(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2015년09월16일
(11) 등록번호 10-1553652
(24) 등록일자 2015년09월10일

(51) 국제특허분류(Int. Cl.)
G06F 9/06 (2006.01) *G06F 9/30* (2006.01)
G06F 9/44 (2006.01) *G06F 9/46* (2006.01)
(21) 출원번호 10-2009-0013529
(22) 출원일자 2009년02월18일
심사청구일자 2014년01월06일
(65) 공개번호 10-2010-0094211
(43) 공개일자 2010년08월26일
(56) 선행기술조사문헌
KR1020050037575 A*
JP2006505055 A*
KR1020050030014 A
KR1020030067892 A
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
삼성전자 주식회사
경기도 수원시 영통구 삼성로 129 (매탄동)
(72) 발명자
이강웅
서울특별시 관악구 인헌길 160, 302호 (봉천동)
류수정
충청남도 천안시 동남구 목천읍 동리4길 110-9
(뒷면에 계속)
(74) 대리인
특허법인 신지, 유경열, 천성훈

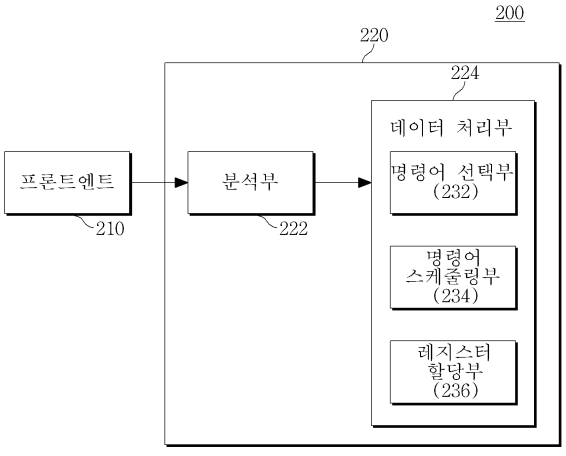
전체 청구항 수 : 총 16 항 심사관 : 박승철

(54) 발명의 명칭 **이종 프로세서에 대한 명령어 컴파일링 장치 및 방법**

(57) 요 약

혼합된 폭의 데이터 경로(mixed-width data paths)를 가지는 이종의(heterogeneous) 프로세서에서 이용하기 위한 명령어 컴파일링 장치 및 방법이 개시된다. 이종 프로세서는 서로 다른 데이터 폭을 지원하는 이종의 구성요소를 포함한다. 이종의 구성요소는 서로 다른 데이터 폭의 데이터를 처리하는 복수 개의 연산 유닛, 서로 다른 데이터 폭의 데이터를 저장하는 복수 개의 레지스터 파일 및 서로 다른 데이터 폭을 가지는 연결 와이어를 포함할 수 있다. 소스 코드를 중간 코드로 생성할 때 생성되는 오퍼랜드의 타입 정보를 이용하여 데이터 플로우 그래프상의 각 오퍼랜드 및 각 에지에 대한 데이터 폭을 결정하고, 결정된 데이터 폭을 이용하여 명령어 컴파일링이 수행된다.

대 표 도 - 도2



(72) 발명자

유동훈

서울특별시 강동구 성안로 182, 101동 302호 (길동, 노블레스)

박일현

경기도 용인시 기흥구 삼성2로 97, 기숙사 A동 520호 (농서동, 삼성종합기술원)

명 세 서

청구범위

청구항 1

오퍼랜드의 타입 정보를 이용하여 데이터 플로우 그래프상의 각 노드의 입출력 오퍼랜드 및 각 에지에 대한 데이터 폭을 결정하는 분석부; 및

상기 각 노드의 입출력 오퍼랜드 및 각 에지에 대한 데이터 폭에 기초하여 서로 다른 데이터 폭을 지원하는 이종의 구성요소를 포함하는 프로세서에서 수행될 명령어를 제공하는 데이터 처리부를 포함하는 컴파일링 장치.

청구항 2

제1항에 있어서,

상기 이종의 구성요소는 서로 다른 데이터 폭의 데이터를 처리하는 복수 개의 연산 유닛, 서로 다른 데이터 폭의 데이터를 저장하는 복수 개의 레지스터 파일 및 서로 다른 데이터 폭을 가지는 연결 와이어 중 적어도 하나를 포함하는 컴파일링 장치.

청구항 3

제1항에 있어서,

상기 분석부는 상기 오퍼랜드의 타입 정보에 기초하여 상기 데이터 플로우 그래프상에 입출력 노드들의 데이터 폭을 초기화하고, 고정점 알고리즘을 이용하여 미지의 오퍼랜드 및 에지에 대한 데이터 폭을 결정하는 컴파일링 장치.

청구항 4

제1항에 있어서,

상기 데이터 처리부는 상기 데이터 플로우 그래프상의 각 노드의 입출력 오퍼랜드 및 각 에지에 대해 결정된 데이터 폭에 기초하여 명령어를 선택하는 컴파일링 장치.

청구항 5

제4항에 있어서,

상기 데이터 처리부는 상기 선택된 명령어를 수행할 수 있는 연산 유닛을 결정하는 컴파일링 장치.

청구항 6

제4항에 있어서,

상기 데이터 처리부는 상기 데이터 플로우 그래프상의 각 노드의 입출력 오퍼랜드 및 각 에지에 대해 결정된 데이터 폭에 기초하여 레지스터를 할당하는 컴파일링 장치.

청구항 7

제4항에 있어서,

상기 프로세서가 CGA 프로세서인 경우, 상기 데이터 처리부는 상기 데이터 플로우 그래프상의 각 노드의 입출력 오퍼랜드 및 각 에지에 대해 결정된 데이터 폭에 기초하여 상기 선택된 명령어를 수행하기 위한 CGA 프로세서의 라우팅 경로상의 노드에 대한 입출력 오퍼랜드 및 에지의 데이터 폭을 결정하는 컴파일링 장치.

청구항 8

제7항에 있어서,

상기 프로세서는 VLIW 프로세서 또는 CGA 프로세서인 컴파일링 장치.

청구항 9

오퍼랜드의 타입 정보를 이용하여 데이터 플로우 그래프상의 각 노드의 입출력 오퍼랜드 및 각 에지에 대한 데이터 폭을 결정하는 단계; 및

상기 각 노드의 입출력 오퍼랜드 및 각 에지에 대한 데이터 폭에 기초하여 서로 다른 데이터 폭을 지원하는 이종의 구성요소를 포함하는 이종 프로세서에서 수행될 명령어를 제공하는 단계를 포함하는 컴파일링 방법.

청구항 10

제9항에 있어서,

상기 이종의 구성요소는 서로 다른 데이터 폭의 데이터를 처리하는 복수 개의 연산 유닛, 서로 다른 데이터 폭의 데이터를 저장하는 복수 개의 레지스터 파일 및 서로 다른 데이터 폭을 가지는 연결 와이어 중 적어도 하나를 포함하는 컴파일링 방법.

청구항 11

제9항에 있어서,

데이터 플로우 그래프상의 각 노드의 입출력 오퍼랜드 및 각 에지에 대한 데이터 폭을 결정하는 단계는,

상기 오퍼랜드의 타입 정보에 기초하여 상기 데이터 플로우 그래프상에 입출력 노드에 대한 데이터 폭을 초기화하는 단계; 및

고정점 알고리즘을 이용하여 미지의 오퍼랜드 및 에지에 대한 데이터 폭을 결정하는 단계를 포함하는 컴파일링 방법.

청구항 12

제9항에 있어서,

이종 프로세서에서 수행될 명령어를 제공하는 단계는, 상기 데이터 플로우 그래프상의 각 노드의 오퍼랜드 및 각 에지에 대해 결정된 데이터 폭에 기초하여 명령어를 선택하는 단계를 더 포함하는 컴파일링 방법.

청구항 13

제12항에 있어서,

이종 프로세서에서 수행될 명령어를 제공하는 단계는, 상기 선택된 명령어를 수행할 수 있는 연산 유닛을 결정하는 단계를 더 포함하는 컴파일링 방법.

청구항 14

제12항에 있어서,

이종 프로세서에서 수행될 명령어를 제공하는 단계는, 상기 데이터 플로우 그래프상의 각 노드의 입출력 오퍼랜드 및 각 에지에 대해 결정된 데이터 폭에 기초하여 레지스터를 할당하는 단계를 더 포함하는 컴파일링 방법.

청구항 15

제12항에 있어서,

이종 프로세서에서 수행될 명령어를 제공하는 단계는,

상기 프로세서가 CGA 프로세서인 경우 상기 데이터 플로우 그래프상의 각 노드의 입출력 오퍼랜드 및 각 에지에 대해 결정된 데이터 폭에 기초하여 상기 선택된 명령어를 수행하기 위한 라우팅 경로상의 노드에 대한 입출력 오퍼랜드 및 에지의 데이터 폭을 결정하는 단계를 더 포함하는 컴파일링 방법.

청구항 16

제9항에 있어서,

상기 프로세서는 VLIW 프로세서 또는 CGA 프로세서인 컴파일링 방법.

발명의 설명

발명의 상세한 설명

기 술 분 야

[0001] 본 발명의 하나 이상의 양상은 데이터 처리 시스템에 관한 것으로, 상세하게는 프로세서에서 실행될 코드를 처리하기 위한 컴파일링 장치 및 방법에 관한 것이다.

배 경 기 술

[0002] 컴파일러는 특정 프로그램 언어로 작성된 문장을 처리하여 기계어 또는 컴퓨터가 사용할 수 있는 코드로 변경시켜준다. C나 파스칼과 같은 언어로 프로그램을 개발할 경우, 프로그래머는 편집기를 이용하여 한줄 한줄 문장을 작성하게 되는데, 이러한 파일들을 소스 코드라고 부른다. 소스 코드의 작성이 끝나면 프로그래머는 그 소스 코드의 언어에 맞는 컴파일러를 실행시킨다.

[0003] 컴파일러는 실행시에 모든 문장을 먼저 구문적으로 하나씩 분해하고, 다른 문장을 참조하는 경우 문장이 정확하게 참조될 수 있도록 여러 번의 연속적인 상태에서 결과 코드를 만든다. 컴파일로 생긴 결과물은 목적 코드 또는 목적 모듈이라 불리는데, 목적 코드는 프로세서가 한 번에 한 명령씩 처리하거나 또는 실행시킬 수 있는 기계 코드이다.

발명의 내용

해결 하고자하는 과제

[0004] 혼합된 폭의 데이터 경로(mixed-width data paths)를 가지는 이종의(heterogeneous) 프로세서에서 이용하기 위한 명령어 컴파일링 장치 및 방법이 제안된다.

과제 해결수단

[0005] 일 양상에 따른 컴파일링 장치는 오퍼랜드의 타입 정보를 이용하여 데이터 플로우 그래프상의 각 오퍼랜드 및 각 에지에 대한 데이터 폭을 결정하는 분석부; 및 각 오퍼랜드 및 각 에지에 대한 데이터 폭에 기초하여 서로 다른 데이터 폭을 지원하는 이종의 구성요소를 포함하는 프로세서에서 수행될 명령어를 제공하는 데이터 처리부를 포함한다.

[0006] 이종의 구성요소는 서로 다른 데이터 폭의 데이터를 처리하는 복수 개의 연산 유닛, 서로 다른 데이터 폭의 데이터를 저장하는 복수 개의 레지스터 파일 및 서로 다른 데이터 폭을 가지는 연결 와이어 중 적어도 하나를 포함할 수 있다.

[0007] 분석부는 오퍼랜드의 타입 정보에 기초하여 데이터 플로우 그래프상에 입출력 노드들에 대한 데이터 폭을 초기화하고, 고정점 알고리즘(fixed point algorithm)을 이용하여 미지의 오퍼랜드 및 에지에 대한 데이터 폭을 결정할 수 있다.

[0008] 데이터 처리부는 데이터 플로우 그래프상의 각 오퍼랜드 및 각 에지에 대해 결정된 데이터 폭에 기초하여 명령어를 선택할 수 있다. 데이터 처리부는 선택된 명령어를 수행할 수 있는 연산 유닛을 결정할 수 있다. 데이터 처리부는 데이터 플로우 그래프상의 각 오퍼랜드 및 각 에지에 대해 결정된 데이터 폭에 기초하여 레지스터를 할당할 수 있다.

[0009] 프로세서가 CGA 프로세서인 경우 데이터 처리부는 데이터 플로우 그래프상의 각 오퍼랜드 및 각 에지에 대해 결정된 데이터 폭에 기초하여 선택된 명령어를 수행하기 위한 라우팅 경로상의 노드에 대한 입출력 오퍼랜드 및 에지의 데이터 폭을 결정할 수 있다.

[0010] 여기에서, 프로세서는 VLIW 프로세서 또는 CGA 프로세서일 수 있다.

[0011] 다른 양상에 따른 컴파일링 방법은 오퍼랜드의 타입 정보를 이용하여 데이터 플로우 그래프상의 각 오퍼랜드 및 각 에지에 대한 데이터 폭을 결정하는 단계; 및 각 오퍼랜드 및 각 에지에 대한 데이터 폭에 기초하여 서로 다른 데이터 폭을 지원하는 이종의 구성요소를 포함하는 이종 프로세서에서 수행될 명령어를 제공하는 단계를 포

함한다.

효 과

[0012] 일 실시예에 따르면, 칩영역 및 전력소모가 감소되는 이종의 연산 처리를 수행할 수 있는 프로세서에서 이용될 수 있는 소스 코드를 데이터 폭(data width)을 고려하여 효율적으로 컴파일링하는 장치 및 방법을 제공할 수 있다.

발명의 실시를 위한 구체적인 내용

[0013] 이하, 첨부된 도면을 참조하여 본 발명의 일 실시예를 상세하게 설명한다. 본 발명의 일 실시예를 설명함에 있어 관련된 공지 기능 또는 구성에 대한 구체적인 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명을 생략할 것이다. 또한, 후술되는 용어들은 본 발명의 실시예들에서의 기능을 고려하여 정의된 용어들로서 이는 사용자, 운용자의 의도 또는 관례 등에 따라 달라질 수 있다. 그러므로 그 정의는 본 명세서 전반에 걸친 내용을 토대로 내려져야 할 것이다.

[0014] 도 1은 일 실시예에 따른 이종 프로세서(heterogeneous processor)의 구성을 나타내는 블록도이다.

[0015] 프로세서(100)는 혼합된 폭의 데이터 경로(mixed-width data paths)를 가진 서로 다른 폭을 가지는 데이터를 처리하도록 하기 위한 이종의 구성요소를 포함한다. 프로세서(100)는 서로 다른 폭을 가지는 데이터를 처리하도록 하기 위한 이종의 구성요소를 가지는 한 VLIW(Very Long Instruction Word) 프로세서, CGA(Coarse Grained Array) 프로세서 또는 RISC(Reduced Instruction Set Computer) 등으로 구현될 수 있다. 여기에서, 이종의 구성요소는 서로 다른 데이터 폭의 데이터를 처리하는 복수 개의 연산 유닛, 서로 다른 데이터 폭의 데이터를 저장하는 복수 개의 레지스터 파일 및 서로 다른 데이터 폭을 가지는 연결 와이어 중 적어도 하나를 포함할 수 있다.

[0016] 일 실시예에 따른 프로세서(100)는 서로 다른 데이터 폭을 가지는 데이터를 처리할 있는 이종의 복수 개의 연산 유닛(130 내지 135, 150 내지 155), 레지스터 파일(110, 120, 140 내지 143, 160 내지 161) 및 연결 와이어를 포함하는 프로세서의 간략한 구조를 나타내며, 구현예에 따라 다른 구성요소를 더 포함할 수 있다. 도 1에서 굵은 실선은 64비트 데이터 연결선을 나타내고, 얇은 실선은 32비트 데이터 연결선을 나타낸다.

[0017] 연산 유닛(130 내지 135, 150 내지 155)은 연산을 수행하는 유닛이다. 연산 유닛(130 내지 135)는 64비트 데이터 폭의 데이터를 입력받아 연산을 수행하고 결과를 출력하는 64비트 연산 유닛이다. 연산 유닛(150 내지 155)는 32비트 데이터 폭의 데이터를 입력받아 연산을 수행하고 결과를 출력하는 32비트 연산 유닛이다.

[0018] 각 연산 유닛(130 내지 135, 150 내지 155)은 각기 다른 소스(source)로부터 데이터를 받을 수 있으며, 다른 목적지(destination)로 처리 결과를 전달할 수 있다. 도 1에서는 64비트의 데이터 폭을 가지는 데이터를 처리할 수 있는 연산 유닛(130 내지 135) 및 32 비트의 데이터 폭을 가지는 데이터를 처리할 수 있는 연산 유닛(150 내지 155)을 도시하고 있으나, 여기에서 32 비트, 64 비트의 데이터 폭은 예시적인 것이며 이에 제한되지 않는다.

[0019] 레지스터 파일(110, 120, 140 내지 143, 160, 161)은 레지스터들의 집합으로, 연산 유닛(130)에서 사용되는 데이터를 임시 저장한다. 레지스터 파일(110)은 64비트 중앙 레지스터 파일이고, 레지스터 파일(120)은 32비트 중앙 레지스터 파일이다. 레지스터 파일(140 내지 143)은 분산된 64비트 레지스터 파일이고, 레지스터 파일(160, 161)은 분산된 32비트 레지스터 파일이다.

[0020] 통상의 프로세서에서 연산 유닛, 레지스터 파일 및 연결 와이어 등의 구성요소는 동일한 데이터 비트 폭을 가진다. 따라서, 예를 들어 32 비트 이하의 데이터들의 처리에 32비트의 연산 유닛이 요구되는 경우에도, 동종의 64비트 구조의 연산 유닛을 포함하는 프로세서를 이용하여 데이터를 처리하게 된다. 그러나, 큰 폭의 데이터 처리가 가능한 연산 유닛들이 작은 폭의 데이터를 처리하는 것은 반도체 크기 및 에너지 효율면에서 효율적이지 못하다. 이에 비해, 도 1에 도시된 바와 같이, 이종의 연산 유닛들, 이종의 레지스터 파일 이종의 데이터 연결선, 또는 먹스(multiplexer)를 가지는 혼합 폭 구조의 이종 프로세서를 이용하면 유사한 성능을 유지하면서도 다이(die) 영역이 적게 요구되고 에너지도 적게 소모할 수 있다.

[0021] 이하에서는 도 1에 도시된 바와 같은 이종의 프로세서에서 이용하기 위한 명령어 컴파일링 장치 및 방법에 대하여 상세하게 설명한다.

[0022] 도 2는 도 1의 이종 프로세서에 이용하기 위한 일 실시예에 따른 명령어 컴파일링 장치의 구성을 나타내는 블록

도이다.

[0023] 컴파일링 장치(200)는 크게 프론트엔드(210) 및 백엔드(220)로 이루어진다.

[0024] 프론트엔드(210)는 소스 코드를 읽어 중간 코드 형태로 변경해준다. 중간 코드(Intermediate Code)는 컴파일러가 소스 코드를 읽어서 이를 파싱하여 최적화하기 좋은 형태로 만든 코드를 의미한다. 최적화 단계가 모두 끝나면 중간 코드는 어셈블리 코드로 최종 변경된다.

[0025] 백엔드(220)는 중간 코드를 입력으로 받아서 프로그램 성능을 향상시킬 수 있는 다양한 최적화 작업을 수행한 후 어셈블리 코드를 출력한다. 백엔드(220)는 분석부(222) 및 데이터 처리부(224)를 포함할 수 있다.

[0026] 분석부(222)는 여러가지 최적화 방법을 수행하기 위하여 소스 코드들에 대한 중간 코드들을 분석한다.

[0027] 분석부(222)는 재구성가능한 어레이 상에 매핑될 오퍼레이션들과 그 오퍼레이션들 간의 데이터 의존성(dependency)을 보여주는 데이터 플로우 그래프(data flow graph)를 생성한다. 일 실시예에 따르면, 데이터 플로우 그래프에서는 입력(input), 각 오퍼레이션들(operations) 및 출력(output)이 노드(node)로, 데이터 흐름(data flow)이 에지(edge)로 표현된다.

[0028] 분석부(222)는 소스 코드를 중간 코드로 생성할 때 생성되는 오퍼랜드의 타입 정보(type information)를 이용하여 데이터 플로우 그래프상의 각 노드에 대한 입출력 오퍼랜드 및 에지에 대한 데이터 폭을 결정한다. 오퍼랜드의 타입 정보는 오퍼랜드 즉, 변수(variable value), 상수(constant value), 캐릭터(character) 등이 몇 비트의 데이터 폭을 가지는지를 나타내는 값을 의미한다.

[0029] 분석부(222)는 오퍼랜드의 타입 정보에 기초하여 데이터 플로우 그래프상에 입출력 노드들에 대한 데이터 폭을 초기화한다. 초기 설정된 값에 대하여 고정점 알고리즘(fixed point algorithm or fixed point iteration)을 이용하여 결과 값 즉, 미지의(unknown) 오퍼랜드 및 에지에 대한 데이터 폭에 변화가 없을 때까지 고정점 알고리즘을 반복 수행한다. 데이터 폭이 알려지지 않는 오퍼랜드 및 에지에 대한 데이터 폭은 가능한 데이터 폭 중 최소의 비트 폭(minimal bit-width)으로 결정된다. 이러한 과정을 통해, 데이터 플로우 그래프상의 모든 오퍼랜드 및 에지에 대한 데이터 폭이 결정된다.

[0030] 이와 같이 결정된 데이터 플로우 그래프 상의 각 오퍼랜드 및 각 에지에 대한 데이터 폭에 대한 정보는 데이터 플로우 그래프의 각 노드 및 에지에 부가된다. 부가된 각 오퍼랜드 및 각 에지에 대한 데이터 폭에 대한 정보는 이후 명령어 선택, 명령어 스케줄링 및 레지스터 할당 과정에서 이용된다.

[0031] 데이터 처리부(224)는 각 노드에 대한 입출력 오퍼랜드 및 에지에 대한 데이터 폭을 이용하여 이종 프로세서에서 수행될 명령어를 제공한다. 데이터 처리부(224)는 이종 프로세서에서 수행될 명령어를 제공하기 위하여 명령어를 선택하고, 오퍼레이션들을 연산 유닛(Function Unit)에 배치(placement) 또는 매핑(mapping)한다. 도 2를 참조하면, 데이터 처리부(224)는 명령어 선택부(232), 명령어 스케줄링부(234) 및 레지스터 할당부(236)를 포함한다.

[0032] 명령어 선택부(232)는 데이터 플로우 그래프상의 각 오퍼랜드 및 각 에지에 대해 결정된 데이터 폭에 기초하여 명령어(instruction)를 선택한다. 선택될 명령어는 명령어 세트 형태로 소정의 저장 공간에 미리 저장되어 이용될 수 있다. 예를 들어, 합산(add)을 수행하는 어떤 노드의 입력 오퍼랜드의 데이터 폭이 30비트이고, 출력 오퍼랜드의 데이터 폭이 32비트인 경우, 32비트 add 명령어가 선택된다.

[0033] 명령어 스케줄링부(234)는 프로세서에서 어떤 연산 유닛이 어떤 선택된 명령어를 수행할지 결정한다. 레지스터 할당부(236)는 데이터 플로우 그래프상의 각 오퍼랜드 및 각 에지에 대해 결정된 데이터 폭에 기초하여 요구되는 레지스터를 결정한다.

[0034] 프로세서가 CGA인 경우 데이터 플로우 상에서 각각 오퍼레이션에 대응하는 2개의 노드가 하나의 에지로 연결되더라도, 하나의 노드가 매핑되는 연산 유닛과 다른 노드가 매핑되는 연산 유닛이 떨어져 있을 수 있다. 이 경우, 명령어 스케줄링부(234)는 데이터 플로우 그래프상의 각 오퍼랜드 및 각 에지에 대해 결정된 데이터 폭에 기초하여 선택된 명령어를 수행하기 위한 CGA의 라우팅 경로상의 입출력 오퍼랜드 및 에지에 대한 데이터 폭을 결정한다. 즉, 데이터 전달을 위해 노드들 사이의 라우팅 경로상의 노드 예를 들어, 연산 유닛 또는 레지스터 파일을 결정하고, 결정된 노드에 대한 입출력 오퍼랜드 및 에지(즉, 데이터 연결선)의 데이터 폭을 결정한다.

[0035] 도 3은 일 실시예에 따른 데이터 플로우 그래프의 일 예를 나타내는 도면이다.

[0036] 도 3에서, 각 노드(301 내지 309)는 입력(변수 또는 상수)(301, 302, 303, 306), 각 오퍼레이션들(304, 305,

307, 308) 및 출력 변수(309)를 나타낸다.

[0037] 각 노드(301 내지 309)의 입력(in), 입력 상수(const), 오퍼레이션(op) 및 출력(out)을 나타내는 문자(character) 위의 숫자(또는 숫자들)은 노드에 입력되는 오퍼랜드의 데이터 폭을 나타내고, 문자 아래의 숫자는 노드로부터 출력되는 오퍼랜드의 데이터 폭을 나타낸다. 소스 코드가 전처리 과정을 거치게 되면, 입출력 노드의 타입 정보를 알 수 있다. 즉, 입력 노드(301, 302, 303, 306) 및 출력 노드(309)의 오퍼랜드의 데이터 폭이 결정된다.

[0038] 그러면 입력 노드(301, 302, 303, 306) 및 출력 노드(309) 사이의 오퍼레이션 노드들(304, 305, 307, 308)에서의 입출력 오퍼랜드의 데이터 폭이 고정점 알고리즘에 의해서 결정될 수 있다. 그러면, 오퍼레이션 노드들(304, 305, 307, 308)에 대하여 어떤 명령어가 사용될지가 결정될 수 있다.

[0039] 예를 들어, 노드(304, 305, 308)의 경우에는 64비트의 데이터 폭을 가지는 데이터가 입력되고 출력되므로, 64비트 명령어가 선택된다. 노드(307)의 경우에는 32 비트의 데이터 폭을 가지는 데이터가 입력되고 출력되므로, 32비트 명령어가 선택된다.

[0040] 명령어가 선택되면, 각 명령어를 수행할 연산 유닛상에 각 명령어가 매핑된다. 도 1을 참조하면, 예를 들어, 오퍼레이션(305)이 연산 유닛(131)에 매핑되고, 오퍼레이션(307)이 연산 유닛(150)에 매핑되고, 오퍼레이션(308)이 연산 유닛(130)에 매핑되도록 명령어 스케줄링이 수행될 수 있다.

[0041] 도 4는 도 3의 데이터 플로우 그래프를 레지스터를 고려하여 변형한 데이터 플로우 그래프를 나타내는 도면이다.

[0042] 도 3의 데이터 플로우 그래프에서 레지스터를 표시하기 위하여 도 3의 데이터 플로우 그래프를 변형하면 도 4에 도시된 바와 같은 그래프가 된다. 도 4의 데이터 플로우 그래프에서 노드(401)는 입력 변수 노드(301)가 저장되는 레지스터에 대응하고, 노드(402)는 입력 변수 노드(302)가 저장되는 레지스터에 대응하고, 노드(404)는 출력 변수 노드(309)가 저장되는 레지스터에 대응한다. 노드(403)는 노드(308)에서 노드(304) 사이에 데이터를 저장하는데 필요한 레지스터를 나타낸다.

[0043] 이와 같이, 데이터 플로우 그래프가 변형되면, 노드(401), 노드(402) 및 노드(404)에 32비트 레지스터가 요구되며, 노드(403)에는 64비트 레지스터가 요구됨이 결정될 수 있다.

[0044] 도 5는 일 실시예에 따른 라우팅을 고려하여 생성된 데이터 플로우 그래프의 일 예를 나타내는 도면이다.

[0045] CGA는 VLIW와 같은 코어의 제어에 따라서 통상적으로 반복 수행을 요구하는 작업 예를 들어, 반복적으로 수행되는 루프(loop) 작업과 같이 작업 수행시 발생되는 데이터 처리량이 많은 작업을 처리한다. CGA는 통상 복수 개의 프로세싱 유닛을 포함하며, CGA는 애플리케이션내에 존재하는 오퍼레이션들(operations) 간의 병렬성(ILP, Instruction Level Parallelism)을 최대한 이용함으로써 성능 향상을 꾀한다. 즉, 동시에 처리될 수 있는 오퍼레이션들을 CGA를 구성하는 다수 개의 연산 유닛에 분산시켜 한 번에 처리되도록 함으로써 애플리케이션의 수행 시간을 단축하는 것이다. CGA는 연산 유닛들 간의 연결이 성기기(sparse) 때문에, 스케줄러가 오퍼레이션 배치뿐만 아니라 오퍼레이션 사이에서의 오퍼랜드 라우팅도 고려하여 스케줄링을 해주어야 한다.

[0046] 이 경우, 명령어 스케줄링부(234)는 데이터 플로우 그래프상의 각 오퍼랜드 및 각 에지에 대해 결정된 데이터 폭에 기초하여 선택된 명령어를 수행하기 위한 CGA의 라우팅 경로상의 입출력 오퍼랜드 및 에지에 대한 데이터 폭을 결정한다. 즉, 데이터 전달을 위해 노드들 사이의 라우팅 경로상의 노드 예를 들어, 연산 유닛 또는 레지스터 파일을 결정하고, 결정된 노드의 입출력 오퍼랜드 및 에지(즉, 데이터 연결선)의 데이터 폭을 결정한다.

[0047] 도 3에 도시된 데이터 플로우 그래프상의 노드들이 CGA에서 수행되는 경우, 노드(305) 및 노드(308)를 수행하는 연산 유닛이 CGA에서 서로 떨어져 있는 경우에는 노드(305)의 수행 결과를 노드(308)로 전달해주기 위한 라우팅 경로를 생성하여야 한다. 노드들(501, 502, 503)은 노드(305)의 수행 결과를 노드(308)로 전달해주기 위한 라우팅 경로상의 연산 유닛 또는 레지스터 파일에 대응된다.

[0048] 또한, 노드(305) 및 노드(307)를 수행하는 연산 유닛이 CGA에서 서로 떨어져 있는 경우에는 노드(305)의 수행 결과를 노드(307)로 전달해주기 위한 라우팅 경로를 생성하여야 한다. 노드들(501, 502, 504)은 노드(305)의 수행 결과를 노드(307)로 전달해주기 위한 라우팅 경로상의 연산 유닛 또는 레지스터 파일에 대응된다.

[0049] 도 6은 일 실시예에 따른 명령어 컴파일링 방법을 나타내는 순서도이다.

[0050] 오퍼랜드의 타입 정보를 이용하여 데이터 플로우 그래프상의 각 오퍼랜드 및 각 에지에 대한 데이터 폭을 결정

한다(S 610). 오퍼랜드의 타입 정보에 기초하여 데이터 플로우 그래프상에 입출력 노드에 대한 데이터 폭을 초기화하고, 고정점 알고리즘을 이용하여 미지의 오퍼랜드 및 에지에 대한 데이터 폭을 결정할 수 있다.

[0051] 각 오퍼랜드 및 각 에지에 대한 데이터 폭에 기초하여 서로 다른 데이터 폭을 지원하는 이종의 구성요소를 포함하는 이종 프로세서에서 수행될 명령어를 제공한다(S 620). 전술한 바와 같이, 이종의 구성요소는 서로 다른 데이터 폭의 데이터를 처리하는 복수 개의 연산 유닛, 복수 개의 레지스터 파일 및 서로 다른 데이터 폭을 가지는 연결 와이어 중 적어도 하나를 포함할 수 있다.

[0052] 데이터 플로우 그래프상의 각 오퍼랜드 및 각 에지에 대해 결정된 데이터 폭에 기초하여 명령어를 선택하고, 선택된 명령어를 수행할 수 있는 연산 유닛을 결정한다. 또한, 데이터 플로우 그래프상의 각 오퍼랜드 및 각 에지에 대해 결정된 데이터 폭에 기초하여 레지스터를 할당한다.

[0053] 프로세서가 코어스 그레인 어레이인 경우 코어스 그레인 어레이에서 데이터 플로우 그래프상의 각 오퍼랜드 및 각 에지에 대해 결정된 데이터 폭에 기초하여 선택된 명령어를 수행하기 위한 라우팅 경로상의 노드에 대한 입출력 오퍼랜드 및 에지에 대한 데이터 폭을 결정한다.

[0054] 일 실시예에 따르면, 칩영역 및 전력소모가 감소되는 이종의 연산 처리를 수행할 수 있는 프로세서에서 이용하기 위하여 소스 코드를 데이터 폭(data width)을 고려하여 효율적으로 컴파일링할 수 있다.

[0055] 본 발명의 일 양상은 컴퓨터로 읽을 수 있는 기록 매체에 컴퓨터가 읽을 수 있는 코드로서 구현될 수 있다. 상기의 프로그램을 구현하는 코드들 및 코드 세그먼트들은 당해 분야의 컴퓨터 프로그래머에 의하여 용이하게 추론될 수 있다. 컴퓨터가 읽을 수 있는 기록매체는 컴퓨터 시스템에 의하여 읽혀질 수 있는 데이터가 저장되는 모든 종류의 기록 장치를 포함한다. 컴퓨터가 읽을 수 있는 기록 매체의 예로는 ROM, RAM, CD-ROM, 자기 테이프, 플로피 디스크, 광 디스크 등을 포함한다. 또한, 컴퓨터가 읽을 수 있는 기록 매체는 네트워크로 연결된 컴퓨터 시스템에 분산되어, 분산 방식으로 컴퓨터가 읽을 수 있는 코드로 저장되고 실행될 수 있다.

[0056] 이상의 설명은 본 발명의 일 실시예에 불과할 뿐, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자는 본 발명의 본질적 특성에서 벗어나지 않는 범위에서 변형된 형태로 구현할 수 있을 것이다. 따라서, 본 발명의 범위는 전술한 실시예에 한정되지 않고 특허 청구범위에 기재된 내용과 동등한 범위 내에 있는 다양한 실시 형태가 포함되도록 해석되어야 할 것이다.

도면의 간단한 설명

[0057] 도 1은 일 실시예에 따른 이종 프로세서(heterogeneous processor)의 구성을 나타내는 블록도이다.

[0058] 도 2는 도 1의 이종 프로세서에 이용하기 위한 일 실시예에 따른 명령어 컴파일링 장치의 구성을 나타내는 블록도이다.

[0059] 도 3은 일 실시예에 따른 데이터 플로우 그래프의 일 예를 나타내는 도면이다.

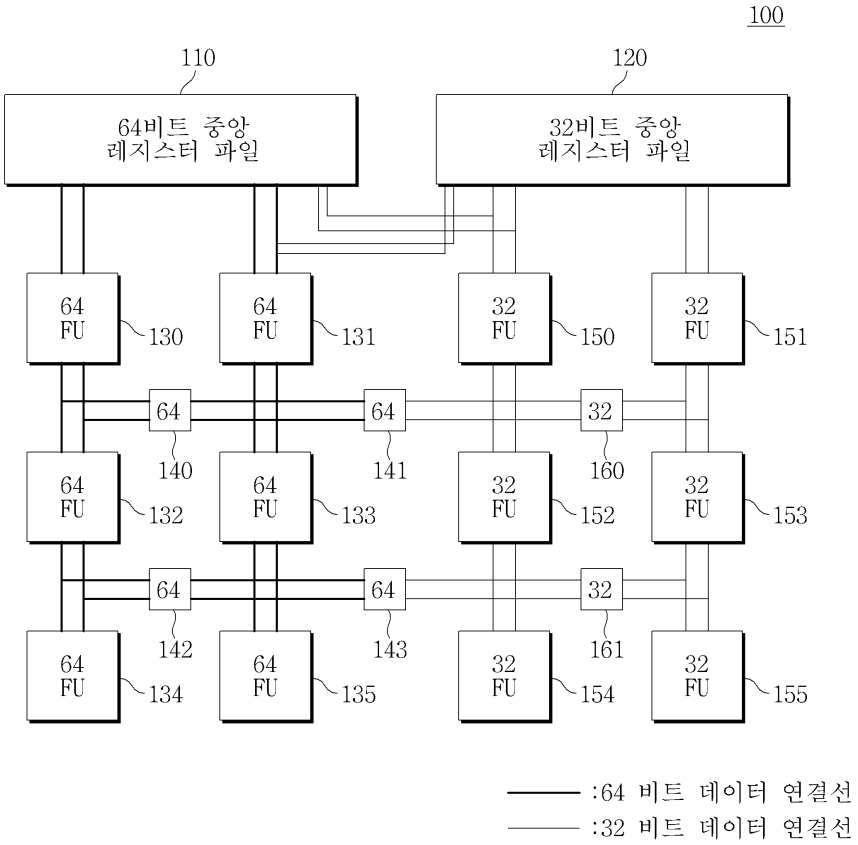
[0060] 도 4는 도 3의 데이터 플로우 그래프를 레지스터를 고려하여 변형한 데이터 플로우 그래프를 나타내는 도면이다.

[0061] 도 5는 일 실시예에 따른 라우팅을 고려하여 생성된 데이터 플로우 그래프의 일 예를 나타내는 도면이다.

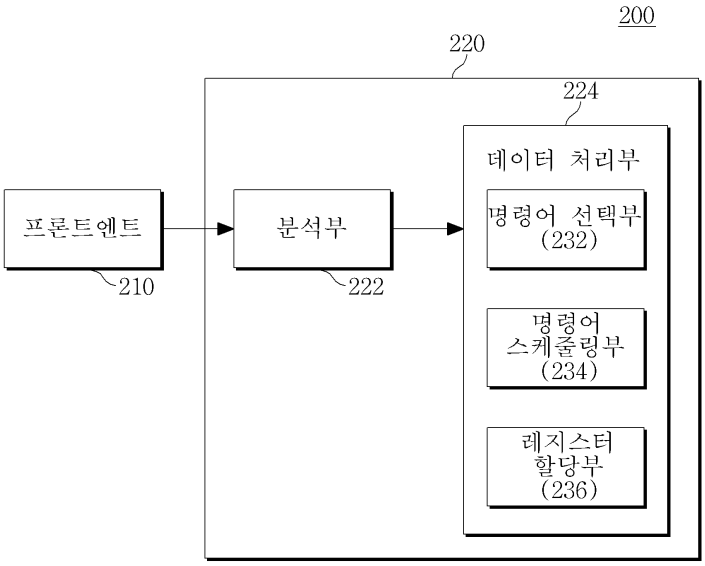
[0062] 도 6은 일 실시예에 따른 명령어 컴파일링 방법을 나타내는 순서도이다.

도면

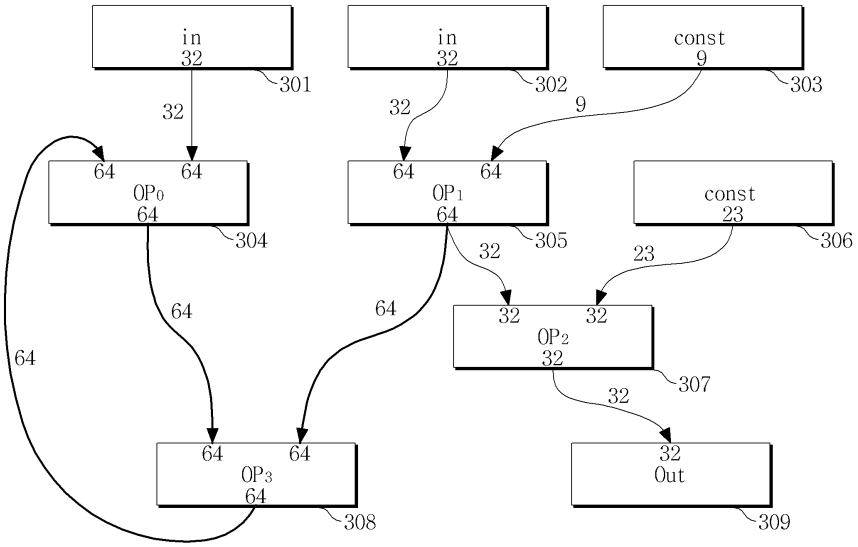
도면1


100
110
64비트 중앙
레지스터 파일
120
32비트 중앙
레지스터 파일
64
FU
130
64
FU
131
32
FU
150
32
FU
151
64
140
64
141
32
160
64
FU
132
64
FU
133
32
FU
152
32
FU
153
64
142
64
143
32
161
64
FU
134
64
FU
135
32
FU
154
32
FU
155
:64 비트 데이터 연결선
:32 비트 데이터 연결선

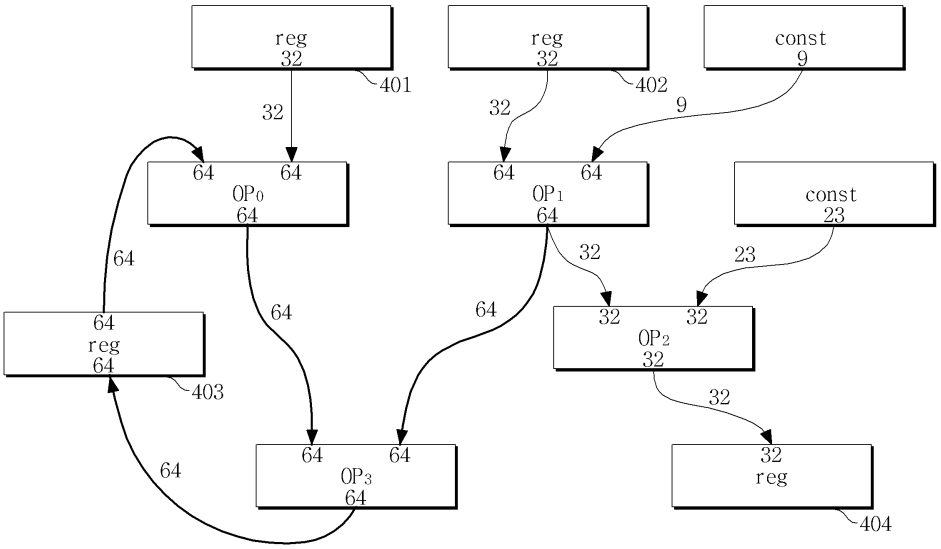
도면2


200
220
224
데이터 처리부
프론트엔트
210
분석부
222
명령어 선택부
(232)
명령어
스케줄링부
(234)
레지스터
할당부
(236)

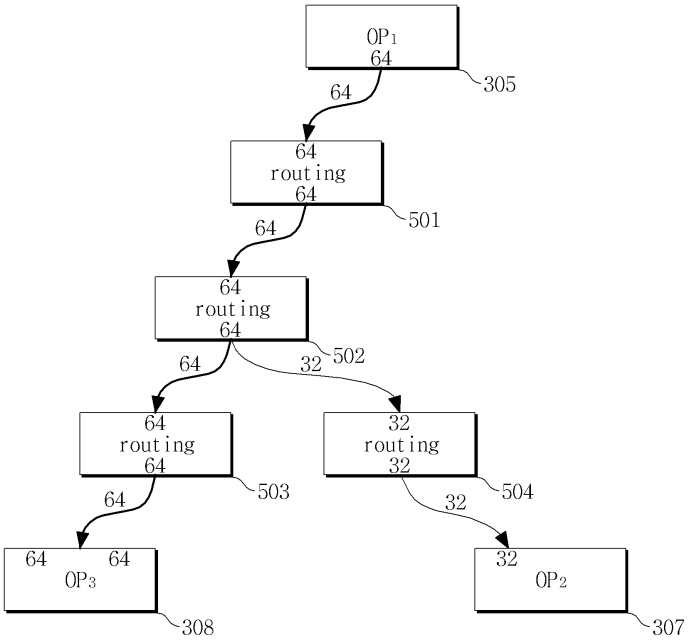
도면3


in
32
301
in
32
302
const
9
303
32
32
9
64
64
OP0
64
304
64
64
OP1
64
305
const
23
306
32
23
64
64
64
32
32
OP2
32
307
32
64
64
OP3
64
308
32
Out
309

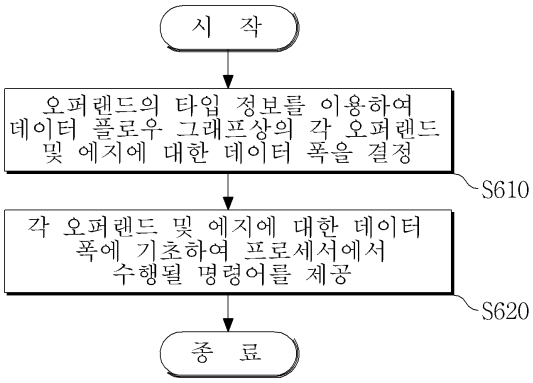
도면4


reg
32
401
reg
32
402
const
9
32
32
9
64
64
OP0
64
64
64
OP1
64
const
23
64
32
23
64
reg
64
403
64
64
64
32
32
OP2
32
32
64
64
OP3
64
32
reg
404

도면5


OP1
64
305
64
64
routing
64
501
64
64
routing
64
502
64
32
64
routing
64
503
32
routing
32
504
64
32
64
64
OP3
308
32
OP2
307

도면6


시 작
오퍼랜드의 타입 정보를 이용하여
데이터 플로우 그래프상의 각 오퍼랜드
및 에지에 대한 데이터 폭을 결정
S610
각 오퍼랜드 및 에지에 대한 데이터
폭에 기초하여 프로세서에서
수행될 명령어를 제공
S620
종 료