펑션 식별자 또는 매크로 식별자 및 0개 또는 그 이상의 파라미터들로 변환하는 변환/맵핑 회로(506)를 포함한다. 이에 따라, 명령 시퀀스 결정 회로(504)가 명령 메모리(202)의 명령들의 시퀀스가 FFT 펑션을 구현한다고 결정하면, 변환/맵핑 회로(506)는 이러한 명령들의 시퀀스를 FFT 펑션 식별자 및 0개 또는 그 이상의 파라미터들로 변환한다.

도 7 - 패턴 인식 회로

도 7을 참조하여, 본 발명의 일 실시예에서, 펑션 프리프로세서(204)는 패턴 인식 회로 또는 패턴 인식 검출기(512)를 포함하는바, 이는 명령 메모리(202)의 명령들의 시퀀스가 디지털 신호 처리 펑션을 구현하는 지의 여부를 결정한다. 이 패턴 인식 검출기(512)는 디지털 신호 처리 펑션들을 구현하는 명령 시퀀스들의 디수의 패턴들을 저장한다. 이 패턴 인식 검출기(512)는 FFT, 내적, 행렬 조작, 상관 관계 및 컨벌루션 등과 같은 DSP 펑션을 수행하는 기계 언어 명령들의 연산 코드 시퀀스들에 대응하는 비트 패턴들을 저장한다.

패턴 인식 검출기(512)는 명령 메모리(202)에 저장된 명령들의 시퀀스를 검사한 다음, 이러한 명령들의 시퀀스를 다수의 저장된 패턴들과 비교한다. 패턴 인식 검출기(512)의 동작은 도 8에 도시된다. 이 패턴 인식 검출기(512)는 패턴들 각각을 명령 시퀀스의 주기적인 위치들에 있는 명령 시퀀스와 비교한다. 대안적으로, 패턴 인식 검출기(512)는 패턴들 각각을 명령 시퀀스의 미리 규정된 위치들에 있는 명령 시퀀스와 비교한다. 패턴 인식 검출기(512)는 필요한 경우, 패턴 비교를 수행하는 유닛으로서 룩업 테이블을 포함할 수 있다. 이 패턴 인식 검출기(512)는 또한 명령 시퀀스들에 대해 매크로 예측을 수행하여, 성능을 개선한다.

패턴 인식 검출기(512)는 명령 메모리(202)의 명령들의 시퀀스가 다수의 저장된 패턴들중 하나와 실질적으로 일치하는지의 여부를 결정한다. 실질적인 일치는 명령들의 시퀀스가 디지털 신호 처리 펑션을 구현한다는 것을 나타낸다. 바람직한 실시예에서, 실질적인 일치는 명령 시퀀스가 저장된 패턴과 90% 이상 일치하는 경우 이루어진다. 필요에 따라, 95% 또는 100%와 같은 다른 일치 임계값들이 이용될 수 있다. 일치가 이루어지면, 패턴 인식 검출기(512)는 명령들의 시퀀스와 일치하는 DSP 펑션 패턴의 타입을 결정한 다음, 이 DSP 펑션 패턴의 타입을 변환/맵핑 회로(506)로 전달한다.

도 9 -룩업 테이블

도 9를 참조하여, 다른 실시예에서, 펑션 프리프로세서(204)는 룩업 테이블(514)을 포함하는바, 이 룩업 테이블(514)은 명령 메모리(202)의 명령들의 시퀀스가 디지털 신호 처리 펑션을 구현하는 지의 여부를 결정한다. 일 실시예에서, 룩업 테이블(514)은 패턴 인식 검출기(512)에 부가적으로 구성되거나, 또는 이 패턴 인식 검출기(512)를 대신하여 구성될 수 있다.

펑션 프리프로세서(204)가 룩업 테이블(514) 만을 포함하는 실시예에서, 이 룩업 테이블(514)은 다수의 패턴들을 저장하는바, 이러한 패턴들 각각은 디지털 신호 처리 펑션을 구현하는 명령 시퀀스의 최소의 서브 세트이다. 따라서, 이 실시예는, 펑션 프리프로세서(204)가 DSP 펑션들을 구현하는 명령 시퀀스들을 검출하는 패턴 인식 검출기(512) 대신 룩업 테이블(514)을 포함한다는 것을 제외하고는, 상기 설명한 도 6의 실시예와 유사하다. 또한, 이 실시예에서, 룩업 테이블(514)은 DSP 펑션을 수행하는 명령들의 보다 작은 시퀀스들, 즉 명령 시퀀스들의 서브 세트들에 대응하는 보다 작은 패턴들을 저장한다. 이 실시예에서, 룩업 테이블(514)은 대응하는 명령들의 시퀀스와의 정확한 일치를 요구한다. 정확한 일치가 이루어지지 않으면, 명령들의 시퀀스는 1개 이상의 범용 실행 유닛들, 즉 범용 CPU 코어로 전달되어 실행된다.

도 10은 이 실시예의 룩업 테이블(514)의 동작을 나타낸다. 도시된 바와 같이, 명령 캐시(202)의 명령들의 시퀀스는 명령 래치(542)에 일시적으로 저장된다. 이 명령 래치(542)의 내용은 이후 요소(546)에 의해 룩업 테이블(514)의 엔트리들 각각과 비교된다. 명령 래치(542)의 내용이 룩업 테이블(514)의 엔트리들중 하나와 정확히 일치하면, 이 엔트리에 대응하는 DSP 펑션 또는 명령(548)이 DSP 실행 유닛(214)에 제공된다.

상기 도 7 및 도 9의 실시예들에서, 패턴 인식 검출기(지2) 그리고/또는 룩업 테이블(514)은, 결정이 비교적 확실하게 이루어질 때에만, 명령 시퀀스가 DSP 펑션을 구현하는 것으로 결정하도록 구성된다. 이는 "미스된" 명령 시퀀스, 즉 DSP 펑

션을 구현하지만 이 DSP 펑션을 구현하는 것으로서 검출되지 않았던 명령 시퀀스는 범용 코어 또는 실행 유닛에 의해 실행될 수 있어 CPU(102)의 동작에 영향을 끼치지 않기 때문이다. DSP 펑션을 구현하지 않지만 DSP 기능을 구현하는 시퀀스로서 잘못 식별되는 명령 시퀀스가 더욱 문제가 되는바, 이는 오동작을 야기시킬 수 있다. 따라서, 패턴 인식 검출기(512) 또는 룩업 테이블(514)이 DSP 펑션을 구현하는 모든 명령 시퀀스를 정확하게 검출하지 못할 것으로 예상된다. 이 경우, 종래 기술에서와 같이, 명령 시퀀스는 범용 실행 유닛들중 하나로 전달된다.

도 11 -룩업 테이블 및 패턴 인식 회로를 갖는 펑션 프리프로세서

도 11을 참조하여, 본 발명의 다른 실시예에서, 펑션 프리프로세서(204)는 룩업 테이블(514) 및 패턴 인식 검출기(512)를 포함한다. 이 실시예에서, 펑션 프리프로세서(204)는 룩업 테이블(514) 및 패턴 인식 검출기(512)를 각각 이용하여, 명령 메모리(202)의 명령들의 시퀀스가 디지털 신호 처리 펑션을 구현하는 지의 여부를 결정한다. 이 실시예는 바람직하게는 X86 명령들의 시퀀스에 대해 2 단계의 분석을 이용하는바, 먼저 룩업 테이블(514)이 그 시퀀스가 DSP 펑션을 구현할 가능성이 있는 지의 여부를 결정하면, 이후 패턴 인식 검출기(512)가 구현되는 DSP 펑션의 타입을 결정한다. 대안적으로, 먼저 패턴 인식 검출기(512)가 그 시퀀스가 DSP 펑션을 구현할 가능성이 있는 지의 여부를 결정하면, 이후 룩업 테이블(514)이 구현되는 DSP 펑션의 타입을 결정한다.

이 실시예에서, 룩업 테이블(514)은 극소(atomic) DSP 명령들에 대응하는 작은 패턴들을 저장한다. 예를 들어, 룩업 테이블(514)은 DSP 아키텍쳐들에서 공통적인 곱셈 누적 가산 펑션을 수행하는 X86 명령들의 패턴을 저장한다. 룩업 테이블(514)은 또한 극소 DSP 명령들을 구현하는 다른 패턴들을 저장한다. 패턴 인식 검출기(512)는 FFT, 상관 관계 및 컨벌루션 등과 같은 전체 DSP 펑션들에 대응하는 패턴들을 저장한다.

먼저, 룩업 테이블(514)은 각 엔트리와 유입되는 명령 시퀀스들을 비교하여, 어느 시퀀스에 대한 "히트(hit)" 또는 일치의 수를 저장한다. 일치의 수가 미리 규정된 임계치 보다 크면, 이 시퀀스는 다수의 DSP 타입 "명령들"을 포함하며, 이에 따라 DSP 펑션을 구현하는 것으로 추정된다. 이 경우, 패턴 인식 검출기(512)가 인에이블되어, 전체 시퀀스와 저장된 패턴들 각각을 비교함으로써, X86 명령 시퀀스에 의해 구현되는 DSP 펑션의 타입을 결정한다. 상기 설명한 바와 같이, 패턴 인식 검출기(512)는 명령 시퀀스가 저장된 패턴들중 하나와 실질적으로 일치하는 지의 여부를 결정한다.

결론

따라서, 본 발명은 종래의 소프트웨어와 백워드 호환성을 유지하면서, DSP 그리고/또는 수학 연산들의 실행을 최적화하는 새로운 CPU 또는 마이크로프로세서 아키텍쳐를 개시한다.

본 발명의 시스템 및 방법이 바람직한 실시예와 관련하여 설명되었지만, 본 발명은 본원에 개시된 특정한 형태로 제한되지 않는다. 본 발명은 첨부된 청구항들에 의해 정의되는 본 발명의 정신 및 범위 내에 포함되는 모든 대안들, 수정들 및 등가들을 포괄하는 것으로 의도된다.

(57) 청구의 범위

청구항 1.

디지털 신호 처리 펑션을 수행하는 중앙 처리 장치에 있어서,

복수의 명령들을 저장하는 명령 메모리와, 여기서 상기 명령 메모리는 상기 디지털 신호 처리 펑션을 수행하도록 의도된 1개 이상의 명령들의 시퀀스들을 저장하며;

상기 명령 메모리에 결합된 펑션 프리프로세서와, 여기서 상기 펑션 프리프로세서는 상기 명령 메모리에 저장된 상기 1개 이상의 명령들의 시퀀스들을 검사하는 수단, 상기 명령 메모리의 상기 명령들의 시퀀스가 상기 디지털 신호 처리 펑션을 수행하도록 의도되었는 지의 여부를 결정하는 수단, 및 상기 디지털 신호 처리 펑션을 수행하도록 의도된 상기 명령 메모리의 상기 명령들의 시퀀스를 디지털 신호 처리 펑션 식별자로 변환하는 수단을 포함하며;

상기 펑션 프리프로세서에 연결되어 상기 명령 메모리의 명령들을 실행하는 적어도 하나의 범용 처리 코어와; 그리고

상기 펑션 프리프로세서에 연결되어 상기 디지털 신호 처리 펑션을 수행하는 적어도 1개의 디지털 신호 처리 코어를 포함하여 구성되며,

여기서 상기 적어도 1개의 디지털 신호 처리 코어는 상기 펑션 프리프로세서로부터 상기 디지털 신호 처리 펑션 식별자를 수신한 다음, 이 디지털 신호 처리 펑션 식별자에 응답하여 상기 디지털 신호 처리 펑션을 수행하는 것을 특징으로 하는 중앙 처리 장치.

청구항 2.

제 1 항에 있어서,

상기 명령 메모리는 상기 디지털 신호 처리 펑션을 수행하지 않는 명령들의 제 1 시퀀스를 저장하고, 상기 디지털 신호 처리 펑션을 수행하는 명령들의 제 2 시퀀스를 저장하며;

상기 적어도 1개의 범용 처리 코어는 상기 명령들의 제 1 시퀀스를 실행하고;

상기 적어도 1개의 디지털 신호 처리 코어는 상기 수신된 디지털 신호 처리 평션 식별자에 응답하여 상기 디지털 신호 처리 평션을 수행하며, 상기 디지털 신호 처리 코어에 의해 수행되는 상기 디지털 신호 처리 평션은 상기 명령들의 제 2 시퀀스의 실행과 같은 것을 특징으로 하는 중앙 처리 장치.

청구항 3.

제 1 항에 있어서,

상기 적어도 1개의 디지털 신호 처리 코어는 상기 적어도 1개의 범용 처리 코어에 데이터 및 타이밍 신호들을 제공하는 것을 특징으로 하는 중앙 처리 장치.

청구항 4.

제 1 항에 있어서,

상기 평션 프리프로세서는, 상기 결정 수단이 상기 명령 메모리의 상기 명령들의 시퀀스가 상기 디지털 신호 처리 평션을 수행하도록 의도되었다고 결정하는 것에 응답하여, 상기 디지털 신호 처리 평션 식별자 및 1개 이상의 파라미터들을 생성하는 것을 특징으로 하는 중앙 처리 장치.

청구항 5.

제 1 항에 있어서,

상기 적어도 1개의 범용 처리 코어는 X86 패밀리의 마이크로프로세서와 호환성인 것을 특징으로 하는 중앙 처리 장치.

청구항 6.

제 5 항에 있어서,

상기 복수의 명령들은 X86 연산 코드들인 것을 특징으로 하는 중앙 처리 장치.

청구항 7.

제 1 항에 있어서,

상기 적어도 1개의 디지털 신호 처리 코어는, 컨벌루션, 상관 관계, 고속 퓨리에 변환 및 내적으로 구성되는 그룹으로부터 1개 이상의 수학 연산들을 수행하는 것을 특징으로 하는 중앙 처리 장치.

청구항 8.

제 1 항에 있어서,

상기 적어도 1개의 범용 처리 코어 및 상기 적어도 1개의 디지털 신호 처리 코어는 동시에 동작하는 것을 특징으로 하는 중앙 처리 장치.

청구항 9.

중앙 처리 장치(CPU)-이 CPU는 적어도 1개의 범용 CPU 코어 및 적어도 1개의 디지털 신호 처리(DSP) 코어를 포함한다-에서 명령들을 실행하는 방법에 있어서,

상기 중앙 처리 유닛에 의해 실행될 수 있도록 명령 메모리에 1개 이상의 명령들의 시퀀스들을 저장하는 단계와;

상기 명령 메모리의 상기 명령들의 시퀀스를 검사하는 단계와;

상기 명령 메모리의 상기 명령들의 시퀀스가 디지털 신호 처리 펑션을 수행하도록 의도되었는 지의 여부를 결정하는 단계와;

상기 디지털 신호 처리 펑션을 수행하도록 의도된 상기 메모리의 상기 명령들의 시퀀스를 디지털 신호 처리 펑션 식별자로 변환하는 단계와;

상기 디지털 신호 처리 코어가 상기 디지털 신호 처리 펑션 식별자를 수신하는 단계와; 그리고

상기 디지털 신호 0처리 코어가 상기 수신된 디지털 신호 처리 펑션 식별자에 응답하여 상기 디지털 신호 처리 펑션을 수행하는 단계를 포함하는 것을 특징으로 하는 CPU에서 명령들을 실행하는 방법.

청구항 10.

제 9 항에 있어서,

상기 저장 단계는 제 1 디지털 신호 처리 펑션을 수행하는 제 1 명령 시퀀스를 상기 명령 메모리에 저장하고;

상기 저장 단계는 상기 제 1 디지털 신호 처리 펑션을 수행하지 않는 제 2 명령 시퀀스를 상기 명령 메모리에 저장하며,

상기 변환 단계는 상기 제 1 디지털 신호 처리 펑션을 수행하도록 의도된, 상기 명령 메모리 내의 상기 제 1 명령 시퀀스를 제 1 디지털 신호 처리 펑션 식별자로 변환하고;

상기 수행 단계는 상기 디지털 신호 처리 코어가 상기 제 1 디지털 신호 처리 펑션 식별자에 응답하여 상기 제 1 디지털 신호 처리 펑션을 수행하는 것을 포함하고, 상기 제 1 디지털 신호 처리 펑션의 수행은 상기 제 1 명령 시퀀스의 실행과 같으며; 그리고

상기 범용 중앙 처리 장치 코어가 상기 명령들의 제 2 시퀀스를 실행하는 것을 특징으로 하는 CPU에서 명령들을 실행하는 방법.

청구항 11.

제 10 항에 있어서,

상기 디지털 신호 처리 코어 및 상기 범용 중앙 처리 장치 코어는 동시에 동작하는 것을 특징으로 하는 CPU에서 명령들을 실행하는 방법.

청구항 12.

제 10 항에 있어서,

상기 디지털 신호 처리 코어는 상기 범용 중앙 처리 장치 코어에 데이터 및 타이밍 신호들을 제공하는 것을 특징으로 하는 CPU에서 명령들을 실행하는 방법.

청구항 13.

제 9 항에 있어서,

상기 펑션 프리프로세서가, 상기 명령 메모리의 상기 명령들의 시퀀스가 상기 디지털 신호 처리 펑션을 수행하도록 의도되었다는 상기 결정에 응답하여, 상기 디지털 신호 처리 펑션 식별자 및 1개 이상의 파라미터들을 생성하는 것을 특징으로 하는 CPU에서 명령들을 실행하는 방법.

청구항 14.

제 9 항에 있어서,

상기 범용 중앙 처리 장치 코어는 X86 패밀리의 마이크로프로세서와 호환성인 것을 특징으로 하는 CPU에서 명령들을 실행하는 방법.

청구항 15.

제 14 항에 있어서,

상기 1개 이상의 명령들의 시퀀스들은 X86 연산 코드들을 포함하는 것을 특징으로 하는 CPU에서 명령들을 실행하는 방법.

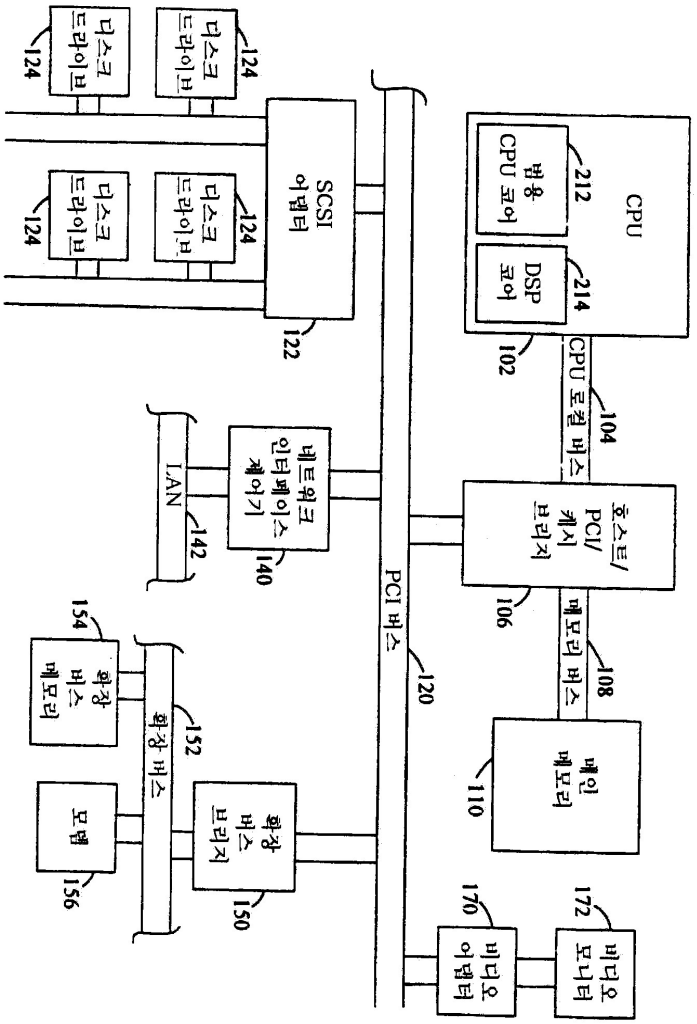
청구항 16.

제 9 항에 있어서,

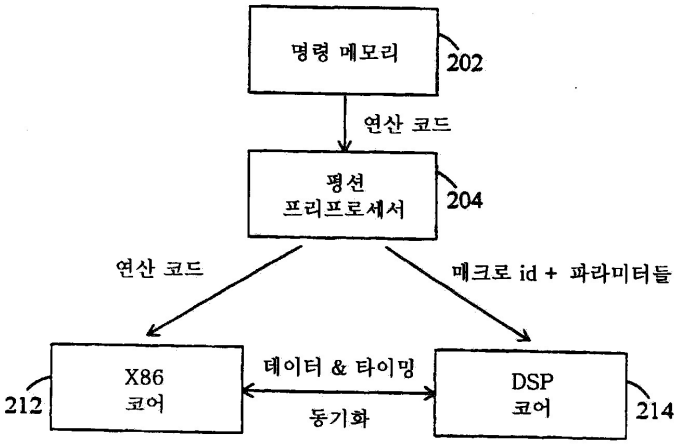
상기 디지털 신호 처리 코어는 컨벌루션, 상관 관계, 고속 퓨리에 변환 및 내적으로 구성되는 그룹으로부터 1개 이상의 수학 연산들을 수행하는 것을 특징으로 하는 CPU에서 명령들을 실행하는 방법.

도면

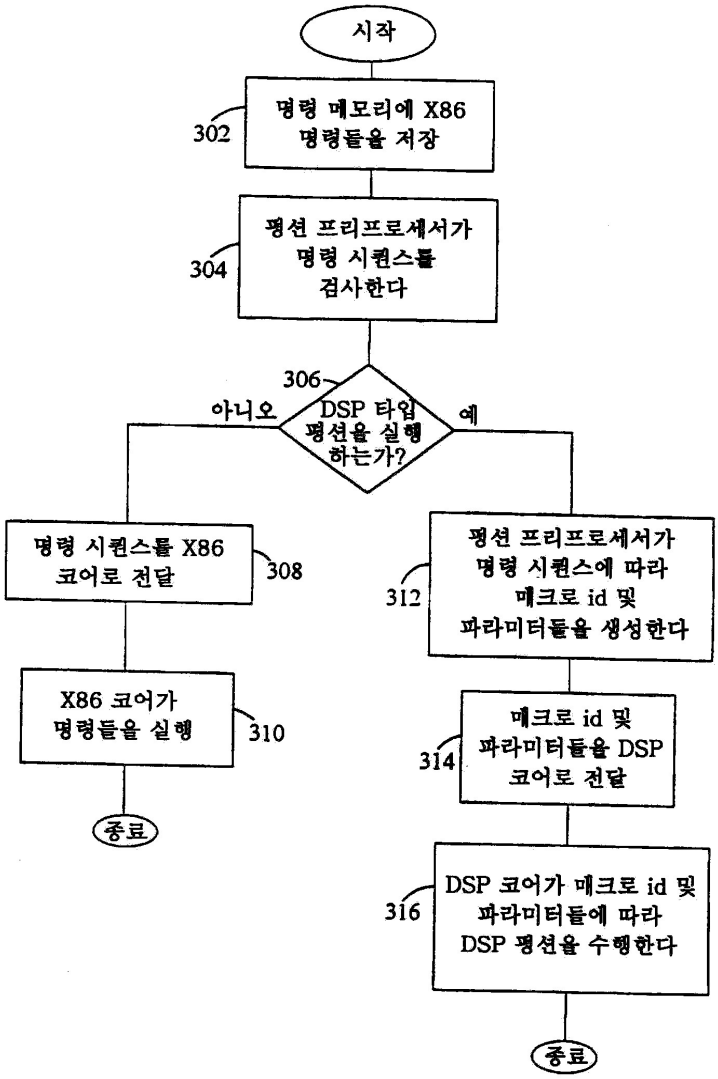
도면1



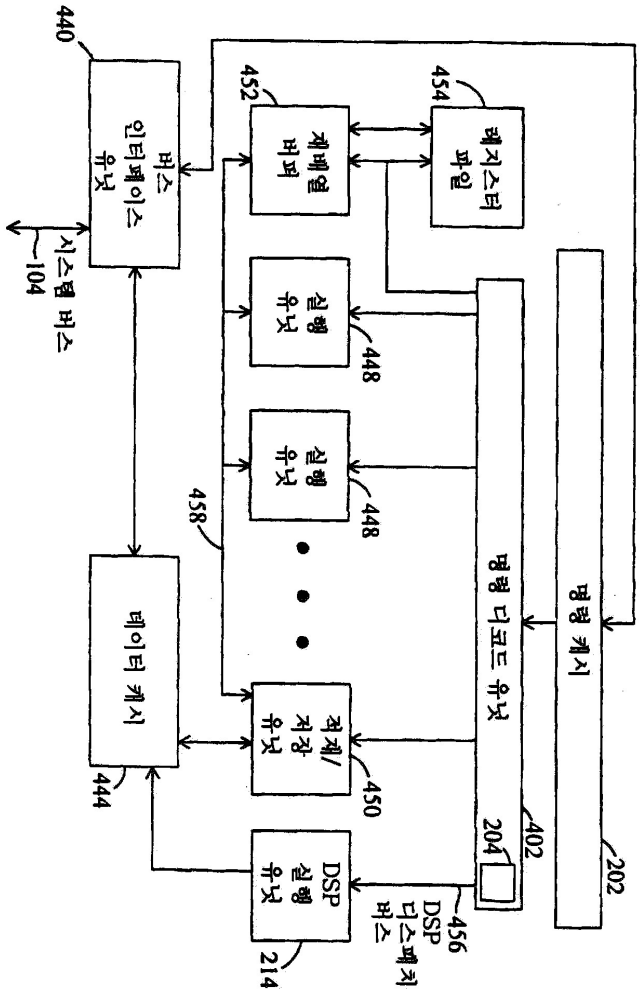
도면2


명령 메모리
202
연산 코드
펑션
프리프로세서
204
연산 코드
매크로 id + 파라미터들
X86
코어
212
데이터 & 타이밍
동기화
DSP
코어
214

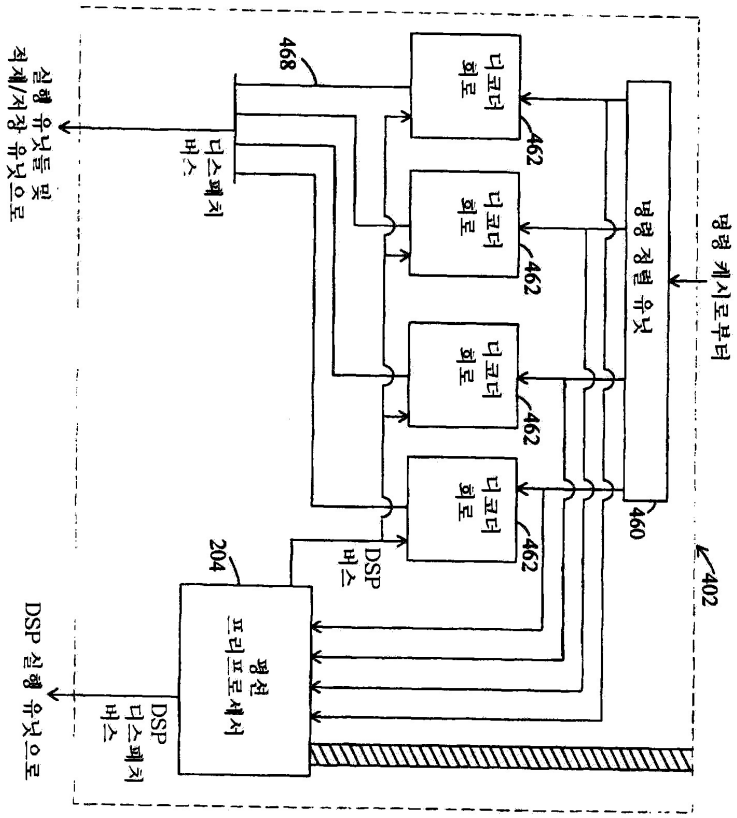
도면3


시작
명령 메모리에 X86
명령들을 저장
302
펑션 프리프로세서가
명령 시퀀스를
검사한다
304
306
DSP 타입
펑션을 실행
하는가?
아니오
예
명령 시퀀스를 X86
코어로 전달
308
X86 코어가
명령들을 실행
310
종료
펑션 프리프로세서가
명령 시퀀스에 따라
매크로 id 및
파라미터들을 생성한다
312
매크로 id 및
파라미터들을 DSP
코어로 전달
314
DSP 코어가 매크로 id 및
파라미터들에 따라
DSP 펑션을 수행한다
316
종료

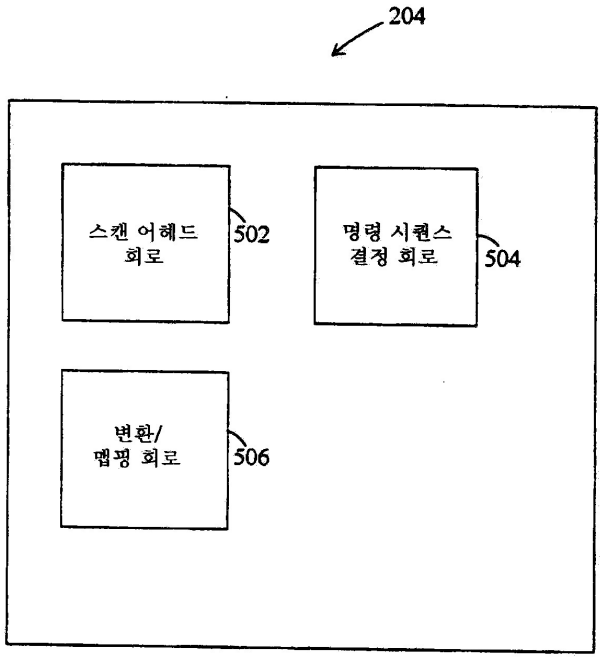
도면4


440
버스 인터페이스 유닛
시스템 버스
104
452
제네럴 버퍼
454
레지스터 파일
실행 유닛
448
실행 유닛
448
458
데이터 캐시
444
적재/저장 유닛
450
DSP 실행 유닛
214
명령 디코드 유닛
204
402
DSP 디스패치 버스
456
명령 캐시
202

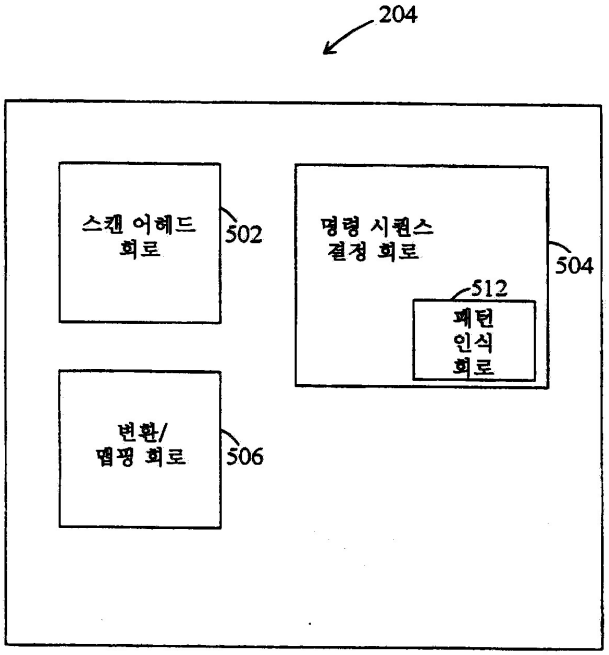
도면5


실행 유닛들 및 적재/저장 유닛으로
468
디스패치 버스
디코더 회로
462
디코더 회로
462
디코더 회로
462
디코더 회로
462
명령 정렬 유닛
460
명령 캐시로부터
402
DSP 버스
204
평션 프리프로세서
DSP 디스패치 버스
DSP 실행 유닛으로

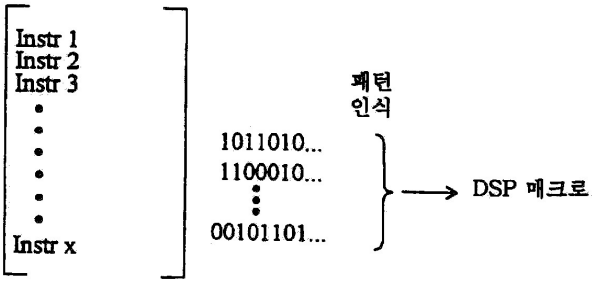
도면6


204
스캔 어헤드 회로
502
명령 시퀀스 결정 회로
504
변환/ 맵핑 회로
506

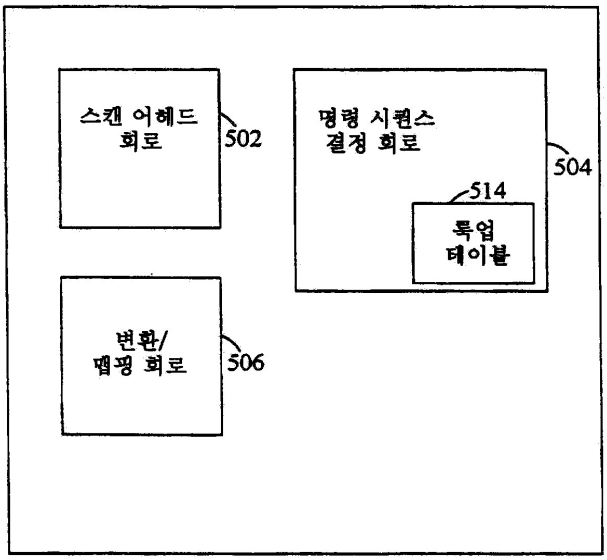
도면7


204
스캔 어헤드 회로
502
명령 시퀀스 결정 회로
504
512
패턴 인식 회로
변환/ 맵핑 회로
506

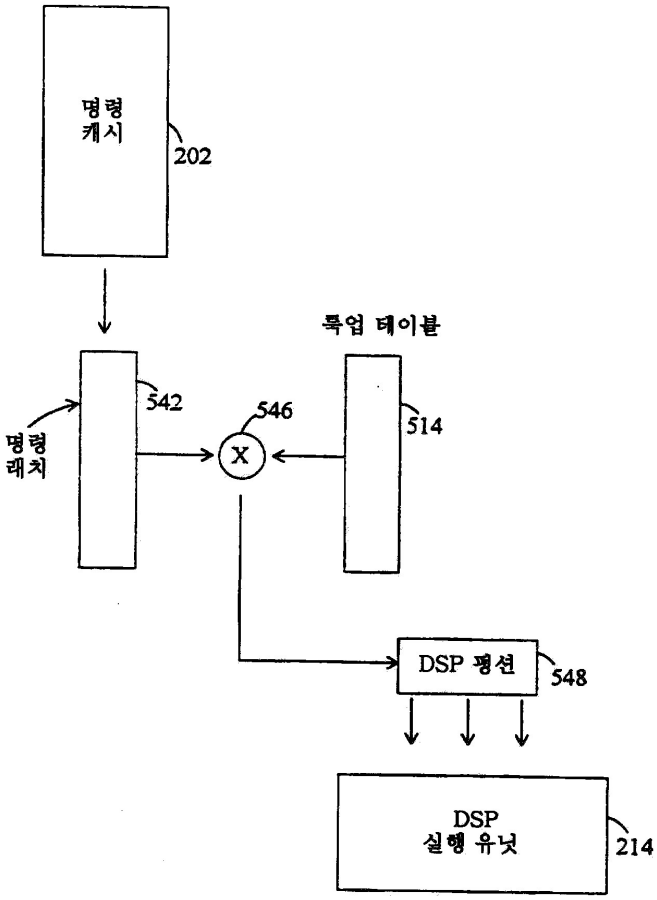
도면8


Instr 1
Instr 2
Instr 3
Instr x
패턴 인식
1011010...
1100010...
00101101...
DSP 매크로

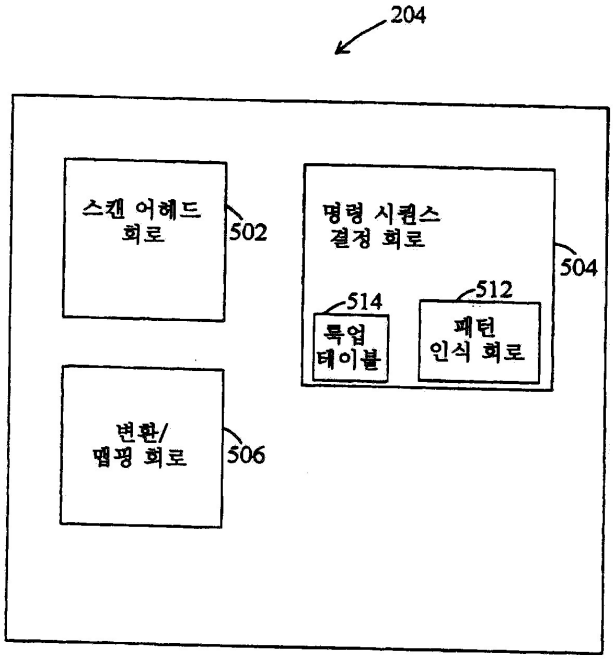
도면9


스캔 어헤드 회로
502
명령 시퀀스 결정 회로
504
514
룩업 테이블
변환/ 맵핑 회로
506

도면10


명령
캐시
202
룩업 테이블
542
546
514
명령
래치
X
DSP 평션
548
DSP
실행 유닛
214

도면11


204
스캔 어헤드
회로
502
명령 시퀀스
결정 회로
504
514
512
룩업
테이블
패턴
인식 회로
변환/
맵핑 회로
506