(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2021-0112330
(43) 공개일자 2021년09월14일

(51) 국제특허분류(Int. Cl.)
G06F 8/41 (2018.01) *G06F 30/327* (2020.01)
G06F 30/398 (2020.01)
(52) CPC특허분류
G06F 8/4452 (2013.01)
G06F 1/12 (2013.01)
(21) 출원번호 10-2021-7021969
(22) 출원일자(국제) 2020년01월04일
심사청구일자 없음
(85) 번역문제출일자 2021년07월13일
(86) 국제출원번호 PCT/US2020/012278
(87) 국제공개번호 WO 2020/150013
국제공개일자 2020년07월23일
(30) 우선권주장
16/247,269 2019년01월14일 미국(US)

(71) 출원인
마이크로소프트 테크놀로지 라이센싱, 엘엘씨
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
펠튼 블레이크 디
미국 워싱턴주 98052-6399 레드몬드 원 마이크로소프트 웨이 마이크로소프트 테크놀로지 라이센싱, 엘엘씨
카울필드 아드리안 마이클
미국 워싱턴주 98052-6399 레드몬드 원 마이크로소프트 웨이 마이크로소프트 테크놀로지 라이센싱, 엘엘씨
(74) 대리인
제일특허법인(유)

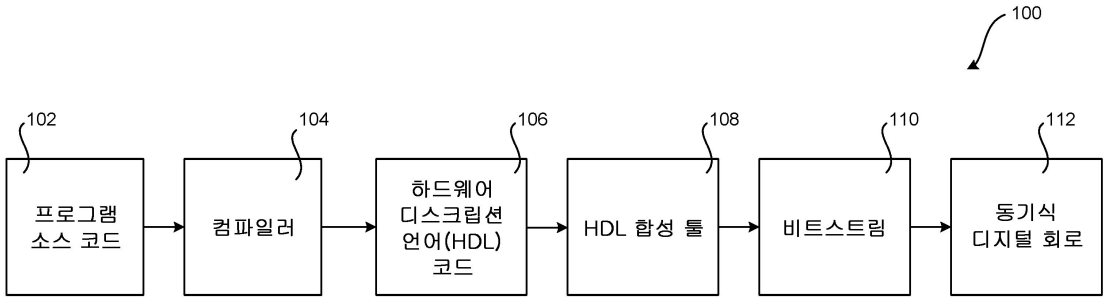
전체 청구항 수 : 총 15 항

(54) 발명의 명칭 **스레드 실행 순서를 유지하는 동기식 디지털 회로를 생성하는 언어 및 컴파일러**

(57) 요 약

멀티-스레딩된 프로그래밍 언어 및 컴파일러는 동일한 수의 스테이지를 갖는 코드 경로가 있는 파이프라인을 생성함으로써 스레드 실행 순서를 유지하는 동기식 디지털 회로를 생성한다. 컴파일러는 더 적은 스테이지를 갖는 코드 경로에 추가 스테이지를 추가함으로써 파이프라인 내의 관련 코드 경로를 밸런싱한다. 설계에 의해, 스레드 실행이 재정렬될 수 있게 하는 프로그래밍 구성자는, 스레드가 프로그래밍 구성자에 진입한 순서대로 스레드를 해제하는 재정렬 블록 구성자 내에 배치될 수 있다. 선입선출(FIFO) 큐는 파이프라인 사이에서 로컬 변수를 전달한다. 로컬 변수는 FIFO로부터 푸시되었던 순서로 팝핑되어, 파이프라인에 걸쳐 스레드 실행 순서를 보존한다.

대 표 도



(52) CPC특허분류
G06F 30/327 (2020.01)
G06F 30/398 (2020.01)
G06F 8/41 (2013.01)
G06F 8/43 (2013.01)

명 세 서

청구범위

청구항 1

컴퓨터로 구현된 방법으로서,

멀티-스레딩된 프로그래밍 언어로 표현된 소스 코드를 수신하는 단계 - 상기 소스 코드는 복수의 소스 코드 경로 중 하나로 실행을 지시하는 분기문을 포함함 - 와,

파이프라인을 포함하는 회로 디스크립션(circuit description)으로 상기 소스 코드를 컴파일하는 단계 - 상기 파이프라인은 상기 복수의 소스 코드 경로와 연관된 복수의 코드 경로를 포함하고, 하나 이상의 파이프라인 스테이지가 상기 복수의 코드 경로 중 하나 이상에 추가되어 상기 복수의 코드 경로가 동일한 수의 파이프라인 스테이지를 가짐 - 와,

상기 회로 디스크립션에 기초하여, 회로 구현을 포함하는 동기식 디지털 회로를 생성하는 단계를 포함하는

컴퓨터로 구현된 방법.

청구항 2

제1항에 있어서,

복수의 스레드가 제1 순서대로 상기 파이프라인에 진입하고, 상기 복수의 스레드가 상기 제1 순서대로 상기 파이프라인을 떠나는

컴퓨터로 구현된 방법.

청구항 3

제1항 또는 제2항에 있어서,

상기 복수의 코드 경로 중 하나 이상에 파이프라인 스테이지를 추가하는 것은, 최장 코드 경로 내의 파이프라인 스테이지의 수를 결정하는 것 및 상기 최장 코드 경로 내의 파이프라인 스테이지의 수에서 각각의 코드 경로 내의 파이프라인 스테이지의 수를 뺀 것과 동일한 수의 파이프라인 스테이지를 상기 각각의 코드 경로에 추가하는 것을 포함하는

컴퓨터로 구현된 방법.

청구항 4

제1항 내지 제3항 중 어느 한 항에 있어서,

상기 파이프라인은 제1 파이프라인을 포함하고, 상기 회로 디스크립션은 제2 파이프라인을 포함하며, 상기 제1 파이프라인을 실행하는 스레드는 로컬 변수를 선입선출 큐(first-in-first-out queue)로 푸시함으로써 상기 제2 파이프라인으로 실행을 이송하고, 상기 제2 파이프라인은 상기 선입선출 큐로부터 로컬 변수를 상기 로컬 변수가 푸시되었던 순서대로 판독함으로써 파이프라인에 걸쳐 스레드 실행 순서를 유지하는

컴퓨터로 구현된 방법.

청구항 5

제1항 내지 제4항 중 어느 한 항에 있어서,

상기 소스 코드는 스레드 실행 순서를 유지하지 않는 프로그래밍 구성자를 래핑(wrap)하는 재정렬 블록 구성자를 포함하고, 상기 재정렬 블록 구성자는 회로 구현에 매핑하며, 상기 회로 구현은,

인입 스레드 실행 순서를 기록하고,

스레드가 스레드 실행 순서를 유지하지 않는 상기 구성자를 실행할 수 있게 하며,

모든 하위 순서의 스레드가 재개될 때까지 스레드를 재개하는 것을 차단하는

컴퓨터로 구현된 방법.

청구항 6

제1항 내지 제5항 중 어느 한 항에 있어서,

스레드는 실행을 위해 상기 파이프라인에 제공되는 로컬 변수의 집합을 포함하는

컴퓨터로 구현된 방법.

청구항 7

제1항 내지 제6항 중 어느 한 항에 있어서,

상기 파이프라인은 순서대로 실행되는 스테이지를 포함하고, 복수의 스레드는 상기 순서에서 상기 스테이지를 통해 흐름으로써 실행 순서를 유지하는

컴퓨터로 구현된 방법.

청구항 8

컴퓨팅 디바이스로서,

하나 이상의 프로세서와,

컴퓨터 실행가능 명령어가 저장된 적어도 하나의 컴퓨터 저장 매체를 포함하되,

상기 컴퓨터 실행가능 명령어는, 상기 하나 이상의 프로세서에 의해 실행될 때, 상기 컴퓨팅 디바이스로 하여금,

멀티-스레딩된 프로그래밍 언어로 표현된 소스 코드를 수신하게 하고,

제1 파이프라인, 제2 파이프라인, 및 상기 제1 파이프라인으로부터 상기 제2 파이프라인으로 전달되는 로컬 스레드 변수의 세트를 저장하는 선입선출(FIFO) 큐를 포함하는 회로 디스크립션으로 상기 소스 코드를 컴파일하게 하며 - 상기 제1 파이프라인은 스레드 실행 순서대로 상기 FIFO 큐에 로컬 스레드 변수의 세트를 저장하고, 상기 제2 파이프라인은 상기 스레드 실행 순서에서 상기 FIFO 큐로부터 로컬 스레드 변수의 세트를 취출함으로써 상기 스레드 실행 순서를 유지함 - ,

상기 회로 디스크립션에 기초하여, 회로 구현을 포함하는 동기식 디지털 회로를 생성하게 하는

컴퓨팅 디바이스.

청구항 9

제8항에 있어서,

상기 소스 코드는 복수의 소스 코드 경로 중 하나로 실행을 지시하는 분기문을 포함하고,

상기 제1 파이프라인은 상기 복수의 소스 코드 경로와 연관된 복수의 코드 경로를 포함하고, 하나 이상의 파이

프라인 스테이지가 상기 복수의 코드 경로 중 하나 이상에 추가되어 상기 복수의 코드 경로가 동일한 수의 파이프라인 스테이지를 갖는

컴퓨팅 디바이스.

청구항 10

제8항 또는 제9항에 있어서,

상기 복수의 코드 경로 중 하나 이상에 파이프라인 스테이지를 추가하는 것은, 최장 코드 경로 내의 파이프라인 스테이지의 수를 결정하는 것 및 상기 최장 코드 경로 내의 파이프라인 스테이지의 수에서 각각의 코드 경로 내의 파이프라인 스테이지의 수를 뺀 것과 동일한 수의 파이프라인 스테이지를 상기 각각의 코드 경로에 추가하는 것을 포함하는

컴퓨팅 디바이스.

청구항 11

제8항 내지 제10항 중 어느 한 항에 있어서,

상기 소스 코드는 스레드 실행 순서를 유지하지 않는 프로그래밍 구성자를 래핑하는 재정렬 블록 구성자를 포함하고, 상기 재정렬 블록 구성자는 회로 구현에 매핑하며, 상기 회로 구현은,

인입 스레드 실행 순서를 기록하고,

스레드가 스레드 실행 순서를 유지하지 않는 상기 구성자를 실행할 수 있게 하며,

모든 하위 순서의 스레드가 재개될 때까지 스레드를 재개하는 것을 차단하는

컴퓨팅 디바이스.

청구항 12

제8항 내지 제11항 중 어느 한 항에 있어서,

스레드는 진입한 순서대로 상기 재정렬 블록을 떠나는

컴퓨팅 디바이스.

청구항 13

제8항 내지 제12항 중 어느 한 항에 있어서,

스레드는 실행을 위해 상기 제1 파이프라인에 제공되는 로컬 스레드 변수의 집합을 포함하는

컴퓨팅 디바이스.

청구항 14

제8항 내지 제13항 중 어느 한 항에 있어서,

상기 제1 파이프라인은 순서대로 실행되는 스테이지를 포함하고, 복수의 스레드는 상기 순서에서 상기 스테이지를 통해 흐름으로써 실행 순서를 유지하는

컴퓨팅 디바이스.

청구항 15

컴퓨터 실행가능 명령어가 저장된 적어도 하나의 컴퓨터 저장 매체로서,

상기 컴퓨터 실행가능 명령어는, 하나 이상의 프로세서에 의해 실행될 때, 컴퓨팅 디바이스로 하여금,

멀티-스레딩된 프로그래밍 언어로 표현된 소스 코드를 수신 - 상기 소스 코드는 회로 구현에 매핑하는 구성자를 포함하고, 상기 구성자는 재정렬 블록 및 스레드 실행 순서를 유지하지 않는 구성자를 포함하며,

상기 회로 구현은,

복수의 스레드가 수신되는 순서대로 스레드 식별자를 등록하는 재정렬 버퍼와,

상기 복수의 스레드의 각각에 대해 미지의 수의 클록 사이클에 대해 실행하는 회로를 포함하며, 상기 재정렬 버퍼는 더 낮은 실행 순서를 갖는 모든 스레드가 재개될 때까지 스레드가 재개하는 것을 차단함 - 하게 하고,

상기 구성자를 회로 디스크립션으로 컴파일하게 하며,

상기 회로 디스크립션에 기초하여, 상기 회로 구현을 포함하는 동기식 디지털 회로를 생성하게 하는

적어도 하나의 컴퓨터 저장 매체.

발명의 설명

기 술 분 야

배 경 기 술

[0001] 하드웨어 디스크립션 언어(Hardware description language: "HDL")는 전자 회로, 가장 일반적으로 디지털 로직 회로의 구조 및 거동을 설명하기 위해 하드웨어 엔지니어에 의해 사용되는 모델링 언어이다. HDL의 예는 초고속 집적 회로(Very High Speed Integrated Circuit: "VHSIC") HDL 및 VERILOG를 포함한다.

[0002] HDL은 일반적으로 디지털 로직 회로를 모델링하기 위해 많은 라인의 코드를 필요로 한다. HDL에 매우 익숙한 하드웨어 엔지니어의 경우에도, 이러한 코드를 만드는 데 시간이 많이 걸릴 수 있다. 더욱이, 설계에 존재하는 코드의 라인이 많을수록, 설계가 에러를 포함하거나 성능이 저하될 가능성이 높아진다.

[0003] HDL은 전형적으로 명령형 프로그래밍 언어와 다른 프로그래밍 패러다임을 사용하기 때문에 HDL에 익숙하지 않은 소프트웨어 엔지니어는 일반적으로 이러한 언어를 이용하는 데 매우 어려움을 겪는다. 결과적으로, 소프트웨어 엔지니어가 만든 HDL로부터 생성된 전자 회로도 오류를 포함하거나 성능이 저하될 수 있다.

[0004] 본 명세서에서 이루어진 개시가 제시되는 것은 이들 및 다른 기술적 도전과제에 관한 것이다.

발명의 내용

[0005] 스레드 실행 순서를 유지하는 동기식 디지털 회로를 생성하는 언어 및 컴파일러에 대한 기술들이 개시된다. 개시된 기술의 구현을 통해, 하드웨어 엔지니어는 일부 유형의 회로 설계를 구현하는 데 요구되는 코드의 라인의 수를 감소시킴으로써, 그리고 공통 설계 에러의 전체 클래스를 제거하는 동시에 성능을 희생하지 않음으로써 상당한 생산성 이득을 실현할 수 있다. 회로를 설계하는 데 HDL을 사용하는 경험이 거의 없거나 전혀 없는 소프트웨어 엔지니어의 경우, 개시된 기술은 고성능 회로 설계를 생성하는 데 사용될 수 있는 익숙한 프로그래밍 시멘틱을 제공한다.

[0006] 개시된 기술의 구현은 스레드 실행 순서를 유지할 수 있는 하드웨어 회로도 생성할 수 있다. 스레드 실행 순서를 유지하는 것은 저-오버헤드 스레드 동기화와 같은 최적화를 가능하게 한다. 스레드 실행 순서를 유지하는 것은 또한 프로그래머가 순서 특정 동작을 수행할 수 있게 한다. 예를 들어, 짝수 실행 순서의 스레드(예를 들어, 스레드 '2', '4', '6'...)는 홀수 실행 순서의 스레드와는 프로그램 전반에 걸쳐 상이하게 취급될 수 있다. 또한, 스레드가 실행될 첫 번째 스레드 또는 마지막 스레드일 때 특별한 동작이 수행될 수 있다. 본 명세서에서

구체적으로 언급되지 않은 다른 기술적 이점이 또한 개시된 출원 대상(subject matter)의 구현을 통해 실현될 수 있다.

[0007] 앞에서 간략하게 언급된 기술적 이점을 실현하기 위해, 개시된 언어 및 컴파일러는 동일한 수의 스테이지를 갖는 코드 경로가 있는 파이프라인을 생성한다. 예를 들어, 제1 파이프라인의 모든 코드 경로는 6개의 스테이지를 가질 수 있는 반면, 제2 파이프라인의 모든 코드 경로는 15개의 스테이지를 가질 수 있다. 이러한 특성을 유지하기 위해, 컴파일러는 파이프라인 내의 관련된 코드 경로를 밸런싱한다. 예를 들어, 주어진 'if' 문의 경우, 'then' 블록 또는 'else' 블록은 다른 것과 동일한 수의 스테이지를 갖도록 패딩될 수 있다. 유용한 작업을 수행하지 않는 더미 스테이지를 추가함으로써 스테이지가 추가될 수 있거나, 또는 스테이지는 파이프라인 깊이를 증가시킴으로써, 즉, 블록을 구현하는 데 사용되는 스테이지의 수를 증가시킴으로써 추가될 수 있다. 'then' 및 'else' 블록에서의 스테이지의 수를 밸런싱함으로써, 모든 스레드는 동일한 수의 클록 사이클에서 'if' 문을 실행하고, 따라서 스레드들은 그들이 진입한 순서대로 'if' 문을 떠난다.

[0008] 몇몇 프로그래밍 구성자는 스레드 실행이 설계에 의해 재정렬될 수 있게 한다. 예를 들어, 상이한 스레드가 상이한 횟수로 루프를 실행할 수 있기 때문에, 스레드 실행 순서는 루프에 대해 보장되지 않는다. 이와 같이, 하나의 스레드는 10회 반복으로 루프를 완료할 수 있는 반면, 다른 스레드는 500회 반복으로 동일한 루프를 완료할 수 있다. 그러나, 개시된 언어는 스레드들이 리오더 블록을 그들이 진입한 순서대로 떠나는 것을 보장하는 재정렬 블록을 포함한다. 프로그래머는 루프 및 다른 비순서-보존 구성자를 재정렬 블록으로 랩핑하여 스레드 실행 순서를 유지할 수 있다. 추가적으로 또는 대안적으로, 개시된 언어는 재정렬 블록 기능을 "빌드 인"하는 루프 구성자를 포함할 수 있다. 예를 들어, "reorder_for" 루프는 스레드 실행 순서를 보존하는 "for" 루프로서 동작한다. 일부 구성에서, 재정렬 블록은 SDC 상에서 재정렬 버퍼를 사용하여 구현된다.

[0009] 몇몇 구성에서, 실행의 파이프라인은 다른 파이프라인에 의해 사용될 로컬 변수를 저장하는 선입선출 큐('큐' 또는 'FIFO'로 지칭됨)에 의해 연결된다. FIFO들은 함수 호출, 제어 흐름, 및 개시된 언어의 다른 양상을 구현하는 데 사용될 수 있다. 스레드 순서는 FIFO로부터 로컬 변수를 이들이 인큐잉되었던 순서대로 취출함으로써 파이프라인에 걸쳐 유지된다.

[0010] 앞에서 간략하게 논의된 바와 같이, 본 명세서에 개시된 기술의 구현은 저-오버헤드 스레드 동기화를 제공한다. 또한, 프로그래머는 짝수 및 홀수 스레드를 상이하게 처리하는 것, 또는 첫 번째 또는 마지막 스레드에 대한 특별 동작을 수행하는 것과 같은 순서 특정 동작을 수행하는 것이 가능해진다. 본 명세서에서 구체적으로 식별되지 않은 다른 기술적 이점도 개시된 기술의 구현을 통해 실현될 수 있다.

[0011] 전술한 출원 대상은 컴퓨터로 제어된 장치, 컴퓨터로 구현된 방법, 컴퓨팅 디바이스로서, 또는 컴퓨터 판독가능 매체와 같은 제조 물품으로서 구현될 수 있다는 것을 알아야 한다. 이들 및 다양한 다른 특징은 이하의 상세한 설명을 읽고 관련 도면을 검토함으로써 명백해질 것이다.

[0012] 이 요약은, 상세한 설명에서 아래에 추가로 설명되는 간략화된 형태로 개시된 기술의 몇몇 양상의 간단한 설명을 도입하기 위해 제공된다. 본 요약은 청구된 출원 대상의 주요 특징 또는 기본 특징을 식별하기 위한 것이 아니며, 본 요약이 청구된 출원 대상의 범위를 제한하는 데 사용되는 것도 아니다. 또한, 청구된 출원 대상은 본 개시의 임의의 부분에서 언급된 임의의 또는 모든 단점을 해결하는 구현으로 제한되지 않는다.

도면의 간단한 설명

[0013] 도 1은 동기식 디지털 회로에 매핑하는 프로그래밍 구성자를 포함하는 프로그램 소스 코드에 기초하여 동기식 디지털 회로를 생성하기 위해 본 명세서에 개시된 시스템의 양상을 도시하는 컴퓨팅 아키텍처 도면이다.

도 2는 개시된 기술을 사용하여 정의되고 구현될 수 있는 계산 유닛 및 각각 복수의 파이프라인 스테이지를 갖는 몇몇 하드웨어 파이프라인을 포함하는 예시적인 동기식 디지털 회로의 양상들을 도시하는 하드웨어 아키텍처 도면이다.

도 3은 개시된 언어로 기록된 함수를 하드웨어 회로에 매핑하는 것을 도시하는 하드웨어 아키텍처 도면이다.

도 4는 더미 스테이지를 갖는 파이프라인의 패딩 코드 경로를 도시하는 하드웨어 아키텍처 도면이다.

도 5는 for-루프에 적용되는 재정렬 블록 구성자를 도시하는 하드웨어 아키텍처 도면이다.

도 6은 도 1 내지 도 5를 참조하여 설명된 스레드 실행 순서를 유지하는 언어 및 컴파일러의 동작의 양상을 예시하는 루틴을 도시하는 흐름도이다.

도 7은 본 명세서에 제시된 기술의 양상을 구현할 수 있는 컴퓨팅 디바이스에 대한 예시적인 컴퓨터 하드웨어 및 소프트웨어 아키텍처를 도시하는 컴퓨터 아키텍처 도면이다.

도 8은 개시된 기술들의 양상이 구현될 수 있는 분산 컴퓨팅 환경을 예시하는 네트워크 도면이다.

발명을 실시하기 위한 구체적인 내용

[0014] 이하의 상세한 설명은 스레드 실행 순서를 유지하는 언어 및 컴파일러에 관한 것이다. 앞에서 간략하게 논의된 바와 같이, 스레드 실행 순서를 유지하는 것은 저-오버헤드 스레드 동기화와 같은 최적화를 가능하게 한다. 스레드 실행 순서를 유지하는 것은 또한 프로그래머가 순서 특정 동작들을 구현할 수 있게 한다. 예를 들어, 짝수 실행 순서의 스레드는 홀수 실행 순서의 스레드와는 프로그램 전반에 걸쳐 상이하게 취급될 수 있다. 또한, 스레드가 실행될 첫 번째 스레드 또는 마지막 스레드일 때 특별한 동작이 수행될 수 있다. 본 명세서에서 구체적으로 언급되지 않은 다른 기술적 이점이 또한 개시된 출원 대상의 구현을 통해 실현될 수 있다.

[0015] 본 명세서에 설명된 출원 대상이 스레드 실행 순서를 유지하는 언어 및 컴파일러의 일반적인 맥락에서 제시되지만, 당업자는 다른 구현이 다른 유형의 컴퓨팅 시스템 및 모듈과 결합하여 수행될 수 있다는 것을 인식할 것이다. 당업자는 또한 본 명세서에 설명된 출원 대상이 휴대용 디바이스, 멀티프로세서 시스템, 마이크로프로세서 기반 또는 프로그램가능한 소비자 전자제품, 디바이스(예컨대, 착용형 컴퓨팅 디바이스, 자동차, 홈 오토메이션 등)에 내장된 컴퓨팅 또는 프로세싱 시스템, 미니컴퓨터, 메인프레임 컴퓨터 등을 포함하는 다른 컴퓨터 시스템 구성으로 실시될 수 있다는 것을 이해할 것이다.

[0016] 다음의 상세한 설명에서, 본 명세서의 일부를 형성하고 특정 구성 또는 예가 예로서 도시된 첨부 도면을 참조한다. 이제, 여러 도면에 걸쳐 동일한 번호가 동일한 요소를 나타내는 도면을 참조하여, 스레드 실행 순서를 유지하는 언어 및 컴파일러의 양상이 설명될 것이다.

[0017] 도 1은 동기식 디지털 회로(synchronous digital circuit)("SDC")(112)에 매핑하는 프로그래밍 구성자(programming construct)를 포함하는 프로그램 소스 코드(102)에 기초하여 SDC(112)를 정의하고 생성하기 위해 본 명세서에 개시된 예시적인 시스템(100)의 양상을 도시하는 컴퓨팅 아키텍처 도면이다. SDC(112)는 게이트 어레이, 필드 프로그래밍가능 게이트 어레이("FPGA"), 주문형 집적 회로("ASIC") 및 다른 유형의 회로 디바이스에 의해 구현될 수 있다. 개시된 출원 대상이 주로 FPGA에서 구현되는 SDC(112)의 맥락에서 설명되지만, 본 명세서에 개시된 기술은 다른 유형의 디바이스를 사용하여 구현되는 SDC(112)를 정의하는 데 이용될 수 있다는 것을 알아야 한다.

[0018] 도 1에 도시된 바와 같이, 예시적인 시스템(100)은 하드웨어 디스크립션 언어(hardware description language)("HDL") 코드(106) 또는 네트리스트(netlist)와 같은 회로의 하위-레벨 표현을 생성하기 위해 프로그램 소스 코드(102)를 컴파일하는 컴파일러(104)를 포함한다. 앞에서 간략히 논의된 바와 같이, HDL은 전자 회로, 가장 일반적으로 디지털 로직 회로의 구조 및 동작을 설명하기 위해 하드웨어 엔지니어에 의해 사용되는 모델링 언어이다. HDL의 예는 VHSIC HDL 및 VERILOG를 포함한다.

[0019] 아래에서 상세히 설명되는 바와 같이, 프로그램 소스 코드(102)는 SDC(112)를 타깃으로 하도록 설계된 멀티-스레딩된 명령형 프로그래밍 언어를 사용하여 표현된다. 개시된 언어는 'C' 및 'JAVA', 예컨대, 함수 호출, for-루프, 산술 연산자, 및 조건문과 같은 언어의 많은 특징을 제공한다. 그러나, 개시된 언어는 기본 SDC(112) 하드웨어 구현에 직접 매핑하는 구성자를 포함한다. 이것은 하드웨어 및 소프트웨어 엔지니어 모두가 성능에 대해 추론할 수 있게 하고, 그들의 설계를 최적화하는 데 효과적일 수 있게 한다. 앞에서 언급된 바와 같이, 이는 또한 언어가 소프트웨어 엔지니어에게 익숙하게 하고, 하드웨어 엔지니어가 HDL로 코딩할 때 발생하는 버그의 전체 클래스를 처리하지 않아도 되게 할 수 있다.

[0020] 개시된 멀티-스레딩된 명령형 프로그래밍 언어는 프로그램문이 차례로 실행된다는 점에서, 그리고 복수의 실행 스레드가 병렬로 실행될 수 있다는 점에서 필수적이다. 앞에서 논의된 바와 같이, 스레드는 로컬 변수의 집합이다. 스레드는 로컬 변수가 하드웨어 회로에 의해 처리될 때 실행된다.

[0021] 본원에 설명된 스레드는 소프트웨어 스레드와 유사하지만 상이하기도 하다. 소프트웨어 스레드가 로컬 변수를 포함하는 호출 스택을 유지하고 메모리에서 코드를 실행하는 동안, 본 명세서에 설명된 스레드는 하드웨어 회로를 통해 이동하는 로컬 변수의 집합이다. 소프트웨어 스레드는 명령어 포인터에 의해 결정된 실행가능 코드 내의 위치를 갖지만, 개시된 스레드는 주어진 시점에 SDC 상의 물리적 위치를 갖는다. SDC는 수백, 수천, 또는 심지어 수백만 개의 스레드를 실행할 수 있고, SDC 실행은 파이프라이닝될 수 있다 - 즉, 상이한 스레드는 동시에

회로의 상이한 스테이지 내에서 실행할 수 있다.

[0022] 아래에서 더 상세히 설명되는 바와 같이, 언어 구성자는 회로 구현에 매핑하는 프로그램 소스 코드(102)에서 정의될 수 있다. 언어 구성자는 하나 이상의 어휘 토큰으로부터 형성될 수 있는 프로그램의 구문상 허용가능한 부분이다. 본 명세서에 설명된 언어 구성자는 스레드 순서화를 보장(즉, 스레드는 그들이 진입한 것과 동일한 순서대로 회로 구현을 종료할 것임)하는 회로 구현에 매핑된다.

[0023] 또한 아래에서 더 상세히 설명되는 바와 같이, 본 명세서에 개시된 구성자에 의해 생성된 회로 구현은 FPGA, 게이트 어레이, ASIC, 또는 다른 유형의 적합한 디바이스에서 SDC로서 구현될 수 있다. NIC와 같은 다른 하드웨어 컴포넌트는 원하는 기능을 구현하기 위해 FPGA, 게이트 어레이 또는 ASIC로써 구성될 수 있다.

[0024] 도 1에 도시된 바와 같이, 컴파일러(104)는 본 명세서에서 회로 디스크립션(circuit description)에 대해 개시된 하나 이상의 언어 구성자, 이 예에서 HDL 코드(106)를 포함하는 프로그램 소스 코드(102)를 컴파일할 수 있다. HDL 코드(106)는 HDL 합성 툴(108)에 제공될 수 있고, 이는 결과적으로 예를 들어, FPGA 상에서와 같이 SDC(112)를 프로그래밍하는 데 이용될 수 있는 비트스트림(110)을 생성할 수 있다. ASIC를 타깃으로 할 때, HDL 코드(106)는 공장에서의 생산을 위해 ASIC 제조자에게 제공될 수 있다.

[0025] 도 2는 개시된 기술을 사용하여 정의되고 구현될 수 있는 몇몇 하드웨어 파이프라인(200A-200C)(또는 "파이프라인")을 포함하는 예시적인 SDC(112)의 양상을 도시하는 하드웨어 아키텍처 도면이다. 각각의 하드웨어 파이프라인은 복수의 파이프라인 스테이지(206)를 가지며, 이들 각각은 계산 유닛(208)을 갖는다. 도 2에 도시된 바와 같이, 프로그램 소스 코드(102)는 하드웨어 계산 유닛(208)의 파이프라인(200A-200C)으로 컴파일될 수 있다.

[0026] 파이프라인(200A-200C)은 선입선출(first-in-first-out: "FIFO") 큐(본 명세서에서 "FIFO" 또는 "큐"로 지칭될 수 있음)에 의해 연결될 수 있다. 파이프라인(200A-200C)은 프로그램 소스 코드(102)에 의해 정의된 기능을 구현한다. FIFO(202)는 파이프라인(200)에 입력을 제공할 뿐만 아니라 파이프라인(200)에 의해 생성된 출력을 저장하는 데이터 값을 저장한다. 예를 들어, SDC(112)는 자신의 출력을 FIFO(202A)에 공급하는 파이프라인(200A)을 포함한다. 파이프라인(200B)은, 차례로, FIFO(202A)로부터 자신의 입력을 획득하고, 자신의 출력을 FIFO(202B)에 제공한다. 파이프라인(200C)은 FIFO(202B)로부터 자신의 입력을 획득한다.

[0027] 몇몇 구성에서, 파이프라인(200)은 FIFO(202)로부터 다음 값(들)을 취출할 때를 결정하는 정책 회로(210)를 구현한다. 예를 들어, 정책 회로(210)는 입력 FIFO(예를 들어, 파이프라인(200B)의 경우에 FIFO(202A))는 비어있지 않고 출력 FIFO(예컨대, FIFO(202B))는 처리를 위해 입력 FIFO(예를 들어, FIFO(202A))로부터 값을 취출하기 전에 가득 차 있지 않을 것을 요구할 수 있다.

[0028] 도 2에 도시된 바와 같이, 파이프라인(200)은 하나 이상의 파이프라인 스테이지(206A-206B)로 구성될 수 있다. 실행은 파이프라인(200)의 상이한 스테이지(206)에서 상이한 스레드를 동시에 실행함으로써 파이프라이닝된다. 스테이지의 결과는 레지스터(204)에 저장될 수 있고 다음 클록 사이클의 지속기간 동안 다음 스테이지(206)에 제공될 수 있다.

[0029] 각각의 파이프라인 스테이지(206)는 가산기(208A) 및 룩업 테이블("LUT")(208B)과 같은 하나 이상의 계산 유닛(208)을 포함할 수 있다. 도시된 예에서, 가산기(208A)는 기본 산술, 예를 들어, 가산, 감산 또는 승산을 수행할 수 있다. 계산 유닛은 또한 부울 연산자(Boolean operator)(예를 들어, "OR", "NOR" 및 "XOR" 등) 또는 SDC 제조자에 의해 제공되는 다른 커스텀 로직을 구현할 수 있다.

[0030] 계산 유닛은 또한 사용자 프로그램가능 룩업 테이블(208B)에 의해 구현될 수 있다. 예시된 LUT(208B)는 2개의 입력 비트를 단일 출력 비트에 매핑하는 2-입력 진리표(truth table)를 도시한다. LUT(208B)는 상이한 수의 입력 비트를 지원하도록 구성될 수 있다. 더 복잡한 출력 값, 예를 들어, 문자 또는 8-비트 정수를 생성하기 위해, 각각 입력 변수의 상이한 비트에 연결된 복수의 LUT(208B)가 사용될 수도 있다.

[0031] 계산 유닛은 레지스터(204)(또는 "플립-플롭")에 결과를 일시적으로 저장할 수 있다. 이러한 레지스터의 콘텐츠는 동일하거나 상이한 파이프라인(200) 내의 다른 계산 유닛에 제공될 수 있다. 레지스터(204)는 연결된 디지털 클록이 0에서 1로 전이할 때 입력에서 값을 캡처할 수 있고, 다음 클록 사이클의 종료까지(즉, 클록이 다시 0으로부터 1로 전이할 때까지) 출력에서 그 값을 제공할 수 있다. 레지스터는 인에이블 라인도 포함할 수 있다. 인에이블 라인이 false로 설정되면, 레지스터는 복수의 클록 사이클에 걸쳐 현재 출력 값을 유지하면서 앞에서 설명된 동작을 수행하지 않을 것이다.

[0032] 도 2에 도시된 파이프라인 아키텍처가 논의를 위해 단순화되었다는 것을 알아야 한다. 본 명세서에 설명된 프로

그래밍 언어 구성자는 도 2에 도시된 것보다 더 많은 컴포넌트를 포함하는 훨씬 더 복잡한 SDC(112)를 구현하는 데 이용될 수 있다.

[0033] 도 3은 일 실시예에 따른, 개시된 언어로 기록된 함수를 하드웨어 회로에 매핑하는 것을 도시하는 하드웨어 아키텍처 도면(300)이다. 프로그램 소스 코드(302)는 개시된 언어로 기록된 함수 'f()'(304)를 포함한다. 개시된 언어는, 프로그램문이 차례로 실행되고, 복수의 실행 스레드가 병렬로 및/또는 동시에 실행될 수 있다는 점에서 멀티-스레딩된다는 점에서 필수적이다. 함수 'f()'(304)는 2개의 파라미터, 'x'(306) 및 'y'(308)를 취하고, 정수를 반환한다. 함수 'f()'(304)는 2개의 식을 갖는다: 식(310)은 가산을 수행하고 그 결과를 로컬 변수 'z'에 저장하는 반면, 식(312)은 승산의 결과를 반환한다.

[0034] 함수 'f()'(304)는 계산 유닛(316 및 318) 및 레지스터(320 및 322)를 포함하는 하드웨어 회로(314)에 매핑된다(304). 하드웨어 회로(314)는 2개의 스테이지 - 식(310)에서 설명된 가산을 수행하는 제1 스테이지(324) 및 식(312)에서 설명된 승산을 수행하는 제2 스테이지(326)를 갖는다. 각각의 스테이지는 단일 클록 사이클에서 실행되고, 각각의 스테이지의 결과는 하나 이상의 레지스터에 저장된다.

[0035] 예를 계속하면, 스테이지(324)는 파라미터 'x'(306)에 대해 "+ 1" 연산을 수행하여, 결과('z')를 레지스터(322)에 저장한다. 스테이지(324) 동안, 파라미터 'y'(308)는 레지스터(320)에 직접 제공된다. 스테이지(326) 동안, 계산 유닛(318)은 'y'와 'z'의 값을 곱한다. 그 결과는 레지스터(328)에 저장된다.

[0036] 하드웨어 회로(314)는 스레드(330)에 의해 실행될 수 있고, 이들 중 일부만이 도시된다. SDC는 수백, 수천, 또는 수백만 개의 스레드를 실행할 수 있다. 스레드는 로컬 변수의 집합을 지칭한다. 스레드는 로컬 변수가 하드웨어 회로에 의해 처리될 때 실행된다. 예를 들어, 스레드(330A)는 값(332)(x=1 및 y=2)을 갖고, 스레드(330A)는 값(332)이 하드웨어 회로(314)에 의해 처리될 때 함수 'f()'(304)를 실행한다. 함수에 의해 리턴된 값은 로컬 변수의 세트에 추가될 수 있고, 특정 변수가 하드웨어 회로에 의해 더 이상 사용되지 않을 것이라고 알려지면 언제든지, 그 변수는 로컬 변수 세트로부터 제거될 수 있다.

[0037] 개시된 스레드는 소프트웨어 스레드와 유사하지만 상이하기도 하다. 소프트웨어 스레드가 로컬 변수를 포함하는 호출 스택을 유지하고 메모리에서 코드를 실행하는 동안, 개시된 스레드는 하드웨어 회로를 통해 이동하는 로컬 변수의 집합이다. 소프트웨어 스레드는 명령어 포인터에 의해 결정된 실행가능 코드 내의 위치를 갖지만, 개시된 스레드는 주어진 시점에 SDC 상의 물리적 위치를 갖는다.

[0038] SDC 실행은 파이프라이닝될 수 있다 - 즉, 상이한 스레드는 동시에 회로의 상이한 스테이지 내에서 실행될 수 있다. 표(334)는 하드웨어 회로(314)의 파이프라이닝된 실행이 발생할 때 상이한 스테이지 내의 상이한 스레드로부터의 변수를 도시한다. 열(336)은 스테이지 사이(즉, 전 및 후)에 저장된 값을 표시한다: 336A는 스레드(330A, 330B 및 330C)에 의해 제공되는 x 및 y의 값을 포함하는 한편, 열(336B)은 스테이지(324)가 실행된 후의 값을 포함하고 열(336C)은 스테이지(326)가 실행된 후의 값을 포함한다. 행(338A-C)은 연속적인 클록 사이클 후에 저장된 값을 표시한다.

[0039] 예를 들어, 행(338A)은 스레드(330A)가 값(332A)(x=1 및 y=2)으로 하드웨어 회로(314)에 진입하려고 한다는 것을 표시한다. 행(338A 및 338B) 사이에서, 클록 사이클이 발생하고, 스테이지(324)에서 스레드(330A)를 실행한 결과가 332B에 도시된다(y=2, 332A로부터의 홀드오버, 및 z=2, "+1" 연산의 결과). 동시에, 스레드(330B)(x=3 및 y=5)로부터의 값(340A)은 하드웨어 회로(314)에 진입하려고 한다. 행(338B 및 338C) 사이에서, 다른 클록 사이클이 발생하고, 스테이지(326)를 실행하는 스레드(330A)의 결과가 332C("4")에 도시된다. 동시에, 스테이지(324)를 실행하는 스레드(330B)의 결과는 340B(y=5 및 z=4)에 도시되고, 스레드(330C)로부터의 값(342)은 하드웨어 회로(314)(x=7 및 y=1)에 진입하려고 한다. 파이프라이닝된 실행은 SDC 활용을 증가시킴으로써 더 높은 스루풋을 가능하게 한다 - 즉, 한번에 하나의 스레드만이 하드웨어 회로(305)를 실행할 수 있는 경우보다 더 많은 SDC가 주어진 시점에 유용한 작업을 수행하고 있다.

[0040] 도 4는 파이프라인의 모든 경로가 동일한 수의 스테이지를 갖도록 더미 스테이지를 갖는 파이프라인의 패딩 코드 경로를 도시하는 하드웨어 아키텍처 도면(400)이다. 동일한 수의 스테이지를 갖는 것은 파이프라인을 통해 스레드 실행 순서가 유지되는 것을 보장한다. 몇몇 구성에서, 코드 리스팅(402)은 함수 'f()'(404)를 포함하며, 이는 2개의 파라미터: int 'x'(406) 및 int 'y'(408)를 수용한다. 라인(410)은 'x'에 '1을 가산하고 값을 'z'에 할당하며, 이는 계산 유닛(420)이 '1'을 'x'에 가산하고 결과를 레지스터(422)에 저장하는 것으로 회로 디스크립션(418)에 도시된다.

[0041] 라인(411)은 "z > 2"의 조건을 갖는 "if" 문을 도입한다. 몇몇 구성에서, 'if' 문과 같은 분기문(branching

statement)은 양 분기들을 병렬로 실행하고 조건에 기초하여 어느 결과를 진행시킬지를 선택함으로써 구현된다. 회로 디스크립션(418) 내의 대응하는 컴포넌트는 멀티플렉서(424)이며, 이는 'z'가 '2'보다 큰 경우 스레드가 코드 경로(427)에 의해 생성된 값으로 진행하게 하고, 그렇지 않으면 스레드가 코드 경로(435)에 의해 생성되는 값(N)으로 진행하게 한다.

[0042] 코드 블록(412)('then' block)은 2개의 문을 포함한다. "y = y + 3"은 'y'(408)의 값에 '3'을 가산하는 계산 유닛(428)에 대응한다. 그 결과는 레지스터(430)에 저장되고, 컴파일러는 'y'의 정확한 값을 포함하는 것으로 이해하며, 파라미터로서 전달되었던 값을 대체한다. 다음 문 "y = y * 2"는 레지스터(430)에 저장된 'y'의 값을 '2'와 곱하고 그 결과를 멀티플렉서(424)에 제공하는 계산 유닛(432)에 의해 구현된다. 'z'의 값이 '2'보다 크면, 계산 유닛(432)에 의해 제공된 값은 레지스터(434)에 제공될 것이다.

[0043] 코드 블록(414)('else' 블록)은 단일문 "y = y - 1"을 포함한다. 이 문은 'y'(408)로부터 '1'을 감산하고 그 결과를 레지스터(438)에 저장하는 계산 유닛(436)에 의해 구현된다. 그러나, 다음 클록 사이클 상에서, 더미 계산 유닛(440)은 레지스터(438)에 저장된 값을 멀티플렉서(424)에 포워딩하고, 멀티플렉서는, 'z'의 값이 '2' 이하이면 레지스터(438)에 저장된 값을 레지스터(434)에 제공한다. 이 더미 계산 유닛(440)은 코드 경로(427) 내의 스테이지의 수를 밸런싱하기 위해 추가된다. 몇몇 구성에서, 레지스터(438)는 더미 계산 유닛(440)에 대한 'y'의 값을 유지하기 위해 추가된다.

[0044] 몇몇 구성에서, 주어진 코드 경로에 추가된 더미 계산 유닛의 수는 주어진 코드 경로의 스테이지의 수보다 적은 가장 긴 코드 경로 내의 스테이지의 수와 동일하다. 이 경우, 스테이지의 최대 수는 2-계산 유닛(428 및 432)이고, 코드 경로(435)는 하나의 스테이지(436)를 갖는다. 이와 같이, 하나의 계산 유닛(435)만 추가되어, 하나의 새로운 파이프라인 스테이지가 생성되게 한다.

[0045] 몇몇 실시예에서, (분기들의 밸런싱을 제외하고) 유용한 작업을 수행하지 않는 계산 유닛을 삽입하는 대신에, 값은 각각의 클록 사이클의 끝에서 레지스터에 단순히 저장된다. 몇몇 실시예에서, 더미 스테이지를 추가하는 대신에, 컴파일러는 더 깊은 파이프라인을 생성하고, 즉, 통상적으로 생성되는 것보다 더 많은 스테이지를 갖는 코드 경로를 생성한다. 이는 계산이 더 많은 스테이지에 걸쳐 확산됨에 따라 스테이지의 평균 실행 시간을 감소시키는 효과를 가질 수 있다.

[0046] 라인(416)은 단일문 "return z * y"를 포함하며, 이는 리턴 전의 'z'의 값과 'y'의 값을 곱한다. 이것은 레지스터(434)에 저장된 'y'의 값을 'z'의 값과 곱하는 계산 유닛(444)에 대응한다.

[0047] 도 5는 "do-while" 루프를 둘러싸는 재정렬 블록 구성자를 도시하는 하드웨어 아키텍처 도면(500)이다. 코드 리스팅(502)은 파라미터 'x'(506) 및 'y'(508) 를 취하는 함수 'f()'(504)를 포함한다. 라인(512)은 int 'z'를 "x+1"로 초기화하고, 실행 파이프라인(522)의 일부에 대응한다. 구체적으로, 실행 파이프라인(522)은 'x'(506)의 값에 '1'을 가산하고 이를 레지스터(535)에 'z'로서 저장하는 "+1" 계산 유닛(534)을 포함한다. 라인(513)은 루프 카운터 변수인 'i'를 '1'로 초기화한다. 파이프라인(522)은 값 'i', 'y' 및 'z'를 FIFO(524)로 푸시함으로써 종료한다.

[0048] 함수(504)의 재정렬 블록(516)은 "do-while" 루프(518)를 래핑한다. 도 5는 설계에 의해 스레드 실행 순서를 유지하지 않는 한 종류의 언어 구성자의 예로서 "do-while" 루프(518)를 사용한다. 그러나, 스레드 실행 순서를 유지하지 않는 다른 언어 구성자가 유사하게 고려된다. 재정렬 블록(516)은 파이프라인(522) 및 파이프라인(526)에 의해 부분적으로 구현된다. 파이프라인(522)은 스레드가 진입하는 순서대로 스레드를 등록하는 재정렬 시작 블록(536)을 포함한다.

[0049] 몇몇 구성에서, 스레드는 인덱스 또는 스레드 ID와 연관된다. 재정렬 시작 블록(536)에 진입하기 위한 제1 스레드의 인덱스/ID는 재정렬 블록의 제1 위치에, 예를 들어, 어레이의 요소 '0'에 저장될 것이다. 후속 스레드는 이들이 진입한 순서에 대응하는 요소에 저장된 인덱스/ID를 가질 것이다. 이러한 방식으로, 스레드 실행 순서의 레코드가 생성된다. 재정렬 블록(516)에 대한 논의는 "do-while" 루프(518)의 논의 이후에 계속된다.

[0050] "do-while" 루프(518)는 'z'번 루핑하고, 매번 '2'를 'y'에 가산한다. "do-while" 루프(518)는 파이프라인(526)에 의해 구현되지만, 몇몇 양상은 명확성을 위해 생략된다. "do-while" 루프(518)는 초기에 파이프라인(522)에 의해 FIFO(524)에 배치된 'z'의 값을 수신한다. 그 후 "do-while" 루프(518)는 블록(538)에서 'i'의 값과 'z'의 값을 비교할 수 있다. 'i'가 'z' 이하이면, 루프는 계속되고, 계산 유닛(540)은 'y'에 '2'를 가산한다. 그 다음에 업데이트된 'y'의 값, 'z'의 값 및 현재 'i'의 값이 FIFO(528)에 저장된다. 그러나, 'i'가 'z'보다 크면, 루프가 종료되고, 실행은 아래에 논의되는 바와 같이 재정렬 종료 블록(542)으로 넘어간다.

[0051] 초기 실행 이후, "do-while" 루프(518)는 계속 다른 'z-1'번을 반복한다. 각각의 반복에서, 현재 'y' 및 'i'의 값은 'z'의 값과 함께 FIFO(528)로부터 취출된다. 'x'는 함수의 나머지에서 사용되지 않기 때문에 이 시점에 유지되지 않는다는 것이 이해될 것이다. 비교(538)가 수행되고, 'i'가 여전히 'z' 미만이면, 실행은 전술한 바와 같이 계산 유닛(540)으로 계속된다. 그러나, 'i'가 'z'보다 크면, "do-while" "do-while" 루프(518)가 종료된다.

[0052] 스레드가 "do-while" "do-while" 루프(518)를 빠져나감에 따라, 재정렬 종료 블록(542)을 통과한다. 재정렬 시작 블록(536) 및 재정렬 종료 블록(542)이 별개로 도시되지만, 이들은 SDC 상의 단일 기능 유닛에 의해 구현될 수 있다. 재정렬 종료 블록(542)은 더 낮은 실행 순서를 갖는 모든 스레드(즉, 스레드 인덱스/ID의 어레이 내의 더 낮은 번호의 요소에 등록된 스레드)가 해제될 때까지 스레드를 차단한다. 가장 낮은 실행 순서를 갖는 스레드에 직면할 때, 즉시 해제되어 실행을 계속한다. 이러한 방식으로, 스레드는 자신들이 재정렬 블록(516)에 진입한 순서대로 다른 계산을 시작하기 위해 해제된다. 파이프라인(526)은 'z' 및 'y'의 값을 FIFO(530)에 저장함으로써 종료한다.

[0053] 라인(520)은 "do-while" 루프(518) 및 재정렬 블록(516)이 완료된 후에 실행되어, "z * y"의 값을 리턴한다. 라인(520)은 파이프라인(532)에 의해 부분적으로 구현되며, 이는 FIFO(530)로부터 'y' 및 'z'의 값을 판독하고, 계산 유닛(543)에 인가하여 'y'와 'z'를 곱한다.

[0054] 도 5는 "do-while" 루프를 실행함으로써 스레드 순서가 변경되고, 이어서 스레드 재정렬 블록에 의해 원래의 순서대로 복원될 수 있는 방법을 도시한다. 스레드(546A-D)는 순서(548)대로 "do-while" "do-while" 루프(518)와 같은 루프에 진입한다. 각각의 스레드는 'z'(544)의 상이한 값을 가지며, 따라서 각각의 스레드는 상이한 횟수의 반복으로 "do-while" 루프(518)를 수행한다. 구체적으로, 첫 번째로 진입하는 스레드(546A)는 6번 반복할 것이고, 두 번째로 진입하는 스레드(546B)는 8번 반복할 것이며, 세 번째로 진입하는 스레드(546C)는 2번 반복할 것이고, 네 번째로 진입하는 스레드(564D)는 4번 반복할 것이다. 스레드(546)는 재정렬 시작 블록(536)을 통과하여, 진입한 순서대로 스레드 인덱스/ID를 등록한다. 차트(550)는 클록 사이클에서 각각의 스레드가 얼마나 오래 실행되는지를 나타낸다. 스레드(546)는 상이한 순서로 "do-while" 루프(518)를 빠져나간다 - 스레드(546A)는 세 번째로 빠져나가고, 스레드(546B)는 네 번째로 빠져나가며, 스레드(546B)는 첫 번째로 빠져나가고, 스레드(546A)는 두 번째로 빠져나간다.

[0055] 스레드(546C)가 먼저 "do-while" 루프(518)를 빠져나가는 동안, 재정렬 종료 블록(542)은 스레드(546A 및 546B)가 "do-while" 루프를 종료하고 실행을 재개할 때까지 추가 실행을 지연시킨다. 다음으로, 스레드(546D)는 "do-while" 루프(518)를 종료하고, 또한 스레드(546A-C)가 종료할 때까지 차단된다. 다음으로, 스레드(546A)는 "do-while" 루프(518)를 종료하지만, 546A는 재정렬 시작 블록(536)에 등록할 제1 스레드였기 때문에, 재개하도록 허용된다. 동시에, 재정렬 종료 블록(542)은 스레드(546A)가 재개되었으므로 임의의 후속 스레드가 계속되도록 허용되는지를 체크한다. 이 경우, 스레드(546B)가 아직 끝나지 않았기 때문에, 존재하지 않는다. 그러나, 일단 스레드(546B)가 "do-while" 루프(518)를 끝내면, 재정렬 종료 블록(542)은 더 낮은 초기 실행 순서를 갖는 모든 스레드(즉, 스레드(546A))가 실행을 재개했으므로, 스레드(546B)가 실행을 재개하도록 허용된다는 것을 주목한다. 다음으로, 스레드(546B) 이후에 재정렬 시작 블록(536)에 진입한 스레드는 이들이 재개하도록 허용되는지를 판정하기 위해 체크된다. 이 예에서, 스레드(546C 및 546D) 양자 모두는 차단되고, 따라서 이들 모두가 (그 순서대로) 재개하도록 허용되는데, 이는 이들보다 먼저 도착한 스레드들 모두가 재개되도록 허용되었기 때문이다. 이러한 방식으로, 스레드(546)는 순서(554)대로, 즉, 그들이 재정렬 시작 블록(536)에 진입한 것과 동일한 순서대로 재정렬 종료 블록(542)을 나간다. 도 4 및 도 5 및 대응하는 논의는 비제한적인 예를 도시한다. 다른 유형의 루프, 파이프라인, 식, 분기 및 순서화가 유사하게 고려된다.

[0056] 도 6은 본 명세서에 개시된 일 실시예에 따른, 도 1 내지 도 5에 예시되고 전술된 스레드 실행 순서를 유지하는 언어 및 컴파일러의 양상을 도시하는 루틴(600)을 나타내는 흐름도이다. 도 6 및 다른 도면과 관련하여 본 명세서에 설명된 논리적 동작이 (1) 컴퓨팅 디바이스 상에서 실행되는 일련의 컴퓨터로 구현된 액트 또는 프로그램 모듈로서 및/또는 (2) 컴퓨팅 디바이스 내의 상호접속된 머신 로직 회로 또는 회로 모듈로서 구현될 수 있다는 것을 알아야 한다.

[0057] 본 명세서에 개시된 기술의 특정 구현은 컴퓨팅 디바이스의 성능 및 다른 요구사항에 의존하는 선택의 문제이다. 따라서, 본 명세서에 설명된 논리적 동작은 상태, 동작, 구조적 디바이스, 액트, 또는 모듈로서 다양하게 지칭된다. 이러한 상태, 동작, 구조적 디바이스, 액트 및 모듈은 하드웨어, 소프트웨어, 펌웨어, 특수 목적 디지털 로직, 및 이들의 임의의 조합으로 구현될 수 있다. 도면에 도시되고 본 명세서에 설명된 것보다 더

많거나 더 적은 동작이 수행될 수 있다는 것을 알아야 한다. 이들 동작은 또한 본 명세서에 설명된 것과 상이한 순서로 수행될 수 있다.

[0058] 루틴(600)은 소스 코드(102)가 컴파일러(104)에 의해 수신되는 동작(602)에서 시작한다. 소스 코드(102)는 멀티-스레딩된 프로그래밍 언어로 표현될 수 있다. 몇몇 구성에서, 소스 코드(102)는 명령형 프로그래밍 언어로 표현된다. 소스 코드는, 스레드로 하여금 복수의 코드 경로(회로에 설명된 코드 경로와 구별하기 위해 "소스 코드 경로"로도 지칭됨) 중 하나 상에서 실행하게 하는, 분기문(411), 예컨대, 'if', 'switch', 'case' 또는 'while' 문을 포함할 수 있다. 몇몇 구성에서, 소스 코드는 "do-while" 루프(518)와 같은 스레드 실행 순서를 유지하지 않는 코드를 래핑하는 재정렬 블록(516)을 포함한다.

[0059] 동작(602)으로부터, 루틴은 컴파일러(104)가 소스 코드(102)를 회로 디스크립션(106)으로 컴파일하는 동작(604)으로 진행한다. 몇몇 구성에서, 회로 디스크립션(106)은 복수의 소스 코드 경로와 연관된 복수의 코드 경로(427, 435)를 포함하고, 컴파일러(104)는 더미 파이프라인 스테이지(440)를 복수의 코드 경로의 일부 또는 전부에 부가하여, 복수의 코드 경로 중 일부 또는 전부가 동일한 수의 파이프라인 스테이지(324)를 갖는다.

[0060] 몇몇 구성에서, 회로 디스크립션은 제1 실행 파이프라인(200A) 및 제2 실행 파이프라인(200B)을 포함하고, 제1 실행 파이프라인(20A)은 로컬 변수 세트를 FIFO(202A)에 푸시함으로써 실행을 제2 실행 파이프라인(200B)으로 핸드오프한다. 그 다음에, 제2 실행 파이프라인(200B)은 FIFO(202A)로부터 로컬 변수 세트를 그들이 푸시된 순서대로 취출할 수 있고, 이에 의해 스레드 실행 순서를 유지한다.

[0061] 몇몇 구성에서, 회로 디스크립션(521) 스레드 실행 순서를 유지하지 않는 구성자(518)의 회로 디스크립션을 래핑하는 재정렬 버퍼(536, 542)를 포함한다. 이들 구성에서, 재정렬 버퍼(536, 542)는 구성자가 실행되기 전에 스레드 순서(548)를 등록하고, 구성자를 떠날 때 스레드를 원래의 실행 순서(548)로 리턴한다. 몇몇 구성에서, 재정렬 버퍼(536, 542)는 더 낮은 실행 순서(즉, 실행할 제1 스레드에 더 가까움)를 갖는 스레드들 모두가 실행을 재개할 때까지 스레드 실행을 차단한다.

[0062] 동작(604)으로부터, 루틴(600)은, 회로 디스크립션(예를 들어, HDL 코드)이 회로 디스크립션(106)에 의해 정의된 회로 구현을 포함하는 SDL(112)을 생성하는 데 이용되는 동작(606)으로 진행한다. 루틴(600)은 그 후 동작(606)에서 종료되는 동작(608)으로 진행한다.

[0063] 도 7은 본 명세서에 제시된 다양한 기술을 구현할 수 있는 컴퓨팅 디바이스에 대한 예시적인 컴퓨터 하드웨어 및 소프트웨어 아키텍처를 도시하는 컴퓨터 아키텍처 도면이다. 특히, 도 7에 도시된 아키텍처는 서버 컴퓨터, 모바일폰, e-리더, 스마트폰, 데스크톱 컴퓨터, AR/VR 디바이스, 태블릿 컴퓨터, 랩톱 컴퓨터, 또는 다른 유형의 컴퓨팅 디바이스를 구현하는 데 이용될 수 있다.

[0064] 도 7에 도시된 컴퓨터(700)는 중앙 처리 장치("CPU")(702), 랜덤 액세스 메모리(706)("RAM") 및 판독 전용 메모리("ROM")(708)를 포함하는 시스템 메모리(604), 및 메모리(704)를 CPU(702)에 연결하는 시스템 버스(710)를 포함한다. 예컨대, 시동 중에, 컴퓨터(700) 내의 구성요소 사이의 정보 전송을 돕는 기본 루틴을 포함하는 기본 입출력 시스템("BIOS" 또는 "펌웨어")이 ROM(708)에 저장될 수 있다. 컴퓨터(700)는 운영 체제(722), 애플리케이션 프로그램, 및 다른 유형의 프로그램을 저장하는 대용량 저장 디바이스(712)를 더 포함한다. 대용량 저장 디바이스(712)는 또한 다른 유형의 프로그램 및 데이터를 저장하도록 구성될 수 있다.

[0065] 대용량 저장 디바이스(712)는 버스(710)에 접속된 대용량 저장 제어기(도시 생략)를 통해 CPU(702)에 접속된다. 대용량 저장 디바이스(712) 및 그와 연관된 컴퓨터 판독가능 매체는 컴퓨터(700)용 비휘발성 저장부를 제공한다. 본 명세서에 포함된 컴퓨터 판독가능 매체의 설명은 하드 디스크, CD-ROM 드라이브, DVD-ROM 드라이브, 또는 USB 저장 키와 같은 대용량 저장 디바이스를 지칭하지만, 컴퓨터 판독가능 매체는 컴퓨터(700)에 의해 액세스될 수 있는 임의의 이용가능한 컴퓨터 저장 매체 또는 통신 매체일 수 있다는 것이 당업자에 의해 인식되어야 한다.

[0066] 통신 매체는 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈, 또는 반송파 또는 다른 전송 메커니즘과 같은 변조된 데이터 신호 내의 다른 데이터를 포함하고, 임의의 전달 매체를 포함한다. "변조된 데이터 신호"라는 용어는 신호 내에 정보를 인코딩하도록 하는 방식으로 그 특성 중 하나 이상이 변경되거나 또는 설정된 신호를 의미한다. 제한이 아닌 예로서, 통신 매체는 유선 네트워크 또는 직접 유선 접속과 같은 유선 매체, 및 음향, 무선 주파수, 적외선 및 다른 무선 매체와 같은 무선 매체를 포함한다. 상기 중 임의의 것의 조합도 컴퓨터 판독가능 매체의 범위 내에 포함되어야 한다.

[0067] 제한이 아닌 예로서, 컴퓨터 저장 매체는 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 또는 다른 데이

터와 같은 정보의 저장을 위한 임의의 방법 또는 기술로 구현되는 휘발성 및 비휘발성, 착탈식 및 비착탈식 매체를 포함할 수 있다. 예를 들어, 컴퓨터 저장 매체는 RAM, ROM, EPROM, EEPROM, 플래시 메모리 또는 다른 고체 상태 메모리 기술, CD-ROM, DVD(digital versatile disk), HD-DVD, BLU-RAY, 또는 다른 광학 저장장치, 자기 카세트, 자기 테이프, 자기 디스크 저장장치 또는 다른 자기 저장 디바이스, 또는 원하는 정보를 저장하는 데 사용될 수 있고 컴퓨터(700)에 의해 액세스될 수 있는 임의의 다른 매체를 포함하지만, 이에 제한되지 않는다. 청구범위의 목적을 위해, "컴퓨터 저장 매체"라는 문구 및 이의 변형은 파 또는 신호 자체 또는 통신 매체를 포함하지 않는다.

[0068] 다양한 구성에 따르면, 컴퓨터(700)는 네트워크(720)와 같은 네트워크를 통해 원격 컴퓨터로의 논리적 접속을 사용하여 네트워킹된 환경에서 동작할 수 있다. 컴퓨터(700)는 버스(710)에 접속된 네트워크 인터페이스 유닛(716)을 통해 네트워크(720)에 접속될 수 있다. 네트워크 인터페이스 유닛(716)은 또한 다른 유형의 네트워크 및 원격 컴퓨터 시스템에 접속하는 데 이용될 수 있다는 것을 알아야 한다. 컴퓨터(700)는 키보드, 마우스, 터치 입력, 전자 스타일러스(도 7에 도시되지 않음), 또는 비디오 카메라와 같은 물리적 센서를 포함하는 다수의 다른 디바이스로부터 입력을 수신하고 처리하기 위한 입출력 제어기(718)도 포함할 수 있다. 유사하게, 입출력 제어기(718)는 디스플레이 스크린 또는 다른 유형의 출력 디바이스(도 7에 또한 도시되지 않음)에 출력을 제공할 수 있다.

[0069] 본 명세서에 설명된 소프트웨어 컴포넌트는, CPU(702)에 로딩되고 실행될 때, 범용 컴퓨팅 디바이스로부터 본 명세서에 제시된 기능을 용이하게 하도록 맞춤화된 특수 목적 컴퓨팅 디바이스로 CPU(802) 및 전체 컴퓨터(700)를 변환할 수 있다는 것을 알아야 한다. CPU(702)는 임의의 개수의 트랜지스터 또는 다른 별개의 회로 요소로 구성될 수 있으며, 이들은 개별적으로 또는 집합적으로 임의의 수의 상태를 가정할 수 있다. 보다 구체적으로, CPU(702)는 본 명세서에 개시된 소프트웨어 모듈 내에 포함된 실행가능 명령어에 응답하여 유한 상태 머신으로서 동작할 수 있다. 이들 컴퓨터 실행가능 명령어는 CPU(702)가 상태들 사이에서 어떻게 전이하는지를 지정함으로써 CPU(702)를 변환할 수 있고, 이에 의해 CPU(702)를 구성하는 트랜지스터 또는 다른 별개의 하드웨어 요소를 변환한다.

[0070] 본 명세서에 제시된 소프트웨어 모듈을 인코딩하는 것은 또한 본 명세서에 제시된 컴퓨터 판독가능 매체의 물리적 구조를 변환할 수 있다. 물리적 구조의 특정 변환은 본 설명의 상이한 구현에서 다양한 요인에 의존한다. 이러한 요인의 예는 컴퓨터 판독가능 매체를 구현하는 데 사용되는 기술, 컴퓨터 판독가능 매체가 1차 또는 2차 저장장치로서 특성화되는지 여부 등을 포함하지만, 이에 제한되지 않는다. 예를 들어, 컴퓨터 판독가능 매체가 반도체 기반 메모리로서 구현되면, 본 명세서에 개시된 소프트웨어는 반도체 메모리의 물리적 상태를 변환함으로써 컴퓨터 판독가능 매체 상에 인코딩될 수 있다. 예를 들어, 소프트웨어는 트랜지스터, 캐패시터, 또는 반도체 메모리를 구성하는 다른 별개의 회로 요소의 상태를 변환할 수 있다. 소프트웨어는 또한 데이터를 저장하기 위해 이러한 컴포넌트의 물리적 상태를 변환할 수 있다.

[0071] 다른 예로서, 본 명세서에 개시된 컴퓨터 판독가능 매체는 자기 또는 광학 기술을 사용하여 구현될 수 있다. 이러한 구현에서, 본 명세서에 제시된 소프트웨어는, 소프트웨어가 내부에 인코딩될 때, 자기 또는 광학 매체의 물리적 상태를 변환할 수 있다. 이러한 변환은 주어진 자기 매체 내의 특정 위치의 자기 특성을 변경하는 것을 포함할 수 있다. 이들 변환은 또한 주어진 광학 매체 내의 특정 위치의 물리적 특징 또는 특성을 변경하여 이들 위치의 광학 특성을 변경하는 것을 포함할 수 있다. 본 설명의 범위 및 사상으로부터 벗어나지 않으면서 물리적 매체의 다른 변환이 가능하며, 전술한 예들은 이 논의를 용이하게 하기 위해서만 제공된다.

[0072] 위의 내용을 고려하여, 본 명세서에 제시된 소프트웨어 컴포넌트를 저장하고 실행하기 위해 컴퓨터(700)에서 다수의 유형의 물리적 변환이 발생한다는 것을 알아야 한다. 컴퓨터(700)에 대해 도 7에 도시된 아키텍처, 또는 유사한 아키텍처가 휴대용 컴퓨터, 비디오 게임 디바이스, 임베디드 컴퓨터 시스템, 스마트폰, 태블릿 및 AR/VR 디바이스와 같은 모바일 디바이스, 및 당업자에게 공지된 다른 유형의 컴퓨팅 디바이스를 포함하는 다른 유형의 컴퓨팅 디바이스를 구현하는 데 이용될 수 있다는 것도 이해해야 한다. 컴퓨터(700)는 도 7에 도시된 모든 컴포넌트를 포함하지 않을 수도 있거나, 도 7에서 명시적으로 도시되지 않은 다른 컴포넌트를 포함할 수 있거나, 또는 도 7에 도시된 것과 완전히 상이한 아키텍처를 이용할 수 있다는 것도 고려된다.

[0073] 도 8은 본 명세서에 제시된 다양한 실시예에 따라, 개시된 기술의 양상이 구현될 수 있는 분산 네트워크 컴퓨팅 환경(800)을 도시하는 네트워크 도면이다. 도 8에 도시된 바와 같이, 하나 이상의 서버 컴퓨터(800A)는 통신 네트워크(720)(고정-와이어 또는 무선 LAN, WAN, 인트라넷, 엑스트라넷, 피어-투-피어 네트워크, 가상 사설 네트워크, 인터넷, 블루투스 통신 네트워크, 독점적인 저전압 통신 네트워크 또는 다른 통신 네트워크 중 어느 하나

또는 이들의 조합일 수 있음)를 통해, 태블릿 컴퓨터(800B), 게임 콘솔(800C), 스마트 워치(800D), 스마트폰과 같은 전화(800E), 개인용 컴퓨터(800F) 및 AR/VR 디바이스(800G)와 같은, 그러나 이에 한정되지 않는 다수의 클라이언트 컴퓨팅 디바이스와 상호접속될 수 있다.

[0074] 통신 네트워크(720)가 예를 들어, 인터넷인 네트워크 환경에서, 서버 컴퓨터(800A)는 하이퍼텍스트 전송 프로토콜(hypertext transfer protocol, "HTTP"), 파일 전송 프로토콜(file transfer protocol, "FTP"), 또는 간단한 객체 액세스 프로토콜(simple object access protocol, "SOAP")과 같은 다수의 알려진 프로토콜 중 임의의 것을 통해 클라이언트 컴퓨팅 디바이스(800b-800g)로 및 이로부터 데이터를 처리하고 통신하도록 동작가능한 전용 서버 컴퓨터일 수 있다. 또한, 네트워킹된 컴퓨팅 환경(800)은 SSL(secured socket layer) 또는 PGP(pretty good privacy)와 같은 다양한 데이터 보안 프로토콜을 이용할 수 있다. 클라이언트 컴퓨팅 디바이스(800B-800G)의 각각은 서버 컴퓨터(800A)에 대한 액세스를 얻기 위해 웹 브라우저(도 8에 도시되지 않음), 또는 다른 그래픽 사용자 인터페이스(도 8에 도시되지 않음), 또는 모바일 데스크톱 환경(도 8에 도시되지 않음)과 같은 하나 이상의 컴퓨팅 애플리케이션 또는 단말 세션을 지원하도록 동작가능한 운영 체제를 구비할 수 있다.

[0075] 서버 컴퓨터(800A)는 다른 컴퓨팅 환경(도 8에 도시되지 않음)에 통신가능하게 연결될 수 있고 참여 사용자의 상호작용/리소스 네트워크에 관한 데이터를 수신할 수 있다. 예시적인 동작에서, 사용자(도 8에 도시되지 않음)는 원하는 데이터를 획득하고/하거나 다른 컴퓨팅 애플리케이션을 수행하기 위해 클라이언트 컴퓨팅 디바이스(800B-800G) 상에서 실행되는 컴퓨팅 애플리케이션과 상호작용할 수 있다.

[0076] 데이터 및/또는 컴퓨팅 애플리케이션은 서버(800A) 또는 서버들(800A) 상에 저장될 수 있고, 예시적인 통신 네트워크(720)를 통해 클라이언트 컴퓨팅 디바이스(800B-800G)를 통해 협력 사용자에게 전달될 수 있다. 참여 사용자(도 8에 도시되지 않음)는 서버 컴퓨터(800A) 상에 전체적으로 또는 부분적으로 하우징된 특정 데이터 및 애플리케이션에 대한 액세스를 요청할 수 있다. 이들 데이터는 프로세싱 및 저장을 위해 클라이언트 컴퓨팅 디바이스(800B-800G)과 서버 컴퓨터(800A) 사이에서 통신될 수 있다.

[0077] 서버 컴퓨터(800A)는 데이터 및 애플리케이션의 생성, 인증, 암호화, 및 통신을 위해 컴퓨팅 애플리케이션, 프로세스 및 애플릿을 호스팅할 수 있고, 다른 서버 컴퓨팅 환경(도 8에 도시되지 않음), 제3자 서비스 제공자(도 8에 도시되지 않음), 네트워크 부착 저장장치("NAS") 및 저장 영역 네트워크("SAN")와 협력하여 애플리케이션/데이터 트랜잭션을 실현할 수 있다.

[0078] 도 7에 도시된 컴퓨팅 아키텍처 및 도 8에 도시된 분산 네트워크 컴퓨팅 환경이 논의의 편의를 위해 단순화되었다는 것을 알아야 한다. 컴퓨팅 아키텍처 및 분산 컴퓨팅 네트워크는 더 많은 컴퓨팅 컴포넌트, 디바이스, 소프트웨어 프로그램, 네트워킹 디바이스, 및 본 명세서에 구체적으로 설명되지 않은 다른 컴포넌트를 포함하고 이용할 수 있다는 것도 이해해야 한다.

[0079] 본 명세서에 제시된 개시내용은 또한 하기 조항에 제시된 출원 대상을 포함한다:

[0080] 조항 1: 컴퓨터로 구현된 방법으로서, 멀티-스레딩된 프로그래밍 언어로 표현된 소스 코드를 수신하는 단계 - 소스 코드는 복수의 소스 코드 경로 중 하나로 실행을 지시하는 분기문을 포함함 - 와, 파이프라인을 포함하는 회로 디스크립션(circuit description)으로 상기 소스 코드를 컴파일하는 단계 - 상기 파이프라인은 상기 복수의 소스 코드 경로와 연관된 복수의 코드 경로를 포함하고, 하나 이상의 파이프라인 스테이지가 상기 복수의 코드 경로 중 하나 이상에 추가되어 상기 복수의 코드 경로가 동일한 수의 파이프라인 스테이지를 가짐 - 와, 상기 회로 디스크립션에 기초하여, 회로 구현을 포함하는 동기식 디지털 회로를 생성하는 단계를 포함하는, 컴퓨터로 구현된 방법.

[0081] 조항 2: 조항 1에 있어서, 복수의 스레드가 제1 순서대로 상기 파이프라인에 진입하고, 상기 복수의 스레드가 상기 제1 순서대로 상기 파이프라인을 떠나는, 컴퓨터로 구현된 방법.

[0082] 조항 3: 조항 1 또는 조항 2에 있어서, 상기 복수의 코드 경로 중 하나 이상에 파이프라인 스테이지를 추가하는 것은, 최장 코드 경로 내의 파이프라인 스테이지의 수를 결정하는 것 및 상기 최장 코드 경로 내의 파이프라인 스테이지의 수에서 각각의 코드 경로 내의 파이프라인 스테이지의 수를 뺀 것과 동일한 수의 파이프라인 스테이지를 상기 각각의 코드 경로에 추가하는 것을 포함하는, 컴퓨터로 구현된 방법.

[0083] 조항 4: 조항 1 내지 조항 3 중 어느 한 조항에 있어서, 상기 파이프라인은 제1 파이프라인을 포함하고, 상기 회로 디스크립션은 제2 파이프라인을 포함하며, 상기 제1 파이프라인을 실행하는 스레드는 로컬 변수를 선입선출 큐(first-in-first-out queue)로 푸시함으로써 상기 제2 파이프라인으로 실행을 이송하고, 상기 제2 파이프라인은 상기 선입선출 큐로부터 로컬 변수를 상기 로컬 변수가 푸시되었던 순서대로 판독함으로써 파이프라인에

걸쳐 스레드 실행 순서를 유지하는, 컴퓨터로 구현된 방법.

[0084] 조항 5: 조항 1 내지 조항 4 중 어느 한 조항에 있어서, 상기 소스 코드는 스레드 실행 순서를 유지하지 않는 프로그래밍 구성자를 래핑(wrap)하는 재정렬 블록 구성자를 포함하고, 상기 재정렬 블록 구성자는 회로 구현에 매핑하며, 상기 회로 구현은, 인입 스레드 실행 순서를 기록하고, 스레드가 스레드 실행 순서를 유지하지 않는 상기 구성자를 실행할 수 있게 하며, 모든 하위 순서의 스레드가 재개될 때까지 스레드를 재개하는 것을 차단하는, 컴퓨터로 구현된 방법.

[0085] 조항 6: 조항 1 내지 조항 5 중 어느 한 조항에 있어서, 스레드는 실행을 위해 상기 파이프라인에 제공되는 로컬 변수의 집합을 포함하는, 컴퓨터로 구현된 방법.

[0086] 조항 7: 조항 1 내지 조항 6 중 어느 한 조항에 있어서, 상기 파이프라인은 순서대로 실행되는 스테이지를 포함하고, 복수의 스레드는 상기 순서에서 상기 스테이지를 통해 흐름으로써 실행 순서를 유지하는, 컴퓨터로 구현된 방법.

[0087] 조항 8: 컴퓨팅 디바이스로서, 하나 이상의 프로세서와, 컴퓨터 실행가능 명령어가 저장된 적어도 하나의 컴퓨터 저장 매체를 포함하되, 컴퓨터 실행가능 명령어는, 상기 하나 이상의 프로세서에 의해 실행될 때, 상기 컴퓨팅 디바이스로 하여금, 멀티-스레딩된 프로그래밍 언어로 표현된 소스 코드를 수신하게 하고, 제1 파이프라인, 제2 파이프라인, 및 상기 제1 파이프라인으로부터 상기 제2 파이프라인으로 전달되는 로컬 스레드 변수의 세트를 저장하는 선입선출(FIFO) 큐를 포함하는 회로 디스크립션으로 상기 소스 코드를 컴파일하게 하며 - 상기 제1 파이프라인은 스레드 실행 순서대로 상기 FIFO 큐에 로컬 스레드 변수의 세트를 저장하고, 상기 제2 파이프라인은 상기 스레드 실행 순서에서 상기 FIFO 큐로부터 로컬 스레드 변수의 세트를 취출함으로써 상기 스레드 실행 순서를 유지함 - , 상기 회로 디스크립션에 기초하여, 회로 구현을 포함하는 동기식 디지털 회로를 생성하게 하는, 컴퓨팅 디바이스.

[0088] 조항 9: 조항 8에 있어서, 상기 소스 코드는 복수의 소스 코드 경로 중 하나로 실행을 지시하는 분기문을 포함하고, 상기 제1 파이프라인은 상기 복수의 소스 코드 경로와 연관된 복수의 코드 경로를 포함하고, 하나 이상의 파이프라인 스테이지가 상기 복수의 코드 경로 중 하나 이상에 추가되어 상기 복수의 코드 경로가 동일한 수의 파이프라인 스테이지를 갖는, 컴퓨팅 디바이스.

[0089] 조항 10: 조항 8 또는 조항 9에 있어서, 상기 복수의 코드 경로 중 하나 이상에 파이프라인 스테이지를 추가하는 것은, 최장 코드 경로 내의 파이프라인 스테이지의 수를 결정하는 것 및 상기 최장 코드 경로 내의 파이프라인 스테이지의 수에서 각각의 코드 경로 내의 파이프라인 스테이지의 수를 뺀 것과 동일한 수의 파이프라인 스테이지를 상기 각각의 코드 경로에 추가하는 것을 포함하는, 컴퓨팅 디바이스.

[0090] 조항 11: 조항 8 내지 조항 10 중 어느 한 조항에 있어서, 상기 소스 코드는 스레드 실행 순서를 유지하지 않는 프로그래밍 구성자를 래핑하는 재정렬 블록 구성자를 포함하고, 상기 재정렬 블록 구성자는 회로 구현에 매핑하며, 상기 회로 구현은, 인입 스레드 실행 순서를 기록하고, 스레드가 스레드 실행 순서를 유지하지 않는 구성자를 실행할 수 있게 하며, 모든 하위 순서의 스레드가 재개될 때까지 스레드를 재개하는 것을 차단하는, 컴퓨팅 디바이스.

[0091] 조항 12: 조항 8 내지 조항 11 중 어느 한 조항에 있어서, 스레드는 진입한 순서대로 상기 재정렬 블록을 떠나는, 컴퓨팅 디바이스.

[0092] 조항 13: 조항 8 내지 조항 12 중 어느 한 조항에 있어서, 스레드는 실행을 위해 상기 제1 파이프라인에 제공되는 로컬 스레드 변수의 집합을 포함하는, 컴퓨팅 디바이스.

[0093] 조항 14: 조항 8 내지 조항 13 중 어느 한 조항에 있어서, 상기 제1 파이프라인은 순서대로 실행되는 스테이지를 포함하고, 복수의 스레드는 상기 순서에서 상기 스테이지를 통해 흐름으로써 실행 순서를 유지하는, 컴퓨팅 디바이스.

[0094] 조항 15: 컴퓨터 실행가능 명령어가 저장된 적어도 하나의 컴퓨터 저장 매체로서, 상기 컴퓨터 실행가능 명령어는, 하나 이상의 프로세서에 의해 실행될 때, 컴퓨팅 디바이스로 하여금, 멀티-스레딩된 프로그래밍 언어로 표현된 소스 코드를 수신 - 상기 소스 코드는 회로 구현에 매핑하는 구성자를 포함하고, 상기 구성자는 재정렬 블록 및 스레드 실행 순서를 유지하지 않는 구성자를 포함하며, 상기 회로 구현은, 복수의 스레드가 수신되는 순서대로 스레드 식별자를 등록하는 재정렬 버퍼와, 상기 복수의 스레드의 각각에 대해 미지의 수의 클록 사이클에 대해 실행하는 회로를 포함하고, 상기 재정렬 버퍼는 더 낮은 실행 순서를 갖는 모든 스레드가 재개될 때까

지 스레드가 재개하는 것을 차단함 - 하게 하고, 상기 구성자를 회로 디스크립션으로 컴파일하게 하며, 상기 회로 디스크립션에 기초하여, 상기 회로 구현을 포함하는 동기식 디지털 회로를 생성하게 하는, 적어도 하나의 컴퓨터 저장 매체.

[0095] 조항 16: 조항 15에 있어서, 상기 소스 코드는 복수의 소스 코드 경로 중 하나로 실행을 지시하는 분기문을 포함하고, 상기 회로 디스크립션은 복수의 소스 코드 경로를 포함하는 파이프라인을 포함하고, 하나 이상의 파이프라인 스테이지가 상기 복수의 코드 경로 중 하나 이상에 추가되어 상기 복수의 코드 경로가 동일한 수의 파이프라인 스테이지를 갖는, 적어도 하나의 컴퓨터 저장 매체.

[0096] 조항 17: 조항 15 또는 조항 16에 있어서, 상기 복수의 코드 경로 중 하나 이상에 파이프라인 스테이지를 추가하는 것은, 최장 코드 경로 내의 파이프라인 스테이지의 수를 결정하는 것 및 상기 최장 코드 경로 내의 파이프라인 스테이지의 수에서 각각의 코드 경로 내의 파이프라인 스테이지의 수를 뺀 것과 동일한 수의 파이프라인 스테이지를 상기 각각의 코드 경로에 추가하는 것을 포함하는, 적어도 하나의 컴퓨터 저장 매체.

[0097] 조항 18: 조항 15 내지 조항 17 중 어느 한 조항에 있어서, 스레드는 실행을 위해 상기 제1 파이프라인에 제공되는 로컬 스레드 변수의 집합을 포함하는, 적어도 하나의 컴퓨터 저장 매체.

[0098] 조항 19: 조항 15 내지 조항 17 중 어느 한 조항에 있어서, 상기 제1 파이프라인은 순서대로 실행되는 스테이지를 포함하고, 복수의 스레드는 상기 순서에서 상기 스테이지를 통해 흐름으로써 실행 순서를 유지하는, 적어도 하나의 컴퓨터 저장 매체.

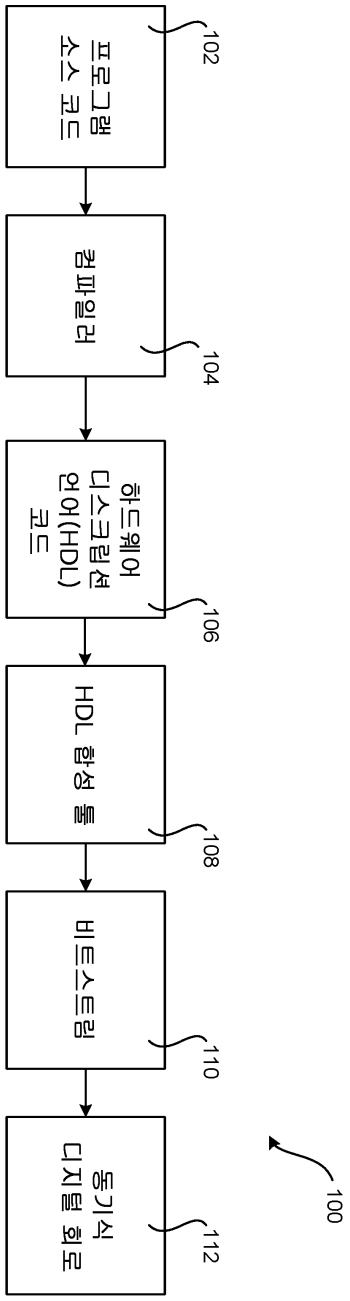
[0099] 조항 20: 조항 15 내지 조항 19 중 어느 한 조항에 있어서, 상기 파이프라인은 제1 파이프라인을 포함하고, 상기 회로 디스크립션은 제2 파이프라인을 포함하며, 상기 제1 파이프라인을 실행하는 스레드는 로컬 변수를 선입선출 큐(first-in-first-out queue)로 푸시함으로써 상기 제2 파이프라인으로 실행을 이송하고, 상기 제2 파이프라인은 상기 선입선출 큐로부터 로컬 변수를 상기 로컬 변수가 푸시되었던 순서대로 판독함으로써 파이프라인에 걸쳐 스레드 실행 순서를 유지하는, 적어도 하나의 컴퓨터 저장 매체.

[0100] 전술한 바에 기초하여, 스레드 실행 순서를 유지하는 언어 및 컴파일러가 본 명세서에 개시되었다는 것을 알아야 한다. 본 명세서에 제시된 출원 대상이 컴퓨터 구조적 특징, 방법적 및 변환적 동작, 특정 컴퓨팅 머신, 및 컴퓨터 판독가능 매체에 특정한 언어로 설명되었지만, 첨부된 청구범위에 제시된 출원 대상은 반드시 본 명세서에 설명된 특정 특징, 동작, 또는 매체에 제한되는 것은 아니라는 것을 이해해야 한다. 오히려, 특정 특징, 동작 및 매체는 청구된 출원 대상을 구현하는 예시적인 형태로서 개시된다.

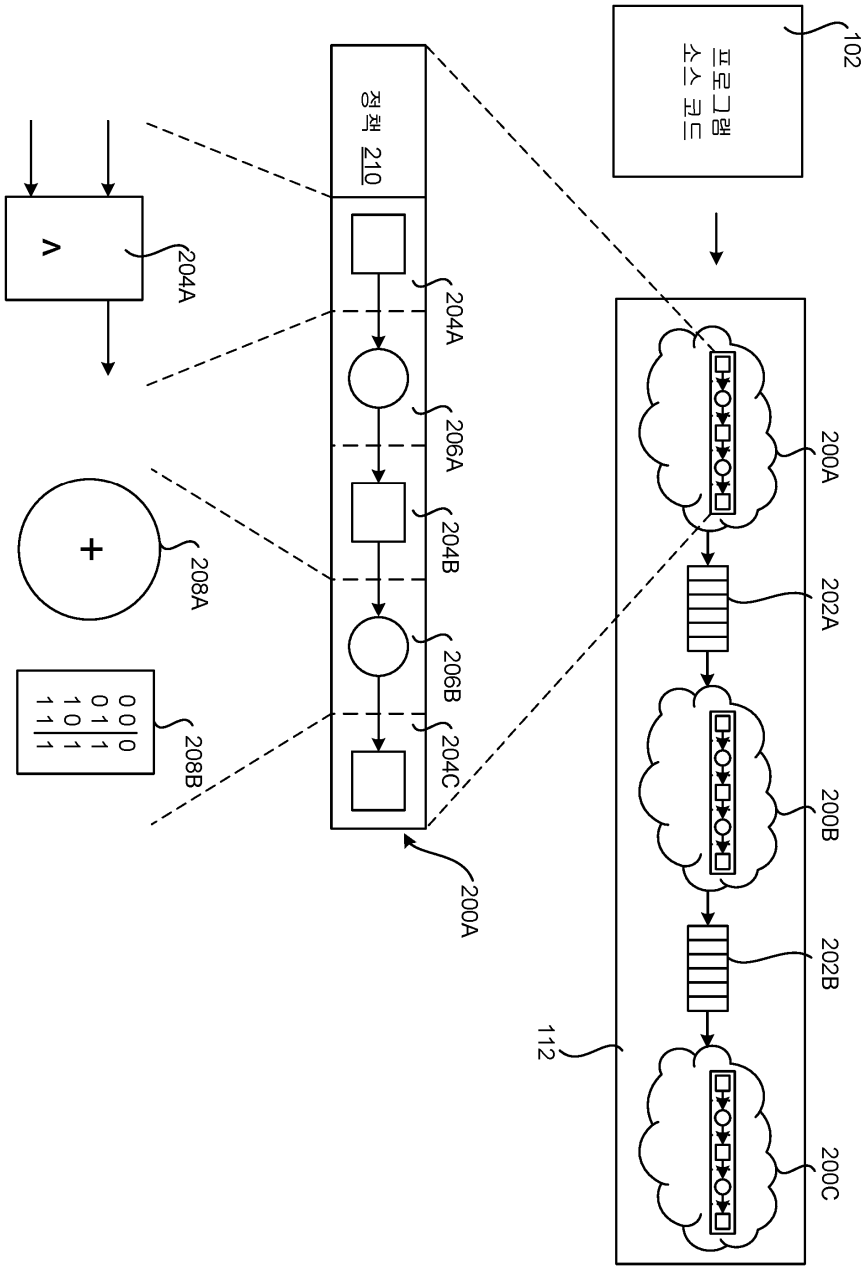
[0101] 앞에서 설명된 출원 대상은 예로써만 제공되고 제한하는 것으로 해석되지 않아야 한다. 도시되고 설명된 예시적인 구성 및 응용례를 따르지 않고, 다음의 청구범위에 설명되는 본 개시의 범위를 벗어나지 않으면서, 본 명세서에 설명된 출원 대상에 다양한 수정 및 변경이 이루어질 수 있다.

도면

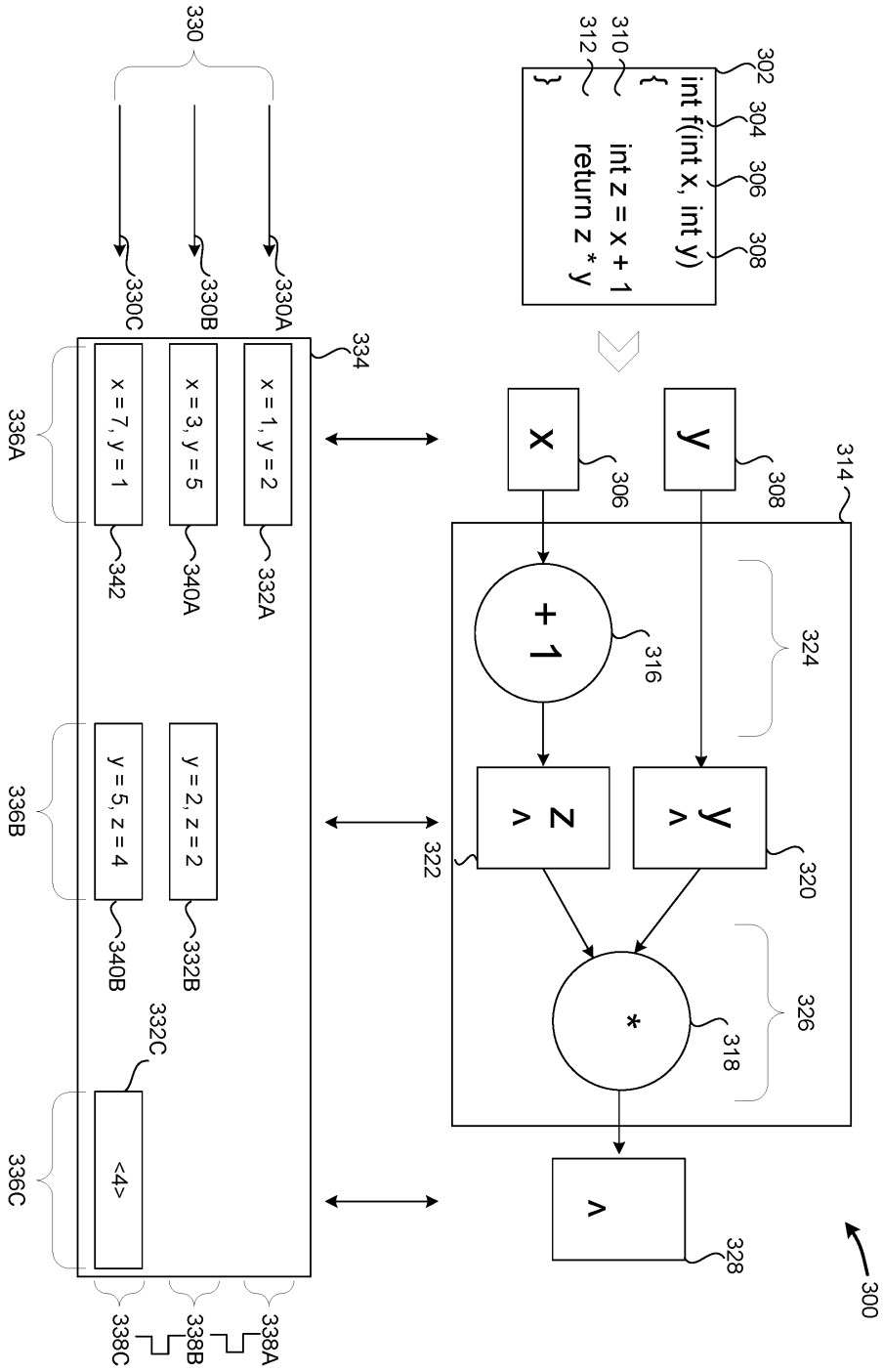
도면1


프로그램 소스 코드
102
컴파일러
104
하드웨어 디스크립션 언어(HDL) 코드
106
HDL 합성 툴
108
비트스트림
110
동기식 디지털 회로
112
100

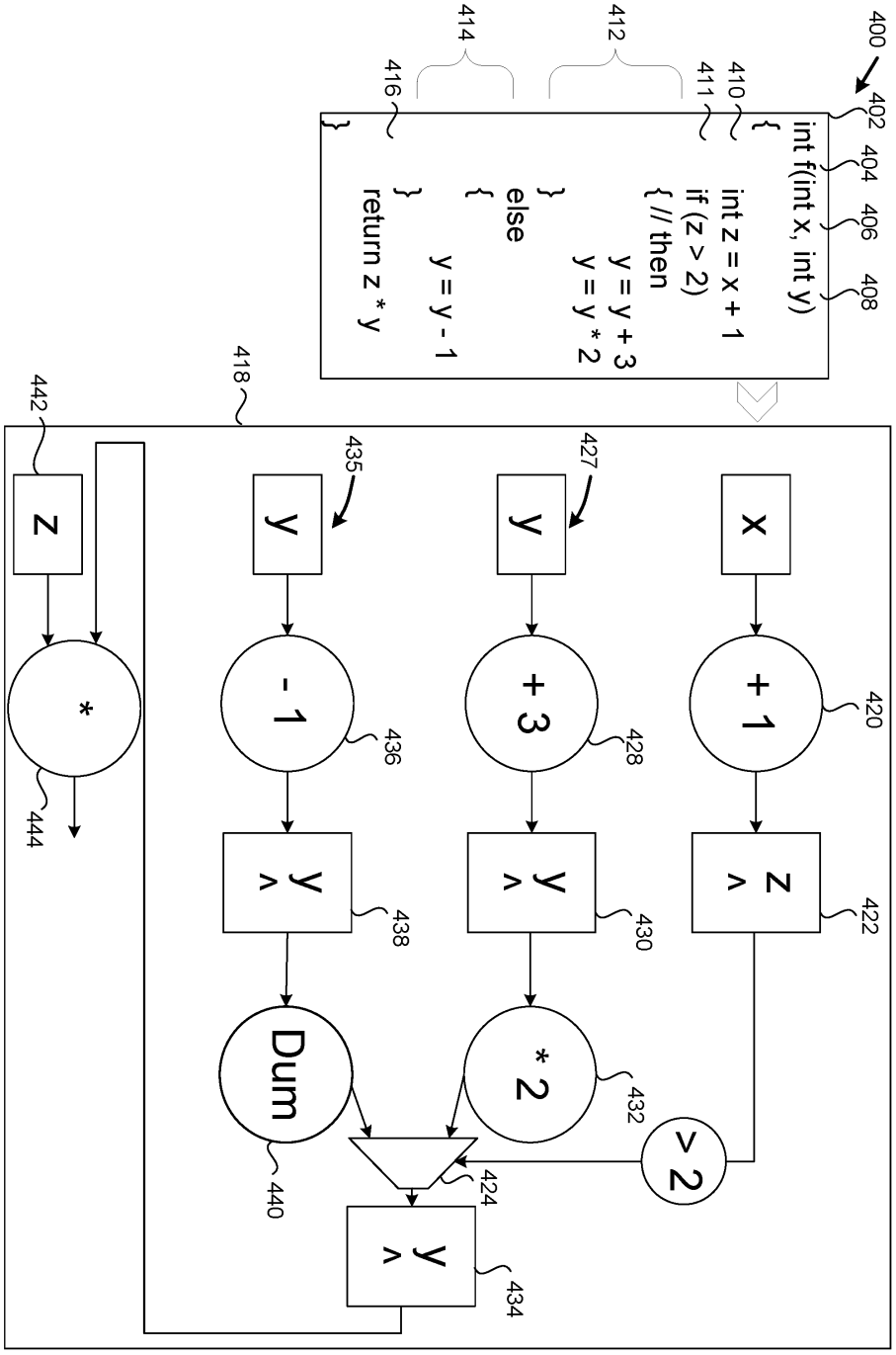
도면2


프로그램 소스 코드
102
정책 210
204A
206A
204B
206B
204C
200A
204A
208A
208B
200A
202A
200B
202B
200C
112

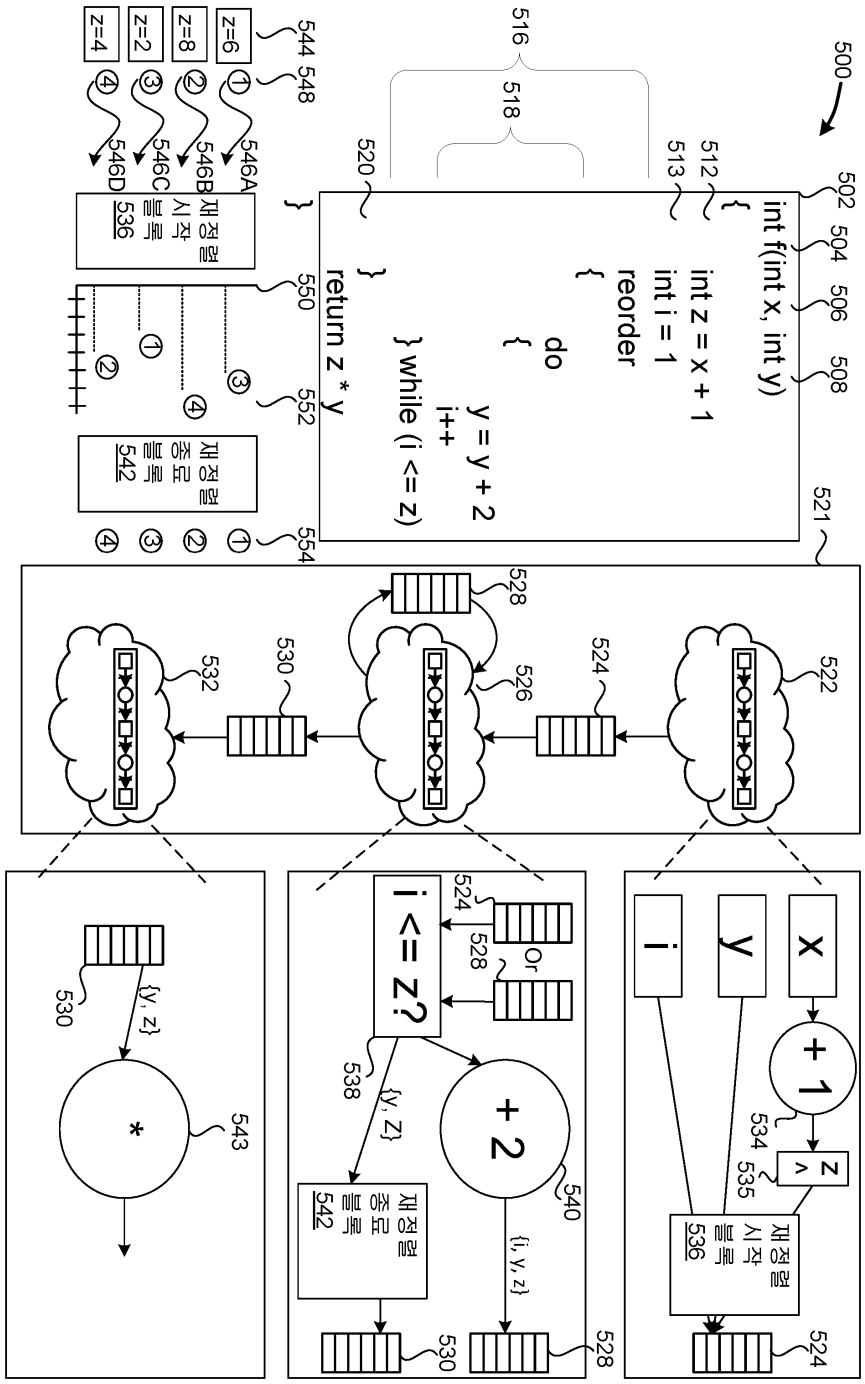
도면3


int f(int x, int y)
{
int z = x + 1
return z * y
}
302
304
306
308
310
312
330
330A
330B
330C
334
336A
336B
336C
x = 1, y = 2
x = 3, y = 5
x = 7, y = 1
y = 2, z = 2
y = 5, z = 4
<4>
332A
332B
332C
340A
340B
342
338A
338B
338C
314
x
y
+ 1
316
z
y
320
322
*
318
324
326
v
328
300

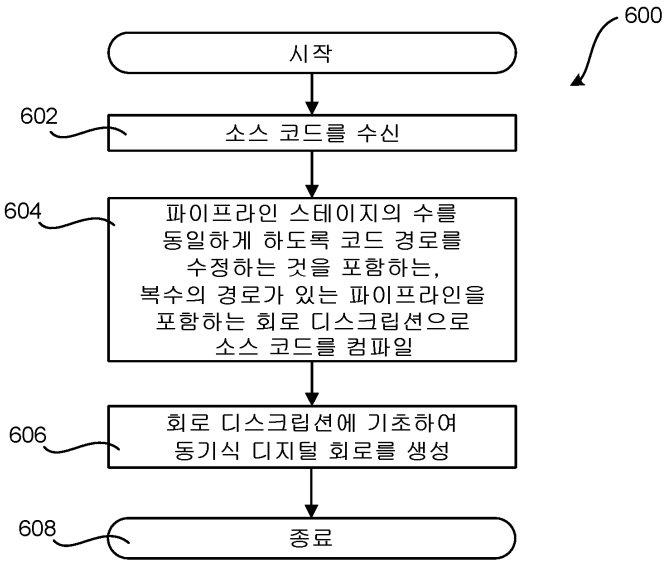
도면4


400
402
404
406
408
410
411
412
414
416
418
int f(int x, int y)
{
int z = x + 1
if (z > 2)
{ // then
y = y + 3
y = y * 2
}
else
{
y = y - 1
}
return z * y
}
x
y
z
+ 1
+ 3
- 1
* 2
> 2
*
Dum
420
422
427
428
430
432
435
436
438
440
442
444
424
434

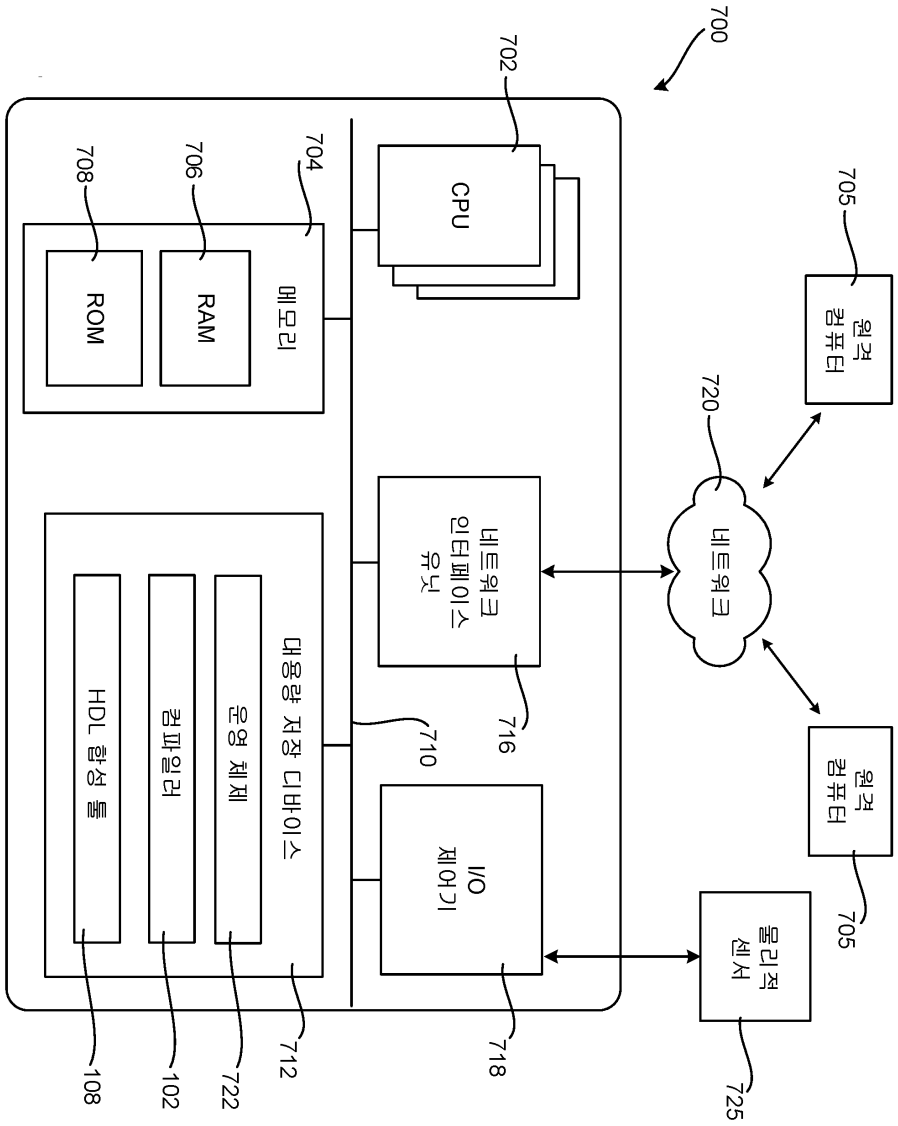
도면5


500
502
504
506
508
512
513
516
518
520
521
int f(int x, int y)
{
int z = x + 1
int i = 1
reorder
{
do
{
y = y + 2
i++
} while (i <= z)
}
return z * y
}
544
548
z=6
z=8
z=2
z=4
546A
546B
546C
546D
재정렬 시작 블록 536
550
552
554
재정렬 종료 블록 542
522
524
526
528
530
532
x
y
i
+ 1
534
535
재정렬 시작 블록 536
i <= z?
538
{y, z}
+ 2
540
{i, y, z}
재정렬 종료 블록 542
Or
*
543

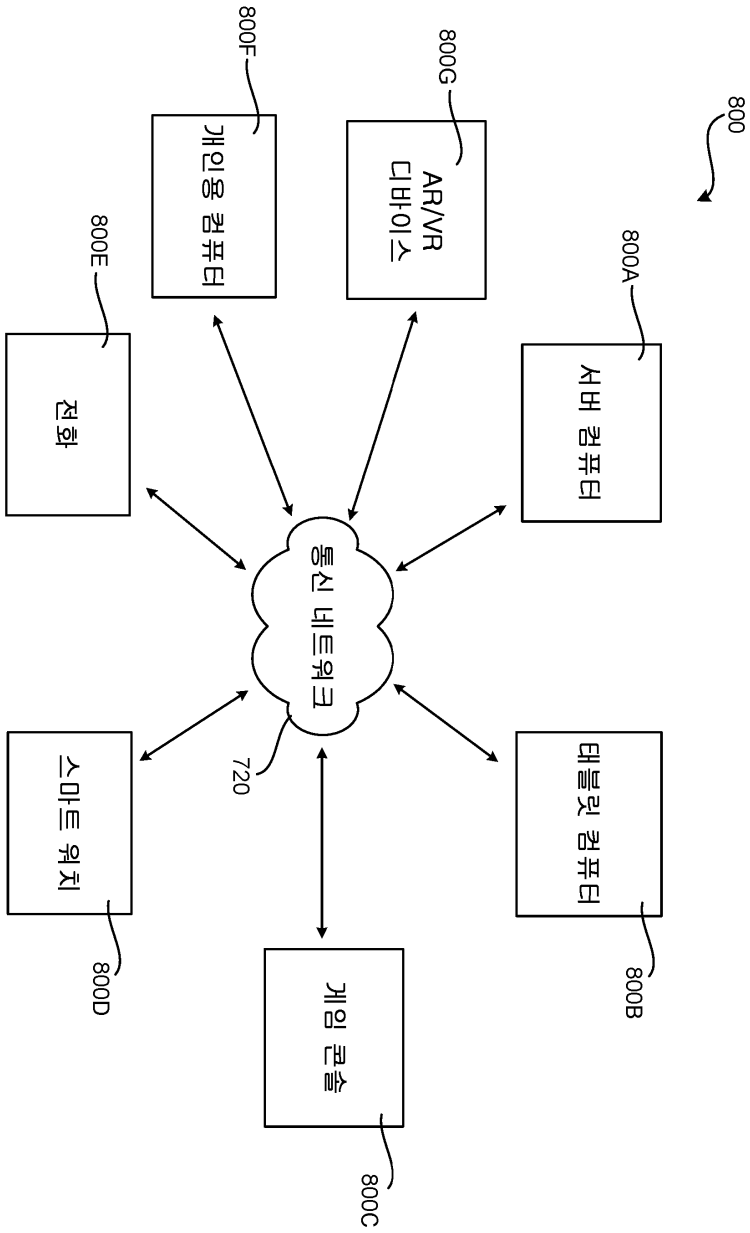
도면6


600
시작
602
소스 코드를 수신
604
파이프라인 스테이지의 수를 동일하게 하도록 코드 경로를 수정하는 것을 포함하는, 복수의 경로가 있는 파이프라인을 포함하는 회로 디스크립션으로 소스 코드를 컴파일
606
회로 디스크립션에 기초하여 동기식 디지털 회로를 생성
608
종료

도면7


700
702
CPU
704
메모리
706
RAM
708
ROM
705
원격 컴퓨터
720
네트워크
네트워크 인터페이스 유닛
716
710
대용량 저장 디바이스
HDL 합성 툴
컴파일러
운영 체제
I/O 제어기
718
물리적 센서
725
712
722
102
108

도면8


800
800A
서버 컴퓨터
800B
태블릿 컴퓨터
800C
게임 콘솔
800D
스마트 워치
800E
전화
800F
개인용 컴퓨터
800G
AR/VR 디바이스
720
통신 네트워크