(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.[7]
G06F 9/38

(45) 공고일자 2005년09월23일
(11) 등록번호 10-0516214
(24) 등록일자 2005년09월13일

(21) 출원번호 10-2003-0015336
(22) 출원일자 2003년03월12일

(65) 공개번호 10-2004-0080520
(43) 공개일자 2004년09월20일

(73) 특허권자 (주)자람테크놀로지
경기도 성남시 분당구 야탑동 344-1 코리아디자인센터 406

전자부품연구원
경기도 성남시 분당구 야탑동 68번지

(72) 발명자 조경연
경기도 평택시 진위면 마산리 455-1 전자부품연구원 시스템I C연구센터

이승은
경기도 평택시 진위면 마산리 455-1 전자부품연구원 시스템I C연구센터

최종찬
경기도 평택시 진위면 마산리 455-1 전자부품연구원 시스템 IC연구센터

백준현
서울특별시 서초구 반포동 한신서래아파트 3-601

박성훈
경기도 성남시 분당구 서현동 310번지 효자촌 617동 905호

이지현
서울특별시 은평구 불광1동 242-9 3층

(74) 대리인 서천석

심사관 : 성경아

(54) 명령어 병렬처리를 위한 디지털 신호처리기 및 그처리방법

요약

본 발명은 명령어 병렬처리를 위한 디지털 신호처리기 및 그 처리방법에 관한 것으로, 특히 슈퍼스칼라(Superscalar) 구조와 VLIW(Very Long Instruction Word) 구조의 장점을 수용한 고성능 신호처리 시스템에 적용 가능한 디지털 신호처리기(Digital Signal Processor, 이하 DSP)에 관한 것이다.

본 발명의 명령어 병렬처리를 위한 디지털 신호처리기 및 그 처리방법은 동시에 수행 가능한 다음 명령어들의 수(M)를 계산하는 제1단계, 제1단계에서 계산한 M중에서 최대값 하나를 선택하는 제2단계, 제2단계에서 선택한 값에 대응하는 명령어들을 그룹으로 만드는 제3단계, 모든 명령어들이 그룹으로 만들어졌다면 제5단계를 수행하고, 그렇지 않으면 제1단계를 수행하는 제4단계, 각각의 그룹에 있는 명령어 들을 해당 명령어를 수행할 연산 블록에 따라 재배치하고, 태그(Tag) 값을 할당하는 제5단계로 하는 처리로 알고리즘이 수행됨에 기술적 특징이 있다.

따라서, 본 발명의 명령어 병렬처리를 위한 디지털 신호처리기 및 그 처리방법은 슈퍼스칼라 구조와 VLIW 구조의 장점을 수용한 고성능 신호처리 시스템에 적용이 가능한 DSP 구조를 제공하는데, 간단한 플래그(Flag)를 이용하여 무연산(No OPeration) 명령을 DSP가 생성하도록 함으로써 프로그램 사이즈를 줄이고, 병렬로 처리할 수 있는 명령어들을 소프트웨어(컴파일러, 어셈블러)가 구성하도록 함으로써 DSP 구조를 간단하게 할 수 있는 장점이 있다.

대표도

도 5

색인어

명령어, 병렬처리, 디지털 신호처리기

명세서

도면의 간단한 설명

도 1은 본 발명의 간략한 명령어 구성도이다.

도 2는 본 발명의 디지털 신호처리기의 간략한 블록도이다.

도 3은 본 발명의 태그의 구성도이다.

도 4는 본 발명의 재배치 전의 명령어 구성도이다.

도 5는 본 발명의 무조건 분기 명령의 예이다.

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 명령어 병렬처리를 위한 디지털 신호처리기 및 그 처리방법에 관한 것으로, 특히 슈퍼스칼라(Superscalar) 구조와 VLIW(Very Long Instruction Word) 구조의 장점을 수용한 고성능 신호처리 시스템에 적용이 가능한 디지털 신호처리기(Digital Signal Processor, 이하 DSP)에 관한 것이다.

SDR(Software Defined Radio) 기술이 대두됨에 따라 통신을 비롯한 이미지, 비디오 등의 신호처리 시스템을 소프트웨어로 처리하기 위한 고성능 DSP가 요구된다. 신호처리용 소프트웨어를 고속으로 처리하기 위하여 현재 일반적으로 사용되는 방식은 ILP(Instruction Level Parallelism)를 사용하는 것이다. 상기 ILP를 구현하기 위하여 상업적으로 제품화되고 있는 기술은 크게 두 가지로 분류할 수 있다.

종래에는 DSP 구조로서 슈퍼스칼라 구조와 VLIW 구조의 두 가지 구조가 있었는데, 대한민국 공개특허공보 제1993-0016896호, 제1999-0062575호, 제1992-0020340호 등에서 자세히 소개하고 있는데, 다음과 같다.

슈퍼스칼라 구조는 ILP구현을 위하여 DSP에 의존한다. 즉, 명령어가 저장된 메모리로부터 정해진 수의 명령어를 읽어온 후에 DSP가 동시에 처리할 수 있는 명령어들을 판단하여 그 명령어들을 병렬로 처리한다. 만약, 읽어온 명령어 중에서 동시에 처리할 수 없는 명령어의 경우에 다음에 읽어온 명령어들과 함께 병렬처리가 가능한 명령어들을 다시 판단하여 처리하게 된다. 즉, 슈퍼스칼라 구조의 경우에 명령어 병렬 처리를 DSP가 하드웨어적으로 처리하기 때문에 DSP 구조가 복잡하고, 읽어온 명령어 중 병렬로 처리할 수 없는 명령어들을 저장할 명령어 버퍼가 필요하다.

VLIW 구조는 ILP 구현을 위하여 소프트웨어(컴파일러 또는 어셈블러)에 의존한다. 즉, 명령어 자체에 DSP에 있는 각각의 연산 모듈에서 수행할 명령들을 포함시켜 하나의 긴 명령어를 만든다. 이 때, DSP는 명령어가 저장된 메모리로부터 정해진 수의 명령어를 읽어온 후에 각각의 연산 모듈에서 수행할 연산 모듈을 위한 명령들을 해당하는 연산 모듈로 전달하여 처리하게 된다. 그러므로, 한 번 읽어온 명령어는 바로 처리를 하기 때문에 명령어 버퍼가 필요없고, 명령어 중에서 동시에 처리할 수 있는 명령어를 판단할 필요가 없기 때문에 슈퍼스칼라 구조보다 하드웨어가 간단하다.

그러나, 상기와 같은 종래기술의 프로세서 구조는 모든 연산 블록이 동시에 수행될 수 없을 경우에 무연산(No OPeration) 명령이 많아지게 된다. 이것은 DSP가 아무 동작도 하지 않는 명령인데, 결과적으로 프로그램 사이즈가 커지게 된다. 특히, 상기 대한민국 공개특허공보 제1993-0016896호에서는 상기 슈퍼스칼라 구조와 VLIW 구조의 방식을 혼재시켜서 두 구조의 장점을 이용한 병렬 연산 기능을 수행할 수 있는 계산기를 소개하고 있는데, 본 발명에서와 같이 그 알고리즘을 제시하지는 못하고 있다.

발명이 이루고자 하는 기술적 과제

따라서, 본 발명은 상기와 같은 종래기술의 문제점을 해결하기 위한 것으로, 상기 슈퍼스칼라 구조와 VLIW 구조는 각각 장단점을 가지고 있기 때문에 현재 상용 제품을 판매하는 기업의 경우에 상기 두 구조의 장점을 이용하여 종래의 제품보다 효율적인 DSP 구조를 제안하고 있다. 본 발명에서는 이러한 신호처리 시스템을 구현할 수 있는 고성능 DSP 구조를 제안하며, 상기의 제안 방식은 슈퍼스칼라 구조와 VLIW 구조의 장점을 수용하여 고성능 신호처리 시스템에 적용 가능한 DSP 구조를 제안함에 본 발명의 목적이 있다.

발명의 구성 및 작용

본 발명의 상기 목적은 슈퍼스칼라 구조와 VLIW 구조의 장점을 수용한 고성능 신호처리 시스템에 적용 가능한 DSP 구조를 제공함에 의해서 달성된다.

본 발명의 상기 목적과 기술적 구성 및 그에 따른 작용 효과에 관한 자세한 사항은 본 발명의 바람직한 실시예를 도시하고 있는 도면을 통하여 상세히 설명하고자 한다.

먼저 본 발명의 프로그램 작성 알고리즘을 위한 4가지의 기본적인 사항을 살펴보도록 한다.

첫째, 명령어 및 태그(이하 Tag)이다. 본 발명에서는 모든 명령어들은 고정된 비트수(예: 32비트)로 정의된다. 즉, RISC(Reduced Instruction Set Computer) 프로세서와 같이 모든 명령어가 같은 비트의 수로 구성되도록 한다. 그리고, DSP에 동시에 처리할 수 있는 연산 블록의 수가 e개이면 각 명령어에 e비트의 Tag를 포함시킨다. 포함되는 e비트의 Tag는 다음에 수행될 명령어 집합에 대한 정보를 가지게 된다. 즉, 도 1과 같이 각 명령어는 실제 명령과 e비트의 Tag로 구성된다. 도 2는 DSP의 간략한 블록도를 나타낸 것으로 명령어를 읽어서 처리하는 디코드(Decode) 블록과 e개의 연산 블록으로 구성된다. 태그는 e비트의 플래그(이하 Flag)들로 구성되는데, 각 Flag는 연산 블록을 지시한다. 도 3은 Tag를 자세히 나타낸 것이다. 각 Flag의 의미는 수학식 1로 정의된다.

수학식 1

$$Flag(i)(i = 1, 2, ... , e-1, e) = 0 \text{ 또는 } 1$$

삭제

여기서, Flag(i) = 0 일 때는 연산 블록 i는 다음 연산 수행 시 수행이 가능하지 않으나, Flag(i) = 1 일 때는 연산 블록 i는 다음 연산 수행 시 수행이 가능하다.

그러므로, 다음에 처리할 명령어 수(Num)는 수학식 2로 계산된다.

수학식 2

$$Num = Flag(1) + Flag(2) + ... Flag(e-1) + Flag(e)$$

삭제

예를 들어, 도 4와 같이 명령어와 Tag가 구성되는 경우, 명령 1을 수행한 후 DSP는 다음에 있는 3개의 명령어를 읽어와서 처리한다. 왜냐하면, Flag(2), Flag(3), 그리고, Flag(6)이 1이고 나머지 Flag가 0이기 때문이다. 이 때, Flag(1) 이 0이기 때문에 다음 연산 수행시 연산 블록 1은 아무런 연산도 수행하지 않으며, Flag(2)가 1이기 때문에 명령 2는 연산 블록 2에서 처리하도록 한다. 상기와 같은 방법으로 Flag(3)이 1이기 때문에 명령 3은 연산 블록 3에서 처리하도록 하고, Flag(6)이 1이기 때문에 명령 4는 연산 블록 6에서 처리하도록 하고, 다른 연산 블록들은 아무 연산도 처리하지 않게된다. 즉, 연산 블록 1, 연산 블록 4, 연산 블록 5, 연산 블록 7, 그리고 연산 블록 8에는 무연산 명령을 전달하게 된다. 이는 명령어에는 존재하지 않는 무연산 명령을 DSP가 생성하는 것이다. 즉, 본 발명에서는 간단한 Flag를 이용하여 무연산 명령을 DSP가 생성하도록 함으로써 프로그램 사이즈를 줄이고, 병렬로 처리할 수 있는 명령어들을 소프트웨어(컴파일러, 어셈블러)가 구성하도록 하게함으로써 DSP 구조를 간단하게 할 수 있다.

둘째, 프로그램의 첫 번째 명령이다. 프로그램의 첫 번째 명령은 반드시 무연산 명령이고, Tag는 다음에 처리할 명령어에 대한 정보를 포함한다. 그리고, DSP가 리셋된 후에 첫 번째 명령을 수행할 때는 한 개의 명령어만 읽어와서 Tag에 포함되어 있는 플래그들을 분석하여 다음에 수행할 명령어들을 읽어온다. 이것은 DSP가 리셋된 후에 명령을 수행하기 시작할 때 Tag 정보가 없기 때문에 몇 개의 명령어를 읽어와야 할 것인지와 어느 연산 블록에서 처리할 지를 판단할 수 없기 때문이다.

셋째, 무조건 분기(Unconditional Branch) 명령이다. 무조건 분기 명령의 경우 목적지 주소가 미리 결정되기 때문에 Tag가 미리 결정될 수 있다. 이 때, Tag는 목적지 명령어 바로 전 명령어의 Tag가 된다. 즉, 도 5의 경우에 무조건 분기 명령의 Tag인 Tag(B)는 수학식 3으로 구해진다.

수학식 3

$$Tag(B) = Tag(N-1)$$

넷째, 조건 분기(Conditional Branch) 명령이다. 조건 분기 명령의 경우에 목적지 주소가 미리 결정되지 않기 때문에 Tag가 미리 결정될 수 없다. 그리고, 조건분기 명령어 다음에는 무연산(No OPeration) 명령이 삽입된다. 조건 분기 명령어의 Tag는 조건이 만족되었을 때 분기할 주소에 저장되어 있는 하나의 명령을 수행하기 위한 Tag가 된다. 그리고, 조건 분기 명령어 다음에 있는 무연산 명령어의 Tag는 다음에 수행할 명령어들에 대한 Tag가 된다.

삭제

삭제

상기 네 가지의 기본적인 사항으로 본 발명에서 제안하는 DSP 구조를 위한 프로그램 작성(명령어 배치)은 알고리즘 1에 따라 작성된다.

[알고리즘 1]

1단계: 각 명령에 대하여 동시에 수행 가능한 다음 명령어들의 수(M)를 계산한다.

2단계: 1단계에서 계산한 M중에서 최대값 하나를 선택한다.

3단계: 2단계에서 선택한 값에 대응하는 명령어들을 그룹으로 만든다.

4단계: 모든 명령어들이 그룹으로 만들어졌다면 5단계를 수행하고, 그렇지 않으면 1단계를 수행한다.

5단계: 각각의 그룹에 있는 명령어들을 해당 명령어를 수행할 연산 블록에 따라 재배치하고, Tag값을 해당하는 명령에 할당한다.

알고리즘 1에 의하여 명령어를 배치하면 한 그룹 내에서 명령어들은 그 명령어가 수행될 연산 블록의 순서에따라 배치된다. 즉, I_i 를 연산 블록 E_i 에서 수행 가능한 명령어라고 하면 하나의 그룹 내에서 수학식 4를 만족한다. 여기서, AddressOf(X)는 X 명령어가 저장되어 있는 메모리의 주소를 나타낸다.

수학식 4

$$\text{AddressOf}(I_m) < \text{AddressOf}(I_n) \rightarrow m < n$$

삭제

그러면, 프로그램 작성(명령어 배치) 방법을 예를 들어서 설명하기로 한다.

수행할 명령어로 무연산, I_2, I_1, I_3, I_5, I_2, I_1과 같은 명령어가 수행된다고 가정하면 알고리즘 1의 1단계에서 동시에 수행 가능한 다음 명령어들의 수(M)를 계산 하면 표 1과 같다.

표 1.

명령어	무연산	I_2	I_1	I_3	I_5	I_2	I_1
M	4	3	3	3	2	1	0

알고리즘 1의 2단계로부터 무연산 다음의 4개의 명령어들을 하나의 그룹으로 형성한다. 그러면, 제일 마지막의 I_2와 I_1이 하나의 그룹이 된다.

알고리즘 1의 5단계로부터 각 그룹에 있는 명령어들을 재배치하면 무연산, I_1, I_2, I_3, I_5, I_1, I_2와 같이 명령어들이 배치되고, 각 명령어에 대한 Tag값을 계산하면, 표 2와 같이 된다.

표 2.

명령어	Tag
무연산	11101 (Tag 1)
I_1	01101
I_2	00101
I_3	00001
I_5	11000 (Tag 5)
I_1	01000
I_2	00000

DSP는 첫 번째 무연산 명령과 Tag 1을 읽어와서 Tag 1을 분석한다. Tag 1을 분석하면 다음에 수행할 명령어가 4개라는 것을 알 수 있기 때문에 DSP는 프로그램 메모리로부터 4개의 명령어를 읽어온다. 그리고, Flag(1)이 1이기 때문에 첫 번째 명령어를 연산 블록 1로 전달한다. 상기와 같은 방법으로 두 번째 명령어를 연산 블록 2로 전달하고, 세 번째 명령어를 연산블록 3으로 전달한다. 그리고, Flag(4)가 0이기 때문에 연산블록 4에 무연산을 전달하고, 네 번째 명령어를 연산블록 5로 전달한다. 그리고, Tag 5를 분석하여 다음에 읽어올 명령어를 결정한다. 다음에 읽어올 명령어의 수는 2개이고, 첫 번째 명령어를 연산 블록 1로 전달하고, 두 번째 명령어를 연산 블록 2로 전달한다.

그룹 내에 마지막 명령어의 Tag값은 다음에 수행할 명령어 그룹에 대한 정보를 가지고 있게 된다. 그리고, 다른 명령어들에 대한 Tag값은 그 그룹에서 그 명령어 다음에 남아 있는 명령어들에 대응하는 Tag값으로 결정된다.

발명의 효과

따라서, 본 발명의 명령어 병렬처리를 위한 디지털 신호처리기 및 그 처리방법은 슈퍼스칼라 구조와 VLIW 구조의 장점을 수용한 고성능 신호처리 시스템에 적용이 가능한 DSP 구조를 제공하는데, 간단한 Flag를 이용하여 무연산 명령을 DSP가 생성하도록 함으로써 프로그램 사이즈를 줄이고, 병렬로 처리할 수 있는 명령어들을 소프트웨어(컴파일러, 어셈블러)가 구성하도록 함으로써 DSP 구조를 간단하게 할 수 있는 장점이 있다.

(57) 청구의 범위

청구항 1.

디지털 신호처리 방법에 있어서,

동시에 수행 가능한 다음 명령어들의 수(M)를 계산하는 제1단계;

제1단계에서 계산한 M중에서 최대값 하나를 선택하는 제2단계;

제2단계에서 선택한 값에 대응하는 명령어들을 그룹으로 만드는 제3단계;

모든 명령어들이 그룹으로 만들어졌다면 제5단계를 수행하고, 그렇지 않으면 명령어 그룹으로 만들어지지 않은 나머지 명령어들에 대해 제1단계를 수행하는 제4단계; 및

각각의 그룹에 있는 명령어들을 해당 명령어를 수행할 연산 블록에 따라 재배치하고, 태그(Tag)값을 할당하는 제5단계

를 포함하는 것을 특징으로 하는 명령어 병렬처리를 위한 디지털 신호처리기의 처리방법

청구항 2.

제 1 항에 있어서,

상기 명령어는 고정된 비트수임을 특징으로 하는 명령어 병렬처리를 위한 디지털 신호처리기의 처리방법

청구항 3.

제 1 항에 있어서,

디지털 신호처리기에 동시에 처리할 수 있는 상기 연산 블록의 수가 e개이면 상기 각 명령어에 e비트의 태그를 포함시킴을 특징으로 하는 명령어 병렬처리를 위한 디지털 신호처리기의 처리방법

청구항 4.

제 1 항에 있어서,

상기 명령어들의 수(M)는 Num = Flag(1) + Flag(2) + ... Flag(e-1) + Flag(e)(여기서, e는 연산 블록의 수, Flag(e)는 연산블록을 지시하는 지시자)에 의하여 계산됨을 특징으로 하는 명령어 병렬처리를 위한 디지털 신호처리기의 처리방법

청구항 5.

제 4 항에 있어서,

상기 Flag(i)(i = 1, 2, ... , e-1, e) = 1 일 때는 연산 블록 i는 다음 연산 수행 시 수행이 가능함을 특징으로 하는 명령어 병렬처리를 위한 디지털 신호처리기의 처리방법

청구항 6.

제 1 항에 있어서,

상기 명령어의 첫 번째 명령은 무연산 명령임을 특징으로 하는 명령어 병렬처리를 위한 디지털 신호처리기의 처리방법

청구항 7.

제 6 항에 있어서,

디지털 신호처리기가 리셋된 후에 상기 첫 번째 명령을 수행할 때는 한 개의 명령어만 읽어와서 태그에 포함되어 있는 플래그들을 분석하여 다음에 수행할 명령어들을 읽어옴을 특징으로 하는 명령어 병렬처리를 위한 디지털 신호처리기의 처리방법

청구항 8.

제 1 항에 있어서,

상기 태그는 다음에 처리할 명령어에 대한 정보를 포함함을 특징으로 하는 명령어 병렬처리를 위한 디지털 신호처리기의 처리방법

청구항 9.

제 1 항에 있어서,

무조건 분기 명령의 상기 태그는 목적지 명령어 바로 전 명령어의 태그가 됨을 특징으로 하는 명령어 병렬처리를 위한 디지털 신호처리기의 처리방법

청구항 10.

제 1 항에 있어서,

조건 분기의 상기 명령어 다음에는 무연산 명령이 삽입됨을 특징으로 하는 명령어 병렬처리를 위한 디지털 신호처리기의 처리방법

청구항 11.

제 10 항에 있어서,

상기 조건 분기의 명령어의 태그는 조건이 만족되었을 때, 분기할 주소에 저장되어 있는 하나의 명령을 수행하기 위한 태그가 됨을 특징으로 하는 명령어 병렬처리를 위한 디지털 신호처리기의 처리방법

청구항 12.

제 11 항에 있어서,

조건 분기 명령어의 다음에 있는 무연산 명령어의 상기 태그는 다음에 수행할 명령어들에 대한 태그가 됨을 특징으로 하는 명령어 병렬처리를 위한 디지털 신호처리기의 처리방법

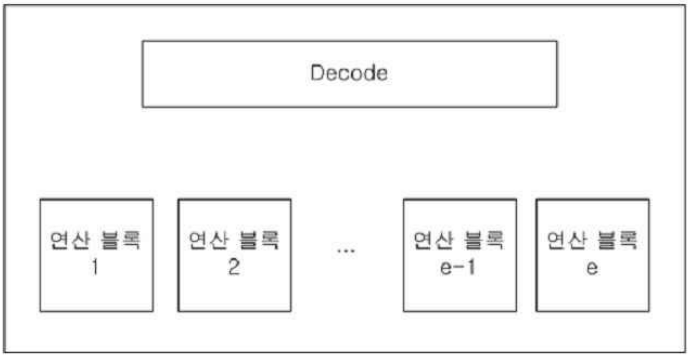
청구항 13.

상기 제 1 항의 방법에 의해 알고리즘이 수행됨을 특징으로 하는 명령어 병렬처리를 위한 디지털 신호처리기

도면

도면1

도면2



도면3

도면4

도면5