(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2018년12월28일
(11) 등록번호 10-1910934
(24) 등록일자 2018년10월17일

(51) 국제특허분류(Int. Cl.)
G06F 9/06 (2018.01) *G06F 9/38* (2006.01)
G06F 9/46 (2006.01)
(21) 출원번호 10-2012-0030695
(22) 출원일자 2012년03월26일
심사청구일자 2017년01월03일
(65) 공개번호 10-2013-0108878
(43) 공개일자 2013년10월07일
(56) 선행기술조사문헌
JP2003108387 A*
(뒷면에 계속)

(73) 특허권자
삼성전자 주식회사
경기도 수원시 영통구 삼성로 129 (매탄동)
서울대학교 산학협력단
서울특별시 관악구 관악로 1 (신림동)
(72) 발명자
버나드 에거
서울 관악구 관악로 1, 컴퓨터공학부 (신림동, 서울대학교)
김원섭
경기 안양시 동안구 흥안대로414번길 39, 201동 201호 (평촌동, 인덕원대림2차아파트)
정성훈
서울 성동구 사근동8길 11, (사근동)
(74) 대리인
특허법인가산

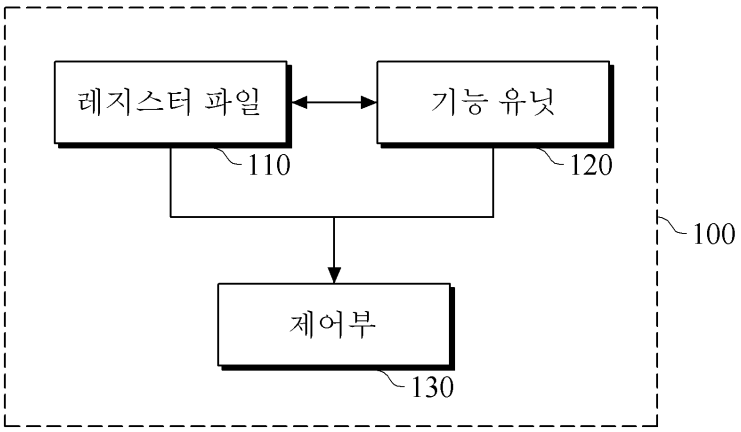
전체 청구항 수 : 총 10 항 심사관 : 김경완

(54) 발명의 명칭 **루프의 프롤로그 또는 에필로그의 비유효 연산을 처리하는 장치 및 방법**

(57) 요 약

소프트웨어 파이프라인된 루프의 프롤로그 또는 에필로그에서 비유효 연산(invalid operation)을 처리하는 장치에 관한 것이다. 일 실시예에 따른 비유효 연산 처리 장치는 데이터의 유효 상태를 저장하는 제1 영역과 그 데이터를 저장하는 제2 영역을 포함하는 레지스터 파일 및 그 레지스터 파일로부터 입력되는 하나 이상의 입력 소스의 제1 영역 값에 기초하여 연산의 비유효(invalid) 여부를 판단하고 제1 영역의 대응값을 포함하는 데스티네이션(destination)을 출력하는 하나 이상의 기능 유닛(functional unit)을 포함할 수 있다. 본 실시예에 따르면 비유효 연산의 가딩을 위해 프리디케이트를 사용하지 않기 때문에 컴파일러의 성능을 향상시킬 수 있다.

대 표 도 - 도1



(56) 선행기술조사문헌
KR1020020004346 A*
US20090070552 A1*
US08099585 B2*
JP3006204 B2*
*는 심사관에 의하여 인용된 문헌

명 세 서

청구범위

청구항 1

데이터의 유효 상태를 저장하는 제1 영역과 그 데이터를 저장하는 제2 영역을 포함하는 레지스터 파일; 및

상기 레지스터 파일로부터 입력되는 하나 이상의 입력 소스의 제1 영역 값에 기초하여 연산의 비유효(invalid) 여부를 판단하고 상기 제1 영역의 대응값을 포함하는 데스티네이션(destination)을 출력하는 하나 이상의 기능 유닛(functional unit); 및

루프의 시작과 동시에 발생되는 리셋 요청에 따라, 상기 레지스터 파일 내의 제1 영역을 리셋하는 제어부를 포함하는 루프의 프롤로그 또는 에필로그에서 비유효 연산을 처리하는 장치.

청구항 2

제1항에 있어서, 상기 기능 유닛은,

상기 입력 소스들의 제1 영역의 값들을 'AND' 연산하여 상기 비유효 연산 여부를 판단하는 루프의 프롤로그 또는 에필로그에서 비유효 연산을 처리하는 장치.

청구항 3

제2항에 있어서, 상기 기능 유닛은,

프리디케이트(predicate)를 더 'AND' 연산하여 상기 비유효 연산 여부를 판단하는 루프의 프롤로그 또는 에필로그에서 비유효 연산을 처리하는 장치.

청구항 4

제1항에 있어서,

상기 제어부는 상기 데스티네이션의 제1 영역의 대응값에 기초하여 그 데스티네이션에 대한 상기 레지스터 파일로의 쓰기 요청을 처리하는 루프의 프롤로그 또는 에필로그에서 비유효 연산을 처리하는 장치.

청구항 5

제4항에 있어서, 상기 제어부는,

상기 데스티네이션의 제1 영역의 대응값과 쓰기 허용(Write Enable, WE) 신호를 'AND' 연산하여 그 결과에 따라 상기 쓰기 요청을 처리하는 루프의 프롤로그 또는 에필로그에서 비유효 연산을 처리하는 장치.

청구항 6

삭제

청구항 7

제1항에 있어서, 상기 제어부는,

상기 리셋 요청에 따라 VLIW(Very Long Instruction Word) 모드에서 CGRA(Coarse Grained Reconfigurable Array) 모드로 전송되는 레지스터 값은 제외하고 리셋하는 루프의 프롤로그 또는 에필로그에서 비유효 연산을 처리하는 장치.

청구항 8

데이터의 유효 상태를 저장하는 제1 영역과 그 데이터를 저장하는 제2 영역을 포함하는 레지스터 파일로부터 하나 이상의 입력 소스가 하나 이상의 기능 유닛(functional unit)으로 입력되는 단계;

상기 기능 유닛이 상기 입력 소스의 제1 영역 값에 기초하여 연산의 비유효(invalid) 여부를 판단하는 단계;

상기 기능 유닛이 상기 제1 영역의 대응값을 포함하는 데스티네이션(destination)을 출력하는 단계; 및

루프의 시작과 동시에 발생되는 리셋 요청에 따라, 제어부가 상기 레지스터 파일 내의 제1 영역을 리셋하는 단계를 포함하는 루프의 프롤로그 또는 에필로그에서 비유효 연산을 처리하는 방법.

청구항 9

제8항에 있어서, 상기 연산의 비유효 여부를 판단하는 단계는,

상기 입력 소스들의 제1 영역의 값들을 'AND' 연산하여 상기 비유효 연산 여부를 판단하는 루프의 프롤로그 또는 에필로그에서 비유효 연산을 처리하는 방법.

청구항 10

제9항에 있어서, 상기 연산의 비유효 여부를 판단하는 단계는,

프리디케이트(predicate)를 더 'AND' 연산하여 상기 비유효 연산 여부를 판단하는 루프의 프롤로그 또는 에필로그에서 비유효 연산을 처리하는 방법.

청구항 11

제8항에 있어서,

상기 제어부가 상기 데스티네이션의 제1 영역의 대응값에 기초하여 그 데스티네이션에 대한 상기 레지스터 파일로의 쓰기 요청을 처리하는 단계;를 더 포함하는 루프의 프롤로그 또는 에필로그에서 비유효 연산을 처리하는 방법.

청구항 12

◈청구항 12은(는) 설정등록료 납부시 포기되었습니다.◈

제11항에 있어서, 상기 데스티네이션의 쓰기 요청을 처리하는 단계는,

상기 데스티네이션의 제1 영역의 대응값과 쓰기 허용(Write Enable, WE) 신호를 'AND' 연산하여 그 결과에 따라 상기 쓰기 요청을 처리하는 루프의 프롤로그 또는 에필로그에서 비유효 연산을 처리하는 방법.

발명의 설명

기 술 분 야

[0001] 소프트웨어 파이프라인된 루프의 프롤로그 또는 에필로그에서 비유효 연산(invalid operation)을 처리하는 기술과 관련된다.

배 경 기 술

[0002] 프로세서에서 소프트웨어 파이프라인(software pipeline) 방법은 수행할 소프트웨어를 단계별로 나누어 파이프라인 형태로 수행하는 방법으로서 처리 성능을 크게 높일 수 있다. 그러나, CGRA(coarse-grained reconfigurable architecture) 프로세서는 모듈로 스케쥴링(modulo scheduling)을 통해 프로그램을 스케쥴링하기 때문에 루프의 프롤로그나 에필로그에서 비유효 연산(invalid operation)이 포함될 수 있다.

[0003] 일반적으로 CGRA 프로세서는 루프의 프롤로그 또는 에필로그의 비유효 연산이 프로그램 상태를 변경하는 것을 가딩(guarding)하기 위해 프리디케이트(predicate) 정보를 사용한다. 가딩이란 그 비유효 연산의 실행을 제어하는 것으로서, 그 비유효 연산이 기능 유닛에서 실제 수행되더라도 레지스터 파일에 기록되지 않도록 하는 것을 말한다. 그 프리디케이트 정보를 사용하기 위해서는 프리디케이트를 위한 CGRA 내의 라우팅 정보가 필요하다. 이때, 프리디케이트의 계산과 라우팅은 CGRA 컴파일러가 담당하는데 이는 컴파일러의 스케쥴링 과정을 매우 복잡하게 하는 원인이 되며, 복잡한 루프의 커널의 경우에는 이로 인하여 스케쥴링을 실패하는 경우도 발생하게 된다.

발명의 내용

해결하려는 과제

[0004] 프로세서에서 데이터 폭(data width)을 일정 비트 확장하여 데이터의 유효 여부를 저장하고, 그 비트를 이용하여 비유효 연산을 처리함으로써 스케쥴링의 복잡성을 감소시키는 장치 및 방법이 제시된다.

과제의 해결 수단

[0005] 일 양상에 따르면, 루프의 프롤로그 또는 에필로그에서 비유효 연산을 처리하는 장치는, 데이터의 유효 상태를 저장하는 제1 영역과 그 데이터를 저장하는 제2 영역을 포함하는 레지스터 파일 및 레지스터 파일로부터 입력되는 하나 이상의 입력 소스의 제1 영역 값에 기초하여 연산의 비유효(invalid) 여부를 판단하고 제1 영역의 대응값을 포함하는 데스티네이션(destination)을 출력하는 하나 이상의 기능 유닛(functional unit)을 포함한다.

[0006] 이때, 기능 유닛은 그 입력 소스들의 제1 영역의 값들을 'AND' 연산하여 비유효 연산 여부를 판단할 수 있다.

[0007] 또한, 기능 유닛은 프리디케이트(predicate)를 더 'AND' 연산하여 비유효 연산 여부를 판단할 수 있다.

[0008] 추가적인 양상에 따르면, 비유효 연산을 처리하는 장치는 데스티네이션의 제1 영역의 대응값에 기초하여 그 데스티네이션에 대한 상기 레지스터 파일로의 쓰기 요청을 처리하는 제어부를 더 포함할 수 있다.

[0009] 이때, 제어부는 데스티네이션의 제1 영역의 대응값과 쓰기 허용(Write Enable, WE) 신호를 'AND' 연산하여 그 결과에 따라 쓰기 요청을 처리할 수 있다.

[0010] 또한, 제어부는 리셋(reset) 요청에 따라 레지스터 파일 내의 제1 영역을 리셋할 수 있다.

[0011] 이때, 제어부는 리셋 요청에 따라 VLIW(Very Long Instruction Word) 모드에서 CGRA(Coarse Grained Reconfigurable Array) 모드로 전송되는 레지스터 값은 제외하고 리셋할 수 있다.

[0012] 일 양상에 따르면, 루프의 프롤로그 또는 에필로그에서 비유효 연산을 처리하는 방법은 데이터의 유효 상태를 저장하는 제1 영역과 그 데이터를 저장하는 제2 영역을 포함하는 레지스터 파일로부터 하나 이상의 입력 소스가 하나 이상의 기능 유닛(functional unit)으로 입력되는 단계, 기능 유닛이 입력 소스의 제1 영역 값에 기초하여 연산의 비유효(invalid) 여부를 판단하는 단계 및 기능 유닛이 제1 영역의 대응값을 포함하는 데스티네이션(destination)을 출력하는 단계를 포함할 수 있다.

[0013] 이때, 연산의 비유효 여부를 판단하는 단계는 입력 소스들의 제1 영역의 값들을 'AND' 연산하여 비유효 연산 여부를 판단할 수 있다.

[0014] 또한, 그 연산의 비유효 여부를 판단하는 단계는 프리디케이트(predicate)를 더 'AND' 연산하여 비유효 연산 여부를 판단할 수 있다.

[0015] 추가적인 양상에 따르면, 비유효 연산을 처리하는 방법은 제어부가 데스티네이션의 제1 영역의 대응값에 기초하여 그 데스티네이션에 대한 레지스터 파일로의 쓰기 요청을 처리할 수 있다.

[0016] 이때, 데스티네이션의 쓰기 요청을 처리하는 단계는 데스티네이션의 제1 영역의 대응값과 쓰기 허용(Write Enable, WE) 신호를 'AND' 연산하여 그 결과에 따라 그 쓰기 요청을 처리할 수 있다.

발명의 효과

[0017] CGRA 프로세서에서 데이터 폭(data width)을 일정 비트 확장하여 데이터의 유효 여부를 저장하고, 그 비트를 이용하여 비유효 연산을 처리하는 장치 및 방법을 제공할 수 있다.

[0018] 이로 인해, 프리디케이트 계산 및 라우팅으로 인한 스케쥴링의 복잡성이 감소되어 컴파일러의 성능을 향상될 수 있으며 프로세서의 ISA-level 테스트 코드 검증과 같이 프로세서 검증에 소요되는 시간을 줄일 수 있다.

도면의 간단한 설명

[0019] 도 1은 일 실시예에 따른 루프의 프롤로그 또는 에필로그의 비유효 연산을 처리하는 장치의 블록도이다.

도 2는 일 실시예에 따른 확장된 데이터와 그 데이터를 지원하기 위한 레지스터 파일의 예이다.

도 3은 프로세서의 프리디케이트를 통해 가딩하는 기능 유닛(functional unit)의 구조이다.

도 4는 일 실시예에 따른 확장된 데이터를 지원하기 위한 기능 유닛의 예이다.

도 5는 리커런스(recurrence) MII(minimum initiation interval)의 예이다.

도 6은 일 실시예에 따른 루프의 프롤로그 또는 에필로그의 비유효 연산을 처리하는 방법의 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0020] 기타 실시예들의 구체적인 사항들은 상세한 설명 및 도면들에 포함되어 있다. 본 발명의 이점 및 특징, 그리고 그것들을 달성하는 방법은 첨부되는 도면과 함께 상세하게 후술되어 있는 실시예들을 참조하면 명확해질 것이다. 그러나 본 발명은 이하에서 개시되는 실시예들에 한정되는 것이 아니라 서로 다른 다양한 형태로 구현될 수 있으며, 단지 본 실시예들은 본 발명의 개시가 완전하도록 하고, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에게 발명의 범주를 완전하게 알려주기 위해 제공되는 것이며, 본 발명은 청구항의 범주에 의해 정의될 뿐이다. 명세서 전체에 걸쳐 동일 참조 부호는 동일 구성 요소를 지칭한다.

[0021] 이하, 본 발명의 실시예들에 따른 루프의 프롤로그 또는 에필로그의 비유효 연산을 처리하는 장치 및 방법을 도면들을 참고하여 자세히 설명하도록 한다.

[0022] 도 1은 일 실시예에 따른 루프의 프롤로그 또는 에필로그의 비유효 연산을 처리하는 장치의 블록도이다. 본 실시예에 따른 비유효 연산 처리 장치(100)는 CGRA(coarse-grained reconfigurable architecture) 프로세서의 일 구성일 수 있다. 즉, CGRA 프로세서는 다수의 기능 유닛(functional unit)들로 구성될 수 있으며, 각 기능 유닛들은 내부 연결 네트워크로 연결되어 있다. 또한, 다수의 레지스터 파일과 래치(latch)를 포함할 수 있다.

[0023] CGRA 스케쥴러는 루프의 커널을 매핑(mapping)하기 위해 소프트웨어 파이프라이닝 기법을 사용한다. CGRA 스케쥴러는 높은 효율(utilization)을 얻기 위해 모듈로 스케쥴링(modulo scheduling) 기법을 이용하기 때문에 프롤로그 또는 에필로그에서는 비유효 연산(invalid operation)이 포함될 수 있다. 일반적으로, 프리디케이트를 이용하여 그 비유효 연산을 가딩(guarding)하는 방식이 이용되나, 본 실시예에서는 이하에서 자세히 설명하는 바와 같이 컴파일러의 성능 향상을 위하여 데이터의 확장 개념을 통한 가딩 방식이 제시된다.

[0024] 도 2는 일 실시예에 따른 확장된 데이터와 그 데이터를 지원하기 위한 레지스터 파일의 예로서, (a)는 데이터(data)의 유효 상태(예: 유효-true, 비유효-false)를 표현하는 유효 상태 비트(v)(예: 1 비트)와 그 데이터를 표현하는 데이터 비트(data)(예: 16, 32, 64, 128)로 이루어진 확장된 데이터의 개념을 나타낸다. (b)는 그 확장된 데이터를 지원하기 위한 레지스터 파일(110)의 구조를 나타낸 것이다.

[0025] 도 1과 도 2를 참조하여 본 실시예에 따른 비유효 연산 처리 장치(100)를 자세히 설명한다. 비유효 연산 처리 장치(100)는 레지스터 파일(110) 및 기능 유닛(120)을 포함하며, 추가적으로 제어부(130)를 더 포함할 수 있다. 레지스터 파일(110)은 도 2의 (a)에 예시된 확장된 데이터 개념을 지원하기 위해 도 2의 (b)에 도시된 바와 같이 데이터의 유효 상태를 저장하는 제1 영역(111)과 그 데이터가 저장되는 제2 영역(112)을 포함할 수 있다.

[0026] 한편, 기능 유닛(120)은 하나 이상의 입력 소스(in_src)의 제1 영역 값에 기초하여 연산(operation)의 비유효(invalid) 여부를 판단한다. 이때, 입력 소스(in_src)는 레지스터 파일(110) 또는 다른 기능 유닛(120)으로부터 내부 연결 네트워크를 통해 입력받을 수 있다. 기능 유닛(120)은 확장된 데이터 개념을 지원할 수 있다. 즉, 일반적인 프리디케이트를 통한 루프의 프롤로그 또는 에필로그의 비유효 연산을 가딩하는 방식뿐만 아니라, 하나 이상의 입력 소스들의 데이터 유효 여부가 표현된 제1 영역의 값들을 이용하여 비유효 연산 여부를 판단하고 가딩 처리할 수 있다.

[0027] 또한, 기능 유닛(120)은 판단된 연산 비유효 여부에 따라 제1 영역에 대응되는 값을 생성하고, 그 연산의 수행 결과를 포함하여 데스티네이션(out_dst)를 출력한다. 판단 결과 그 연산이 비유효한 연산이라면 제1 영역에 대응되는 값인 그 연산 결과의 유효 여부를 'false'로 설정할 수 있다. 반면에 그 연산이 유효한 연산이라면 제1 영역의 대응값은 'true'로 설정할 수 있다. 출력된 데스티네이션(out_dst)는 레지스터 파일이나 다른 기능 유닛으로 전송된다.

[0028] 제어부(130)는 레지스터 파일(110)로부터 데이터를 읽는 읽기 요청, 레지스터 파일(110)로 연산 결과를 기록하는 쓰기 요청 등을 처리한다. 제어부(130)는 읽기 요청에 따라 도 2의 (b)에 도시된 바와 같이 레지스터 파일(110)로부터 제1 영역(111)의 데이터 유효 상태 값에 상관없이 그 상태 그대로 데이터를 읽는다.

[0029] 또한, 쓰기 요청에 따라 기능 유닛(120)에서 출력된 데스티네이션(out_dst)을 레지스터 파일(110)에 기록한다. 이때, 제어부(130)는 그 데스티네이션(out_dst)의 제1 영역의 대응값 즉, 연산 결과 데이터의 유효 상태를 나타

내는 비트 값에 기초하여 쓰기 여부를 제어한다. 예컨대, 도 2에 도시된 바와 같이 제어부(130)는 데스티네이션(out_dst)의 제1 영역의 대응값을 쓰기 허용(Write Enable, WE) 신호와 'AND' 연산하여 그 결과가 'true'인 경우에는 레지스터 파일(110)에 그 데스티네이션(out_dst)을 기록한다. 이때, 제1 영역의 대응값은 레지스터 파일의 제1 영역에 기록되며, 제2 영역의 대응값은 레지스터 파일(110)의 제2 영역에 기록된다. 반면에, 그 결과가 'false'인 경우에는 레지스터 파일(110)에 기록하지 않는다.

[0030] 또한, 제어부(130)는 CGRA의 루프가 시작됨과 동시에 발생되는 리셋(reset) 요청에 따라 레지스터 파일(110) 내의 제1 영역의 데이터 유효 상태 비트를 리셋할 수 있다. 이때, VLIW(Very Long Instruction Word) 모드에서 CGRA(Coarse Grained Reconfigurable Array) 모드로 전송되는 레지스터 값은 제외하고 리셋할 수 있다.

[0031] 도 3은 프로세서의 프리디케이트를 통해 비유효 연산을 가딩하는 기능 유닛(functional unit)의 구조이다. 도 4는 일 실시예에 따른 확장된 데이터를 지원하기 위한 기능 유닛의 예이다. 도 3과 도 4를 참조하여 기능 유닛(120)에서 확장된 데이터 개념을 이용하여 비유효 연산을 가딩하는 방법을 설명한다.

[0032] 도 3을 참조하면 기능 유닛은 2개의 입력 소스(Input 1, 2)에 대해 비유효 연산을 판단하고 가딩하기 위해 별도의 프리디케이트(Predicate)를 생성하여 그 프리디케이트 정보(Predicate Input)가 라우팅 되어야 하며, 그 프리디케이트 정보를 저장하기 위한 별도의 레지스터 파일이 필요하다. 또한, 구성 메모리(Configuration Memory)의 비유효한 읽기(read)/쓰기(write) 연산 또한 가딩된다. 그러나, 도 3에 도시된 바와 같이 프리디케이트를 생성하고 그 정보를 라우팅하기 위해서는 컴파일러에서 담당하게 되므로 컴파일러의 스케쥴링 과정이 매우 복잡하게 되어 컴파일 성능이 매우 떨어질 수 있다.

[0033] 도 4를 참조하면, 기능 유닛(120)은 데이터의 유효 상태를 표현한 비트인 제1 영역의 값과 데이터가 표현된 제2 영역의 값을 포함하는 n개의 입력 소스(in_src 1 ~ in_src n)를 입력받는다. 이때, 입력 소스는 레지스터 파일(110) 또는 다른 기능 유닛(120)에서 입력받는다.

[0034] 기능 유닛(120)은 입력된 n개의 입력 소스(in_scr 1 ~ in_src n)들의 제1 영역 값들을 'AND' 연산하여 'true'인 경우에는 유효한 연산으로 판단하고, 'false'인 경우에는 비유효 연산으로 판단한다. 즉, 어느 하나의 입력 소스라도 비유효한 데이터를 포함한 경우에는 비유효한 연산으로 판단한다.

[0035] 한편, 추가적인 양상에 따르면, 프리디케이트 정보를 필요에 따라 독립적으로 생성하여 그 프리디케이트 정보와 입력 소스(in_src 1 ~ in_src2)들의 제1 영역의 데이터 유효 상태 비트와 'AND' 연산을 수행하여 비유효 연산 여부를 판단할 수 있다.

[0036] 기능 유닛(120)은 그 연산을 수행하고, 그 수행 결과 데이터와 그 비유효 연산 여부 판단 결과를 포함한 데스티네이션(out_dst)를 출력한다. 즉, 데스티네이션의(out_dst) 제1 영역에는 그 비유효 연산 여부 판단 결과 유효하면 'true', 비유효하면 'false'가 저장되며, 제2 영역에는 그 연산 수행 결과가 저장된다.

[0037] 도 5는 기능 유닛에서 리커런스(recurrence) MII(minimum initiation interval)를 나타낸 예이다. 도 5를 참조하면, 프리디케이트 라우팅으로 인한 컴파일러의 오버헤드를 알 수 있다. 즉, 프리디케이트를 통한 가딩 방식은 기능 유닛, 출력 래치, 레지스터 파일, OS 래치, 입력 래치 및 기능 유닛 순으로 수행되어, 총 사이클은 5(연산 지연(operation latency) 2 + 래치 지연(latch delay) 3)이다. 반면에, 확장된 데이터 개념을 이용한 방식은 기능 유닛, 레지스터 파일, OS 래치, 입력 래치, 기능 유닛 순으로 수행되고, 총 사이클은 3(연산 지연 1 + 래치 지연 2)이다.

[0038] 이를 통해, 컴파일러가 프리디케이트를 라우팅하는데 오버헤드가 많음을 알 수 있다. 따라서, 프리디케이트를 사용하지 않는 것에 의해 컴파일러의 성능이 크게 향상될 수 있음을 알 수 있다.

[0039] 도 6은 일 실시예에 따른 소프트웨어 파이프라인된 루프의 프롤로그 또는 에필로그의 비유효 연산을 처리하는 방법의 흐름도이다.

[0040] 도 6을 참조하여 비유효 연산을 처리하는 방법을 설명하면, 먼저, 하나 이상의 입력 소스(in_src 1 ~ in_scr n)가 기능 유닛(120)으로 입력된다(단계 310). 제어부(130)가 읽기 요청에 따라 레지스터 파일(110)로 부터 읽어 기능 유닛(120)에 입력할 수 있다. 레지스터 파일은 도 2의 (b)에 도시된 바와 같이 데이터 유효 상태 값이

저장되는 제1 영역(110)과 그 데이터 값이 저장되는 제 영역(120)을 포함할 수 있다. 제어부(130)는 읽기 요청에 따라 데이터를 읽을 때, 그 데이터의 유효 상태 즉, 제1 영역의 값은 고려하지 않고 그 상태 그대로 읽어 기능 유닛(120)에 입력할 수 있다.

[0041] 그 다음, 기능 유닛(120)이 하나 이상의 입력 소스(in_src 1 ~ in_scr n)의 제1 영역값에 기초하여 연산의 비유효 여부를 판단한다(단계 320). 즉, 기능 유닛(120)은 입력된 n개의 입력 소스(in_scr 1 ~ in_src n)들의 제1 영역 값들을 'AND' 연산하여 'true'인 경우에는 유효한 연산으로 판단하고, 'false'인 경우에는 비유효 연산으로 판단할 수 있다. 한편, 필요에 따라 프리디케이트 정보를 독립적으로 생성하여 그 프리디케이트 정보와 입력 소스(in_src 1 ~ in_src2)들의 제1 영역의 데이터 유효 상태 비트와 'AND' 연산을 수행하여 비유효 연산 여부를 판단할 수 있다.

[0042] 그 다음, 기능 유닛(120)은 그 연산을 수행하고, 그 수행 결과 데이터와 그 비유효 연산 여부 판단 결과를 포함한 데스티네이션(out_dst)을 출력한다(단계 330). 기능 유닛(120)은 데스티네이션의(out_dst) 제1 영역의 대응값으로 그 비유효 연산 여부 판단 결과 유효하면 'true', 비유효하면 'false'를 생성하며, 제2 영역에는 그 연산 수행 결과를 포함한다.

[0043] 마지막으로, 제어부(130)는 그 데스티네이션(out_dst)의 제1 영역의 대응값에 기초하여 레지스터 파일로 그 데스티네이션(out_dst)을 기록할지를 결정하여 그 결정에 따라 쓰기 요청을 처리할 수 있다(단계 340). 예컨대, 도 2를 참조하면 제어부(130)는 데스티네이션(out_dst)의 제1 영역의 대응값을 쓰기 허용(Write Enable, WE) 신호와 'AND' 연산하여 그 결과가 'true'인 경우에는 레지스터 파일(110)에 그 데스티네이션(out_dst)을 기록한다. 이때, 제1 영역의 대응값을 레지스터 파일의 제1 영역에 기록되며, 제2 영역의 대응값은 레지스터 파일(110)의 제2 영역에 기록된다. 반면에, 그 결과가 'false'인 경우에는 레지스터 파일(110)에 기록하지 않는다.

[0044] 한편, 본 발명의 실시 예들은 컴퓨터로 읽을 수 있는 기록 매체에 컴퓨터가 읽을 수 있는 코드로 구현하는 것이 가능하다. 컴퓨터가 읽을 수 있는 기록 매체는 컴퓨터 시스템에 의하여 읽혀질 수 있는 데이터가 저장되는 모든 종류의 기록 장치를 포함한다.

[0045] 컴퓨터가 읽을 수 있는 기록 매체의 예로는 ROM, RAM, CD-ROM, 자기 테이프, 플로피디스크, 광 데이터 저장장치 등이 있으며, 또한 캐리어 웨이브(예를 들어 인터넷을 통한 전송)의 형태로 구현하는 것을 포함한다. 또한, 컴퓨터가 읽을 수 있는 기록 매체는 네트워크로 연결된 컴퓨터 시스템에 분산되어, 분산 방식으로 컴퓨터가 읽을 수 있는 코드가 저장되고 실행될 수 있다. 그리고 본 발명을 구현하기 위한 기능적인(functional) 프로그램, 코드 및 코드 세그먼트들은 본 발명이 속하는 기술 분야의 프로그래머들에 의하여 용이하게 추론될 수 있다.

[0046] 본 발명이 속하는 기술분야의 통상의 지식을 가진 자는 본 발명이 그 기술적 사상이나 필수적인 특징을 변경하지 않고서 다른 구체적인 형태로 실시될 수 있다는 것을 이해할 수 있을 것이다. 그러므로 이상에서 기술한 실시예들은 모든 면에서 예시적인 것이며 한정적이 아닌 것으로 이해해야만 한다. 본 발명의 범위는 상기 상세한 설명보다는 후술하는 특허청구의 범위에 의하여 나타내어지며, 특허청구의 범위의 의미 및 범위 그리고 그 균등 개념으로부터 도출되는 모든 변경 또는 변형된 형태가 본 발명의 범위에 포함되는 것으로 해석되어야 한다.

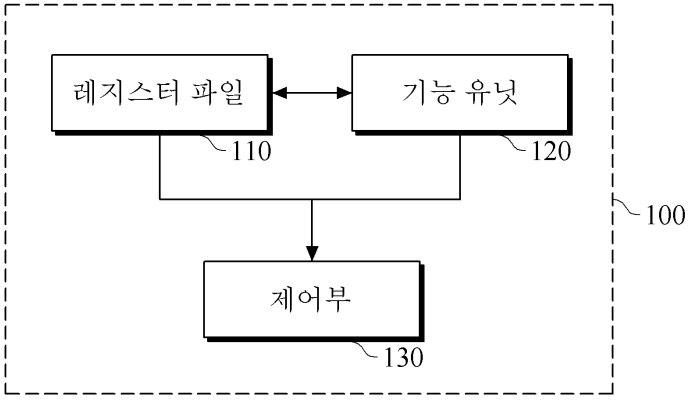
부호의 설명

[0047]

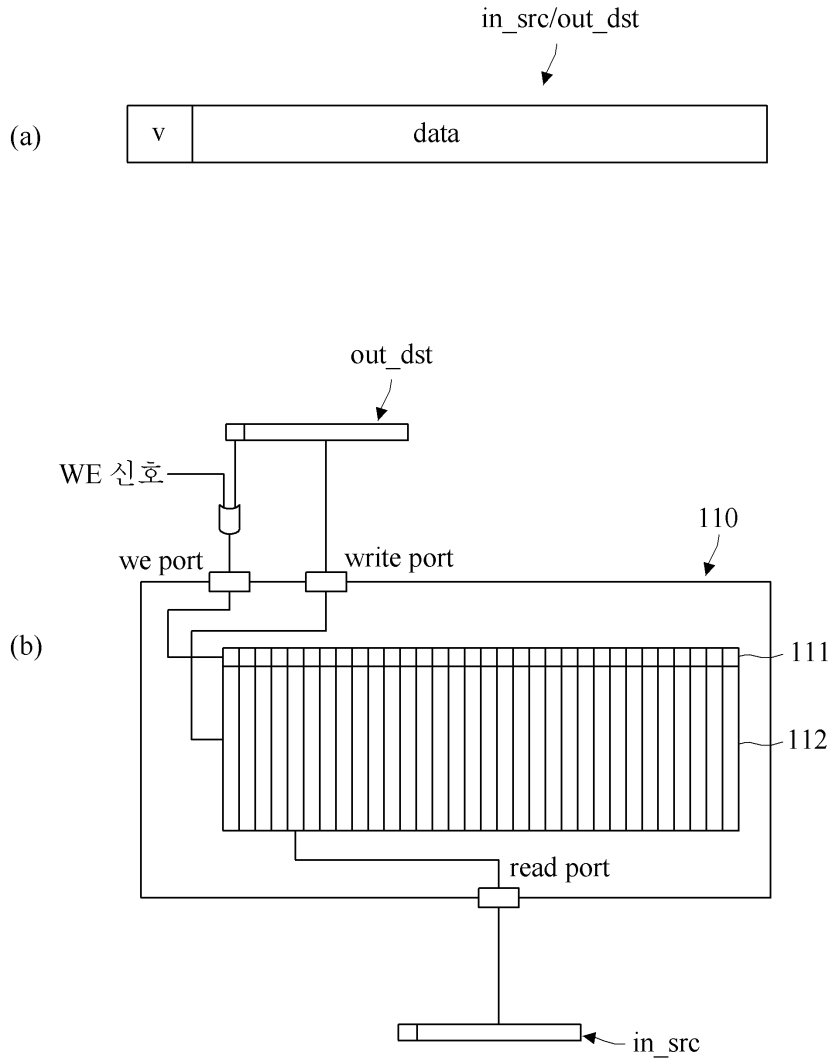
100: 비유효 연산 처리 장치	110: 레지스터 파일
111: 제1 영역	112: 제2 영역
120: 기능 유닛	130: 제어부

도면

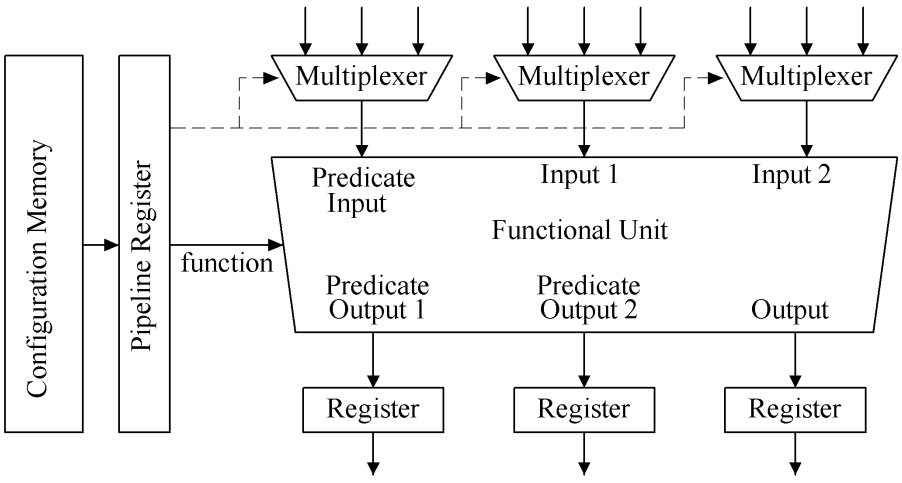
도면1



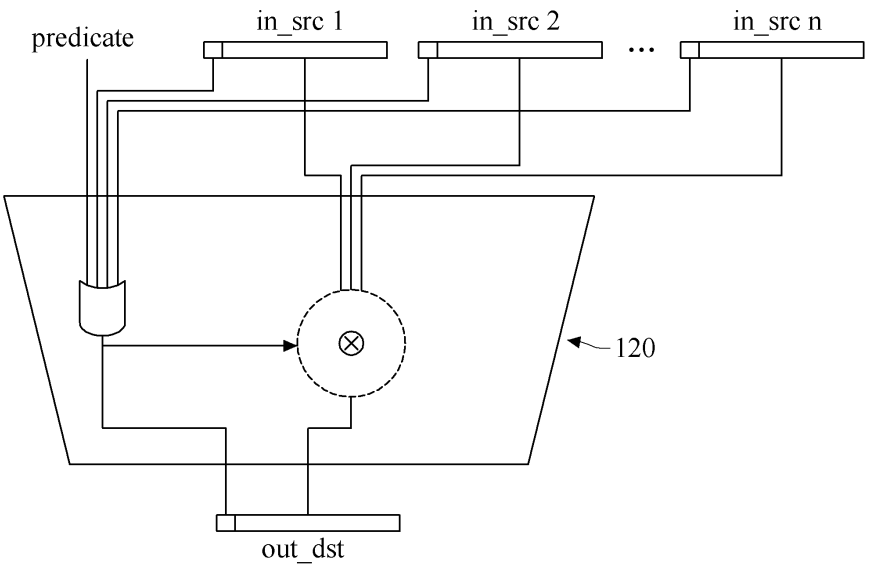
도면2


in_src/out_dst
(a)
v
data
out_dst
WE 신호
we port
write port
110
(b)
111
112
read port
in_src

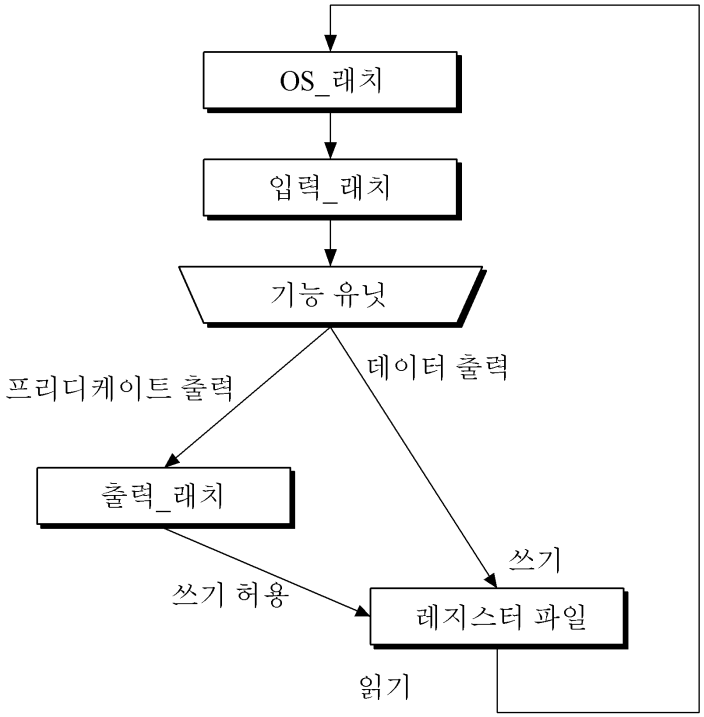
도면3


Configuration Memory
Pipeline Register
function
Multiplexer
Multiplexer
Multiplexer
Predicate Input
Input 1
Input 2
Functional Unit
Predicate Output 1
Predicate Output 2
Output
Register
Register
Register

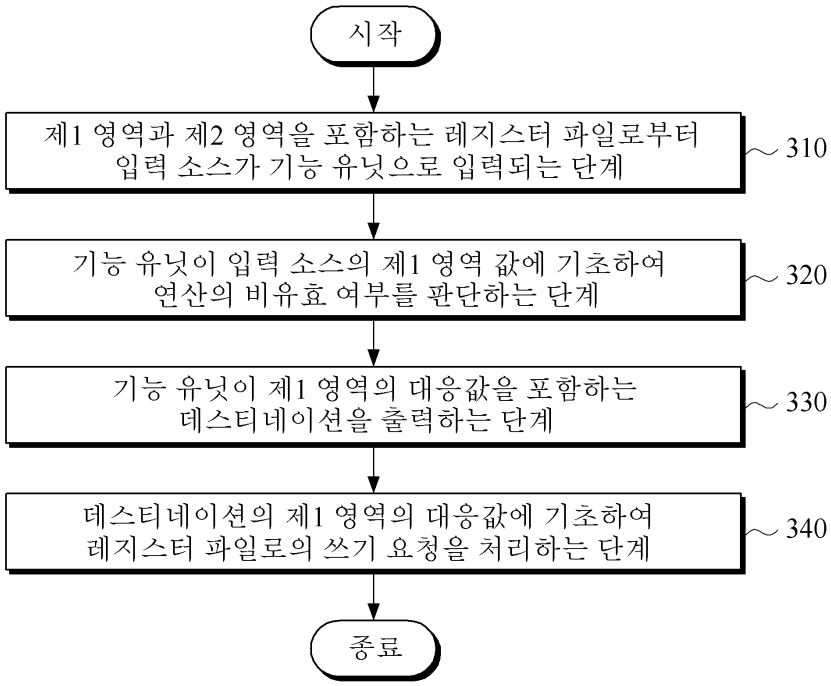
도면4


predicate
in_src 1
in_src 2
...
in_src n
120
out_dst

도면5


OS_래치
입력_래치
기능 유닛
프리디케이트 출력
데이터 출력
출력_래치
쓰기
쓰기 허용
레지스터 파일
읽기

도면6



【심사관 직권보정사항】

【직권보정 1】

【보정항목】 청구범위

【보정세부항목】 제9항

【변경전】

제1항에 있어서

【변경후】

제8항에 있어서

【직권보정 2】

【보정항목】 청구범위

【보정세부항목】 제8항

【변경전】

로프의 시작과 동시에 발생되는

【변경후】

루프의 시작과 동시에 발생되는