(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2010년11월04일
(11) 등록번호 10-0991912
(24) 등록일자 2010년10월28일

(51) Int. Cl.
G06F 9/00 (2006.01) *G06F 9/38* (2006.01)
(21) 출원번호 10-2005-7006030
(22) 출원일자(국제출원일자) 2003년10월09일
심사청구일자 2008년09월10일
(85) 번역문제출일자 2005년04월07일
(65) 공개번호 10-2005-0073484
(43) 공개일자 2005년07월13일
(86) 국제출원번호 PCT/US2003/031905
(87) 국제공개번호 WO 2004/034340
국제공개일자 2004년04월22일
(30) 우선권주장
10/269,245 2002년10월11일 미국(US)
(56) 선행기술조사문헌
US 2001/0047468 A1
US 05799812 A
US 05613114 A
US 05339415 A
전체 청구항 수 : 총 16 항

(73) 특허권자
샌드브리지 테크놀로지스, 인코포레이티드
미국, 뉴욕 10591, 테리타운, 포쓰 플로어, 화이트 플레인스 로드 120
(72) 발명자
호케넥, 에르뎀
미국, 뉴욕 10598, 요크타운 하이츠, 페어뷰 코트 3426
모우드길, 마얀
미국, 뉴욕 10607, 화이트 플레인즈, 주니퍼 힐 로드 143
글로스너, 존 씨.
미국, 뉴욕 10512, 카르멜, 베네딕트 플레이스 26
(74) 대리인
최홍걸, 강명구

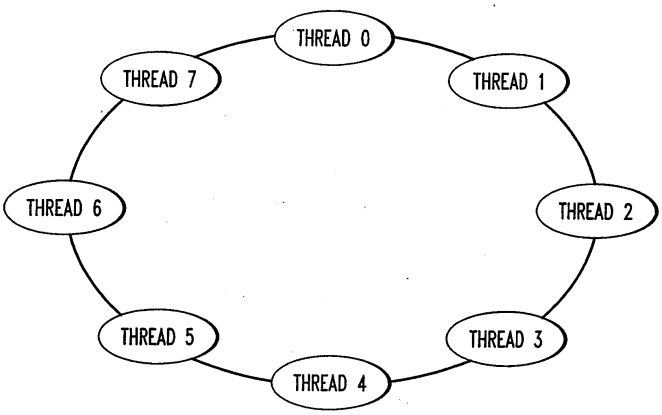
심사관 : 윤혜숙

(54) 토큰 트리거 방식 멀티스레딩을 위한 방법 및 장치

(57) 요 약

본 발명은 멀티스레디드 프로세서에서 토큰 트리거 방식 멀티스레딩 기술에 관한 것이다. 상기 멀티스레디드 프로세서의 다수 스레드에 대한 명령어 발생 시퀀스는 상기 각각의 스레드를 하나이상의 레지스터와 관련시킴으로써 제어되며, 이때 상기 하나 이상의 레지스터는 하나 이상의 명령어를 발생시키도록 다음 스레드를 식별하는 값을 저장하고, 따라서 그리고 상기 저장된 값을 이용하여 상기 명령어 발생 시퀀스를 제어하게 된다. 예를 들어, 상기 멀티스레디드 프로세서의 다수의 하드웨어 스레드 유닛 각각은 상기 하드웨어 스레드 유닛에 의해 업데이트 가능한 상응하는 로컬 레지스터를 포함할 수 있고, 상기 하드웨어 스레드 유닛들 중 주어진 하나에 대한 로컬 레지스터는 상기 주어진 하드웨어 스레드 유닛이 하나이상의 명령어를 발생한 후, 하나 이상의 명령어를 발생시키도록 다음 스레드를 식별하는 값을 저장한다. 전역 레지스터 배열은 또한 대안으로 사용될 수 있다. 상기 프로세서는 스레드 스톨링에 이르는 차단 상태를 야기하지 않고서, 상기 명령어 발생 시퀀스가 임의의 교차하는 짝수-홀수 스레드 시퀀스에 상응하도록 구성될 수 있다.

대 표 도 - 도3



특허청구의 범위

청구항 1

다수의 동시-실행 하드웨어 스레드 유닛을 포함하는 멀티스레디드 프로세서의 다수의 스레드에 대하여 명령어 발생 시퀀스(instruction issuance sequence)를 제어하는 방법에 있어서, 이때 각각의 하드웨어 스레드 유닛이 상기 다수의 스레드 중 해당 스레드에 대해, 프로세서 클럭 사이클 당, 하나 이상의 명령어를 발생시키며, 상기 방법은

- 하나 이상의 명령어를 발생시키도록 허용될 다음번 스레드를 식별하는 값이 저장되며 각각의 스레드와 연계되는 레지스터를 제공하는 단계로서, 이때, 해당 스레드에 대해 레지스터에 저장되는 식별 값은, 주어진 하나의 하드웨어 스레드 유닛이 하나 이상의 명령어를 발생시킨 후 하나 이상의 명령어를 발생시키도록 허용될 또 다른 하드웨어 스레드 유닛을 가리키는 것을 특징으로 하는, 레지스터 제공 단계,

- 다수의 스레드에 대한 다수의 저장된 식별 값들에 의해서 동시-실행 하드웨어 스레드 유닛이 연속적인 프로세서 클럭 사이클에 걸쳐 임의의 순서로 명령어를 발생시키도록 명령어 발생 시퀀스를 제어하기 위해 상기 저장된 식별 값을 이용하는 단계

를 포함하는 것을 특징으로 하는 명령어 발생 시퀀스 제어 방법.

청구항 2

제 1 항에 있어서, 상기 레지스터 제공 단계는, 상기 각각의 하드웨어 스레드 유닛에 의해 업데이트 가능하며 상기 멀티스레디드 프로세서의 다수의 하드웨어 스레드 유닛 각각과 연계된 로컬 레지스터(local register)를 제공하는 단계를 포함하며, 이때, 상기 하드웨어 스레드 유닛 중 주어진 하나에 대하여 상기 로컬 레지스터에 저장되는 식별 값은, 상기 주어진 하드웨어 스레드 유닛이 하나 이상의 명령어를 발생한 후 하나 이상의 명령어를 발생하도록 허용될 다음번 하드웨어 스레드 유닛을 식별하는 값임을 특징으로 하는 명령어 발생 시퀀스 제어 방법.

청구항 3

제 1 항에 있어서, 상기 레지스터 제공 단계는, 각각의 하드웨어 스레드 유닛에 대하여 스레드 식별자 레지스터 및 다음번 스레드 식별자 레지스터를 포함하며 각각의 스레드와 연계되는 다수의 레지스터를 제공하는 단계를 포함하며, 이때, 상기 스레드 식별자 레지스터에 저장되는 식별 값은 현재 하드웨어 스레드 유닛을 특정하는 스레드 식별자이며, 상기 다음번 스레드 식별자 레지스터에 저장되는 식별 값은 다수의 하드웨어 스레드 유닛 중에서, 하나 이상의 명령어를 발생시키도록 허용될 다음번 하드웨어 스레드 유닛을 특정하는 다음번 스레드 식별자인 것을 특징으로 하는 명령어 발생 시퀀스 제어 방법.

청구항 4

제 1 항에 있어서, 상기 레지스터 제공 단계는, 각각의 스레드로 액세스 가능하며 각각의 스레드와 연계된 전역 레지스터(global register)를 제공하는 단계를 포함하며, 이때, 상기 전역 레지스터에 저장되는 값은, 스레드 중 주어진 하나에 의하여 명령어가 발생된 후, 명령어 발생 시퀀스에 따라 하나 이상의 명령어를 발생하도록 허용될 또 다른 하나의 스레드를 식별하는 값임을 특징으로 하는 명령어 발생 시퀀스 제어 방법.

청구항 5

제 1 항에 있어서, 상기 레지스터 제공 단계 및 식별 값 이용 단계는, 명령어 발생 시퀀스가 스레드 스톨링(thread stalling)을 초래하는 차단 상태를 도입하지 않으면서 임의의(arbitrary) 교대하는 짝수-홀수 스레드 시퀀스에 대응하도록 구성되는 것을 특징으로 하는 명령어 발생 시퀀스 제어 방법.

청구항 6

제 1 항에 있어서, 상기 레지스터는 2^n 개의 스레드 중에서 주어진 하나의 스레드의 고유한 식별자를 저장하기 위한 n-비트 레지스터를 포함하는 것을 특징으로 하는 명령어 발생 시퀀스 제어 방법.

청구항 7

제 1 항에 있어서, 상기 레지스터는 상기 멀티스레디드 프로세서의 특정 스레드에 상응하는 스레드 캐시와 연계되는 것을 특징으로 하는 명령어 발생 시퀀스 제어 방법.

청구항 8

제 1 항에 있어서, 상기 레지스터에 저장된 값은 토큰 트리거 방식 스레딩과 관련된 토큰을 포함하는 것을 특징으로 하는 명령어 발생 시퀀스 제어 방법.

청구항 9

제 8 항에 있어서, 상기 토큰 트리거 방식 스레딩은, 현재의 프로세서 클럭 사이클과 관련하여, 후속 클럭 사이클에 대한 명령어를 발생시키도록 다수의 스레드 중 특정 하나를 식별하기 위해 토큰을 이용하는 것을 특징으로 하는 명령어 발생 시퀀스 제어 방법.

청구항 10

제 8 항에 있어서, 상기 토큰 트리거 방식 스레딩은 서로 다른 토큰을 상기 멀티스레디드 프로세서의 다수의 스레드 각각에 할당하는 것을 특징으로 하는 명령어 발생 시퀀스 제어 방법.

청구항 11

제 1 항에 있어서, 상기 멀티스레디드 프로세서는 파이프라인 방식 명령어 프로세싱에 대하여 구성되는 것을 특징으로 하는 명령어 발생 시퀀스 제어 방법.

청구항 12

제 11 항에 있어서, 상기 멀티스레디드 프로세서는 각각의 스레드가 프로세서 클럭 사이클 당 단일 명령어를 발생시키는 명령어 파이프라인을 이용하는 것을 특징으로 하는 명령어 발생 시퀀스 제어 방법.

청구항 13

제 11 항에 있어서, 상기 멀티스레디드 프로세서는 각각의 스레드가 프로세서 클럭 사이클 당 다중 명령어를 발생시키는 명령어 파이프라인을 이용하는 것을 특징으로 하는 명령어 발생 시퀀스 제어 방법.

청구항 14

파이프라인 명령어 프로세싱을 위해 구성된 멀티스레디드 프로세서의 다수의 스레드에 대하여 명령어 발생 시퀀스를 제어하는 방법에 있어서, 상기 방법은

- 하나 이상의 명령어를 발생시키도록 허용될 다음번 스레드를 식별하는 값이 저장되며 각각의 스레드와 연계된 레지스터를 제공하는 단계로서, 이때, 해당 스레드에 대해 레지스터에 저장되는 식별 값은, 주어진 하나의 하드웨어 스레드 유닛이 하나 이상의 명령어를 발생시킨 후 하나 이상의 명령어를 발생시키도록 허용될 또 다른 하드웨어 스레드 유닛을 가리키는 것을 특징으로 하는, 레지스터 제공 단계,

- 각각의 스레드가 프로세서 클럭 사이클 당 다수의 명령어를 발생시키며, 상기 각각의 스레드는 해당 프로세스 클럭 사이클 각각에서 어느 스레드의 스톨링도 없이 로드(load) 명령어와 벡터 곱(vector multiply) 명령어 모두를 발생시키는 명령어 파이프라인(instruction pipeline)을 이용하도록 명령어 발생 시퀀스를 제어하기 위해 상기 저장된 식별 값을 이용하는 단계

를 포함하는 것을 특징으로 하는 명령어 발생 시퀀스 제어 방법.

청구항 15

다수의 동시-실행 하드웨어 스레드 유닛을 포함하는 멀티스레디드 프로세서에 있어서, 상기 멀티스레디드 프로세서는 상기 멀티스레디드 프로세서의 다수의 상응하는 스레드에 대한 명령어 발생 시퀀스를 제어하도록 구성되고, 이때 상기 프로세서 내에는 하나 이상의 명령어를 발생시키도록 허용된 다음번 스레드를 식별하는 값을 저장하도록 상기 각각의 스레드와 연계된 하나 이상의 레지스터가 있으며, 상기 레지스터에 저장된 식별 값은 상기 프로세서의 명령어 발생 시퀀스를 제어하도록 이용되며, 이때,

각각의 하드웨어 스레드 유닛은 스레드들 중 상응하는 하나의 스레드에 대하여 프로세서 클럭 사이클 당 하나

이상의 명령어를 발생시키도록 구성되며, 상응하는 스레드에 대한 저장된 식별 값은, 하드웨어 스레드 유닛 중 주어진 하나에 대하여, 상기 주어진 하드웨어 스레드 유닛이 하나 이상의 명령어를 발생시킨 후에 하나 이상의 명령어를 발생하도록 허용될 하드웨어 스레드 유닛 중 또 다른 하나를 가리키며, 이에 따라서 다수의 스레드에 대한 저장된 값에 의해서 동시-실행 하드웨어 스레드 유닛이 연속적인 프로세서 클럭 사이클에 걸쳐 임의의 순서로 명령어를 발생시키는 것이 가능해지는 것을 특징으로 하는 것을 특징으로 하는 멀티스레디드 프로세서.

청구항 16

다수의 동시-실행 하드웨어 스레드 유닛을 포함하는 멀티스레디드 프로세서의 다수의 스레드에 대하여 명령어 발생 시퀀스(instruction issuance sequence)를 제어하기 위한 방법을 컴퓨터에 실행하는 프로그램이 기록된 컴퓨터 판독형 기록매체에 있어서, 이때 각각의 하드웨어 스레드 유닛이 상기 다수의 스레드 중 해당 스레드에 대해, 프로세서 클럭 사이클 당, 하나 이상의 명령어를 발생시키며, 상기 방법은

- 하나 이상의 명령어를 발생시키도록 허용될 다음번 스레드를 식별하는 값이 저장되고 각각의 스레드와 연계된 레지스터를 제공하는 단계로서, 이때, 해당 스레드에 대해 레지스터에 저장되는 식별 값은, 주어진 하나의 하드웨어 스레드 유닛이 하나 이상의 명령어를 발생시킨 후 하나 이상의 명령어를 발생시키도록 허용될 또 다른 하드웨어 스레드 유닛을 가리키는 것을 특징으로 하는, 레지스터 제공 단계,

- 다수의 스레드에 대한 다수의 저장된 식별 값들에 의해서 동시-실행 하드웨어 스레드 유닛이 연속적인 프로세서 클럭 사이클에 걸쳐 임의의 순서로 명령어를 발생시키도록 명령어 발생 시퀀스를 제어하기 위해 상기 저장된 식별 값을 이용하는 단계

를 포함하는 것을 특징으로 하는 컴퓨터 판독형 기록매체.

명 세 서

기 술 분 야

[0001] 본 발명은 디지털 데이터 프로세서에 관한 것으로, 특히, 멀티스레디드 프로세서에서 사용하기 위한 스레딩 기술에 관한 것이다.

배 경 기 술

[0002] 멀티스레디드 프로세서는 다수의 개별 명령어(instruction) 시퀀스 또는 "스레드(thread)"의 동시 실행을 지원하는 프로세서이다. 종래의 스레딩 기술은 가령, M.J.Flynn, "Computer Architecture: Pipelined and Parallel Processor Design," Jones and Bartlett Publishers, 보스턴, 매사추세츠, 1995 및 G.A.Blaauw and Frederick P.Brooks, "Computer Architecture: Concepts and Evolution," Addison-Wesley, Reading, 매사추세츠, 1997에 설명되어 있으며, 둘다 본원에서 참조로 인용된다.

[0003] 일례로, "배럴 멀티스레딩(barrel multithreading)"으로 알려진 기술은 각각의 스레드가 특정 고정 수의 시퀀스에 따른 명령어를 발생시키도록 한다. 예를 들어, 네 개의 스레드, 즉 Thread0, Thread1, Thread2, 및 Thread3을 갖는 프로세서는 상기 배럴 멀티스레딩에 따라서 상기 스레드들이 고정된 순서로 즉, Thread0, Thread1, Thread2, Thread3, Thread0 등으로 명령어를 발생시키도록 한다.

[0004] 배럴 멀티스레딩 및 다른 현재의 스레딩 기술에 있어서의 문제는 스레드 명령어의 발생의 임의 시퀀싱이 가능하도록 구성되지 않는다는 것이며, 또는 그렇게 구성되더라도 구현을 위해서는 엄청난 양의 하드웨어를 필요로 한다는 점이다.

[0005] 현재의 기술은 따라서 유연성이 없으며, 프로세서 동시성(concurrency)에서 원치않는 제한을 일으킬 수 있다. 더욱이, 이러한 기술은 차단 상태 및 스레딩 스톨링(stalling)을 일으킬 수 있어서 프로세서 성능에 악영향을 미친다.

[0006] 따라서, 멀티스레디드 프로세서에서 사용시 향상된 스레딩 기술을 위한 필요가 존재한다.

발명의 상세한 설명

[0007] 본 발명은 멀티스레디드 프로세서에 있어서 토큰 트리거 방식 스레딩 기술을 제공한다.

[0008] 본 발명의 하나의 형태에 따르면, 멀티스레디드 프로세서의 다수의 스레드의 각각의 스레드와 연계되며, 하나

이상의 명령어를 발생시킬 다음번 스레드를 식별하는 값이 저장되어 있는 하나 이상의 레지스터를 제공하고, 그 후, 저장되어 있는 값을 이용하여, 명령어 발생 시퀀스를 제어함으로써, 멀티스레디드 프로세서의 다수 스레드에 있어서 명령어 발생 시퀀스가 제어된다.

[0009] 전역 레지스터 배열은 또한 대안적으로 사용될 수 있다. 특히, 각각의 스레드와 연계되며, 각각의 스레드에 액세스 가능한 하나 이상의 전역 레지스터가 제공될 수 있고, 전역 레지스터의 콘텐츠는 상기 스레드들 중 하나에 의하여 명령어가 발생된 후 상기 명령어 발생 시퀀스에 따라 하나 이상의 명령어를 발생하도록 상기 스레드들 중 또 다른 하나를 식별한다.

[0010] 본 발명에 따른 멀티스레드 프로세서는 스레드 스톨링에 이르게 되는 차단 상태를 일으키지 않고서 상기 명령어 발생 시퀀스가 가령, 스레드들의 임의의 교차하는 짝수-홀수 시퀀스, 다른 임의의 시퀀스에 상응하도록 구성될 수 있다.

실 시 예

[0019] 본 발명은 예시적인 멀티스레디드 프로세서에서 구현되도록 도시될 것이다. 그러나, 본 발명에서는 도시된 실시예의 특정 멀티스레디드 프로세서 구조가 사용될 필요는 없으며, 보다 일반적으로, 본 발명은, 명령어 파이프라이닝과 결합된 토큰 트리거 방식의 멀티스레딩을 사용함으로써 개선된 성능을 제공하는 것이 바람직한 임의의 멀티스레디드 프로세서 용도에서 사용되기 적합하다.

[0020] 본 발명에 따른 토큰 트리거 방식 스레딩 기술을 구현하는 예시적 프로세싱 시스템(100)은 도 1 및 2와 연계하여 설명될 것이다.

[0021] 도 1의 프로세싱 시스템은 주 메모리(104)에 연결된 멀티스레디드 프로세서(102)를 포함한다. 상기 멀티스레디드 프로세서(102)는 멀티스레디드 캐시 메모리(110) 및 멀티스레디드 데이터 메모리(112)를 포함한다.

[0022] 도 2는 상기 멀티스레디드 프로세서(102)의 가능한 하나의 구현예를 상세히 보여주고 있다. 이 실시예에서, 상기 멀티스레디드 프로세서(102)는 멀티스레디드 캐시 메모리(110), 데이터 메모리(112), 캐시 컨트롤러(114), 명령어 디코더(116), 레지스터 파일(118), 및 산술 연산 유닛(ALU)(120)을 포함한다. 상기 멀티스레디드 캐시 메모리(110)는 또한 멀티스레디드 캐시로 일컬어진다.

[0023] 도 1 및 2에 도시된 특정 배열은 도해의 목적으로 단순화되어 있고, 도시되지 않은 추가 요소 및 다른 대안 요소도, 당업자에게 분명하기 때문에 포함될 수 있다.

[0024] 멀티스레디드 캐시(110)는 다수의 스레드 캐시(110-1, 110-2,...110-N)를 포함하고, 이때 N은 상기 멀티스레디드 프로세서(102)에 의해 지원되는 스레드의 수를 표시한다. 따라서 각각의 스레드는 상기 멀티스레디드 캐시(110) 내에 각각의 스레드와 연계된 상응하는 스레드 캐시를 갖는다. 마찬가지로, 상기 데이터 메모리(112)는 N개의 데이터 메모리, 즉 112-1, 112-2,...112-N을 포함한다.

[0025] 상기 멀티스레디드 캐시(110) 내의 각 스레드 캐시는 하나 이상의 메모리 위치 세트를 갖는 메모리 어레이를 포함할 수 있다. 주어진 스레드 캐시는 관련 스레드 식별자를 저장하기 위한 스레드 식별자 레지스터를 추가로 포함할 수 있으며, 이는 도 7과 관련하여 아래에서 설명될 것이다.

[0026] 멀티스레디드 캐시(110)는 상기 캐시 컨트롤러(114)를 통해 주 메모리(104)와 인터페이스한다. 캐시 컨트롤러(114)는 주 메모리(104)로부터의 적절한 명령어가 상기 멀티스레디드 캐시(110)으로 로딩되는 것을 보장한다. 상기 캐시 컨트롤러(114)는 이 실시예에서, 각각의 스레드 캐시 110-1, 110-2,...110-N과 관련된 논리회로 또는 다른 프로세싱 요소와 결합하여 동작하면서, 어소시에이티브 매핑(associative mapping), 다이렉트(direct) 매핑, 또는 세트-어소시에이티브(set-associative) 매핑 등과 같은 어드레스 매핑 기술의 적어도 일부분을 구현한다. 본 발명과 결합하여 사용하기에 적합한 세트-어소시에이티브 매핑 기술은 미국 특허 출원 10/161,774 및 10/161,874에 설명되어 있으며, 둘다 2002년 6월 4일에 출원되었고 본원에서 참조로 인용된다.

[0027] 일반적으로, 상기 멀티스레디드 캐시(110)는 상기 멀티스레디드 프로세서(102)에 의해 실행될 명령어를 저장하는데 사용되고, 반면 데이터 메모리(112)는 상기 명령어에 의해 연산되는 데이터를 저장한다. 명령어는 종래의 방식으로 명령어의 실행을 제어함에 있어서, 레지스터 파일(118) 및 ALU(120)와 결합하여 동작하는 명령어 디코더(116)에 의해 상기 멀티스레디드 캐시(110)로부터 인출된다. 116, 118, 및 120과 같은 멀티스레디드 프로세서 요소의 동작은 당분야에 공지되어 있기 때문에 본원에서는 추가로 상세히 설명되지 않는다.

[0028] 상기 데이터 메모리(112)는 주 메모리(104)에 직접 연결되어 있지만, 도면에서는 나타나지 않는다.

[0029] 메모리(104, 110, 및 112) 중 하나 이상은 다중 뱅크 또는 다른 지정된 부분을 포함하도록 각각 구성될 수 있다. 실시예에서, 각각의 뱅크는 하나 이상의 메모리 모듈, 또는 단일 메모리의 특정 부분으로 구성되는 것으로 보일 수 있다.

[0030] 멀티스레디드 프로세서와 연계되는 이러한 메모리 및 그 밖의 다른 메모리들의 스레드 방식 뱅킹 기술은 미국 특허 출원 도켓 번호 1007-5 "Method and Apparatus for Thread-Based Memory Access in a Multithreaded Processor"에 설명되어 있다.

[0031] 본 발명은 도 2에 도시된 특별한 멀티스레디드 프로세서 구조를 필요로 하지는 않는다. 본 발명은 다양한 다른 멀티스레디드 프로세서 구조에서 구현될 수 있다.

[0032] 도 2에 도시되어 있고 본 발명과 결합하여 사용하기에 적합한 형태의 멀티스레디드 프로세서의 보다 많은 특별한 예는 2001년 12월 20일에 출원되었고 본원에서 참조로 인용되는 미국 가 특허 출원 번호 60/341,289에 설명되어 있다. 미국 가 특허 출원 번호 60/341,289에 설명된 멀티스레디드 프로세서의 실시예는 RISC-방식 제어 코드, 디지털 신호 프로세서(DSP) 코드, 자바 코드 및 네트워크 프로세싱 코드 등을 실행할 수 있다. 상기 프로세서는 SIMD(single instruction multiple data) 벡터 유닛, 리덕션(reduction) 유닛, 및 LIW(long instruction word) 복합 명령어 실행을 포함한다.

[0033] 본 발명의 하나의 형태에서, 본 발명은 멀티스레디드 프로세서(가령, 도 2의 프로세서(102))에서 향상된 성능을 제공한다. 특히, 아래에서 설명될 것이지만, 상기 프로세서(102)는, 본 발명의 기법에 따라서, 명령어 파이프라이닝과 연계되어 동작하는 토큰 트리거 방식 스레딩 기술을 이용하도록 구성되어, 강화된 프로세서 동시성(processor concurrency)이 제공되고, 스레드 스톨링의 가능성이 낮아진다.

[0034] 도 3은 스레드의 수 N이 8인 프로세서(102)를 구현하기 위한 토큰 트리거 방식 스레딩을 보여준다. 일반적으로, 모든 스레드는 동시에 동작하고, 각 스레드는 상응하는 스레드 캐시(110) 및 데이터 메모리(112)에 액세스한다. 도 3에서, 8개의 스레드는 Thread0, Thread1, Thread2,...Thread7로 표시되고, 링(ring) 형태로 직렬로 상호연결되도록 도시되어 있다. 상기 멀티스레디드 프로세서에서, 주어진 스레드는 일반적으로 하드웨어와 소프트웨어의 측면으로 나타내어질 수 있다. 주어진 스레드와 연계되는 특정 프로세서 하드웨어는 따라서 본원에서 하드웨어 스레드 유닛 또는 단순히 "콘텍스트(context)"로 일컬어진다.

[0035] 도 3에 도시된 토큰 트리거 방식 스레딩에 따르면, 모든 하드웨어 스레드 유닛 또는 콘텍스트는 명령어를 동시에 실행(execute)하도록 허용되지만, 단지 하나의 콘텍스트만이 상기 프로세서의 특정 클럭 사이클에서 명령어를 생성(issue)할 수 있다. 다시 말해서, 모든 콘텍스트는 동시에 실행하지만, 단지 하나의 콘텍스트만이 특정 클럭 사이클에서 활성화된다. 따라서, 만일 총 C개의 콘텍스트가 있다면, 모든 콘텍스트로부터 명령어를 생성하기 위해서는 C개의 클럭 사이클이 필요할 것이다. 각각의 클럭 사이클마다, 상기 콘텍스트 중 하나가 명령어를 생성하고, 명령어를 발생할 다음번 스레드가 토큰에 의해 지시된다.

[0036] 도 3의 예에서, 토큰이 순차적으로 또는 라운드-로빈 방식으로 배열됨으로써 콘텍스트들이 순차적으로 명령어를 발생시킬 것이다. 그러나, 명령어를 생성할 다음번 콘텍스트를 나타내는 토큰은, 가령, 교차하는 짝수-홀수 패턴과 같은 그 외 다른 패턴을 사용하여 배열될 수 있다. 또한 앞서 언급된 바와 같이, 그 밖의 다른 스레딩 방식이 본 발명과 결합하여 사용될 수 있다. 본 발명에 따른 스레딩 기술의 수많은 예는 도 7 및 8과 결합하여 아래에 설명될 것이다.

[0037] 도 4는 예시적 명령어 함수가 본 발명에 따라 멀티스레디드 프로세서(102)에서 파이프라이닝될 수 있는 방식을 보여준다. 본 발명의 실시예에서, 이러한 방식의 파이프라이닝은, 앞서 설명된 토큰 트리거 방식 스레딩과 결합하여 사용되는 것이 선호되지만, 본 발명을 구현함에 있어서 파이프라이닝과 스레딩의 수많은 그 밖의 다른 조합이 사용될 수 있다.

[0038] 도 4의 파이프라인은 도 3의 N=8인 토큰 트리거 방식 스레딩과 결합하여 사용되도록 구성된다. 도 4의 예시적 명령어 함수는 로드/저장(Load/Store)(Ld/St), ALU, 정수곱(I_Mul) 및 벡터곱(V_Mul)을 포함하고, 각각 9개, 6개, 7개 및 8개의 파이프라인 단계를 갖는 것으로 도시되어 있다.

[0039] 도 4에 도시된 각각의 예시적 명령어 파이프라인은 적어도 명령어 디코드 단계, 레지스터 파일(RF) 읽기 단계, 이동(Xfer) 단계 및 재기록(writeback)(WB) 단계를 포함한다. 상기 RF 읽기 단계는 레지스터 파일(가령, 레지스터 파일(118))로부터의 읽기에 관련되어 있으며, 상기 이동 단계는 명령어 결과를 지정된 홀딩 레지스터로 이동하는 것에 관련되어 있고, 상기 WB 단계는 명령어 결과를 메모리 또는 레지스터 파일로 다시 기록하는 것에 관

련되어 있다.

[0040] 상기 Ld/St 파이프라인은 어드레스 생성(Agen) 단계, 내부(Int) 또는 외부(Ext) 결정 단계, 및 세 개의 추가적인 메모리 실행 단계, 즉 Mem0, Mem1, 및 Mem2를 추가로 포함한다. 상기 Ld/St 파이프라인은 따라서 총 네 개의 메모리 실행 단계, 즉 Mem0, Mem1, Mem2 및 WB를 포함한다. 상기 내부 또는 외부 결정 단계는 상기 관련된 메모리 액세스가 내부 또는 외부 메모리로의 액세스인지를 결정하고, 그리고 상기 파이프라인 내의 추가적인 디코드 단계로서 보여질 수 있다. 특정 외부 메모리 액세스를 위해 추가 메모리 실행 단계가 필요할 수 있다. 예를 들어, 만일 외부 메모리 액세스의 WB 단계가 상응하는 스레드가 활성화되는 주기 동안 종료되지 않으면, 상기 스레드는 스톨링(stalling)됨으로써 상기 WB 단계는 상기 스레드가 활성화되는 다음 순간을 종료할 것이다.

[0041] ALU 파이프라인은 Exec1 및 Exec2로 표시된 두 개의 실행 단계를 추가로 포함한다.

[0042] 정수 I_Mul 파이프라인은 Exec1, Exec2 및 Exec3으로 표시된 세 개의 실행 단계를 추가로 포함한다.

[0043] 상기 벡터 V_Mul 파이프라인은 두 개의 곱셈 단계 MPY1 및 MPY2, 두 개의 덧셈 단계 Add1 및 Add2를 추가로 포함한다.

[0044] 상기 멀티스레디드 프로세서(102)는, 특정 콘텍스트로부터의 명령어가 자신의 대응하는 파이프라인에 들어가면, 종료를 실행하도록 구성되는 것이 바람직하다.

[0045] 적절히 구성된 파이프라인 및 충분한 수의 스레드를 가짐으로써, 비록 사이클 당 하나의 콘텍스트에서 하나의 명령어가 발생되더라도 모든 하드웨어 콘텍스트가 동시에 실행될 수 있다. 앞서 언급한 바와 같이, 스레드 및 파이프라인 단계의 특정 수는 단지 도시의 목적이며, 선호되는 구현예를 반영하는 것은 아니다. 당업자는 본원에서 제공되는 특별한 용도에서 스레드 및 파이프라인 단계의 적절한 수를 결정할 수 있을 것이다.

[0046] 도 4의 파이프라인의 동작이 도 5 및 6을 참조하여 지금부터 설명될 것이다. 도 5 및 6은 특정 프로세서 스레드에 의해 발생된 명령어의 시퀀스들을 각각 보여주고, 또한 상기 레지스터 파일(118)의 짝수 부분 또는 홀수 부분이 상응하는 스레드에 의해 이용되는지를 각 명령어에 대해 표시한다. 앞서 언급된 미국 특허 출원 도켓 번호 1007-7 "Method and Apparatus for Register File Port Reduction in a Multithreaded Processor"는 상기 레지스터 파일(118)이 어떻게 짝수 부분 및 홀수 부분으로 분리되는지를 보여주고, 이때 상기 두 부분 중 하나는 스레드 식별자를 이용하여 선택가능하다.

[0047] 도 5 및 6에서는 설명의 단순성을 위해 스레드의 수 N을 4로 가정하고, 각각의 스레드가 도 3의 토큰 트리거 방식 스레딩의 라운드-로빈 구현예에 따른 명령어를 발생시킨다. 특히, 상기 예에서 각 스레드는 로드(Load)와 벡터 곱셈(Vector Multiply) 명령어를 교대로 발생하는데, 이는 다수의 신호 처리 적용예에서의 통상적인 명령어 시퀀스이다. 상기 로드 및 벡터 곱셈 명령어는 도 4의 명령어 함수 파이프라인에 따라 설명되는 바와 같이 구성된다.

[0048] 도 5에서, 단일 발생 파이프라인의 한 예가 도시되어 있으며, 여기서, 각각의 스레드는 사이클 당 하나의 명령어를 발생한다. 도 5에서, 발생된 명령어에 의해 액세스되는 레지스터 파일 부분은 스레드들 간에서 짝수(e)와 홀수(o)로 교대함을 알 수 있다. 이는, 인접한 레지스터 재기록(writeback) 연산들, 가령 Thread4의 제 1 로드 명령어와 Thread1의 벡터 곱셈 명령어의 재기록 단계와 관련된 레지스터 재기록 연산들은 레지스터 파일의 서로 다른 부분에 관련됨을 보장한다. 특히, Thread4의 제 1 로드 명령어는 레지스터 파일의 짝수 부분과 관련되고, 반면 Thread1의 벡터 곱셈 명령어는 상기 레지스터 파일의 홀수 부분과 관련된다. 다이어그램에서 나타나는 그 밖의 다른 명령어들도 이와 유사하게 구성된다.

[0049] 스레드 식별자는, 주어진 스레드에 의해 레지스터 파일의 짝수 부분 또는 홀수 부분이 액세스될 것인지를 선택하는데 사용된다. 예를 들어, 도 5 및 6의 N=4인 경우에, 스레드 식별자의 최하위 비트(LSB)가 레지스터 파일의 짝수 부분과 홀수 부분 간 선택을 위해 사용될 수 있다.

[0050] 도 6은 각각의 프로세서 스레드가 사이클 당 2개의 명령어를 발생시키는 예시적인 다중 발생 파이프라인(multiple issue pipeline)을 보여준다. 여기서, 하나의 단일 스레드는 각 사이클마다 로드 명령어와 벡터 곱셈 명령어 모두를 발생시킨다. 사이클 당 다수의 명령어가 발생되기 때문에, 도 5의 예에서는 두 개의 추가 레지스터 파일 읽기 포트가 필요하다. 그러나, 도면에서 알 수 있듯이, 스레드 식별자의 최하위 비트를 바탕으로 결정되는 바와 같이 모든 동시적인 기록은 레지스터 파일의 짝수 또는 홀수 부분으로 이루어지며, 따라서 필요한 레지스터 파일 쓰기 포트의 개수가 감소되고, 따라서 프로세서 전력 소비가 감소된다.

[0051] 도 5 및 6과 결합하여 보여진 스레드의 특정 수는 예에 불과하며, 본 발명이 특정 스레드 수를 사용하는 것에

제한되는 것은 아니다.

[0052] 도 7은 본 발명에 따른 도 2의 멀티스레디드 프로세서(102)에서 구현될 수 있는 하드웨어 스레드 유닛(702-i)의 세트(700)를 보여준다(이때 i=1, 2,...N). 앞서 언급된 바와 같이, 이러한 하드웨어 스레드 유닛은 본원에서 콘텍스트라고도 일컬어지며, 상기 프로세서에 의해 지원되는 각각의 스레드와 연계되는 하나의 유닛 또는 콘텍스트가 있다. 상기 하드웨어 스레드 유닛(702) 각각은 스레드 식별자(TID) 레지스터(704) 및 다음번 스레드 식별자(NTID) 레지스터(706)를 포함한다. 게다가, 상기 하드웨어 스레드 유닛(702) 각각은 상응하는 스레드 캐시(110) 및 데이터 캐시(112)를 포함할 수 있으며, 이에 추가하여, 또는 이를 대체하여, 캐시 컨트롤러(114)나 그 밖의 다른 프로세서 회로의 관련된 부분을 포함할 수 있다.

[0053] 본 발명에 따르면, 각각의 하드웨어 스레드 유닛(702)은 주어진 프로세서 사이클 당 하나 이상의 명령어를 발생시킬 수 있다. 주어진 하드웨어 스레드 유닛의 TID 레지스터(704)는 해당 스레드의 스레드 식별자를 저장한다. 주어진 하드웨어 스레드 유닛의 NTID 레지스터(706)는, 상기 주어진 하드웨어 스레드 유닛이 자신의 명령어를 발생한 후 명령어를 발생시킬 다음번 하드웨어 스레드 유닛의 스레드 식별자를 저장한다. 따라서 상기 NTID 레지스터(706)는 본 발명의 토큰 트리거 방식 스레딩 기술과 관련된 토큰을 저장하는 것으로 보여질 수 있다. 상기 토큰은 상기 주어진 하드웨어 스레드 유닛에 의하여 그 밖의 다른 하드웨어 스레드 유닛의 스레드 식별자로 설정될 수 있는데, 이는, 상기 토큰 트리거 방식 멀티스레딩 프로세서에 상당한 유연성을 제공한다.

[0054] 상응하는 하드웨어 스레드 유닛과 연계되고 개별적으로 프로그램가능한 로컬 레지스터로서 도 7에 도시되어 있지만, 이에 추가로, 또는 이를 대체하여, 상기 NTID 레지스터가 상기 모든 하드웨어 스레드 유닛에 액세스할 수 있는 전역 레지스터(global register)로서 구현될 수도 있다. 이 경우, 각각의 하드웨어 스레드 유닛이 상기 전역 NTID 레지스터를 연속으로 증분시킬 필요는 없다. 예를 들어, 상기 하드웨어 스레드 유닛의 외부에 구현된 상태 머신(state machine) 또는 그 밖의 다른 유사한 회로가 사용되어, 임의의 희망하는 토큰 시퀀스를 제공하도록 전역 NTID 레지스터를 프로그램할 수 있다.

[0055] 도 8은 교대하는 짝수-홀수 패턴에 대응하는 예시적인 토큰 시퀀스를 보여준다. N=8인 토큰 트리거 방식 스레딩 예에서 토큰 시퀀스는 Thread0(T0), Thread3(T3), Thread2(T2), Thread1(T1), Thread6(T6), Thread5(T5), Thread4(T4), Thread7(T7), Thread0(T0) 등이다.

[0056] 상기 예는 도 7에 도시된 NTID 레지스터를 사용함으로써, 실행 파이프라인에서 어떠한 충돌도 일으키지 않으며, 따라서 스레드 스톨링(thread stalling) 없이,각각의 콘텍스트가 짝수-홀수 패턴으로, 그 밖의 다른 임의의 콘텍스트에게 토큰을 부여할 수 있음을 보여준다. 명령어 파이프라이닝과 이러한 스레딩 기술의 조합은 차단 상태의 발생 횟수를 상당히 줄여주고, 짝수-홀수 스레드 시퀀스의 임의적 실행을 가능하게 한다.

[0057] 앞서 언급된 바와 같이, 상기 스레드 식별자 레지스터(704)는 특정 스레드를 식별하기 위해 멀티스레디드 프로세서(102)에 의해 사용되는 멀티-비트 스레드 식별자를 저장한다. 이러한 스레드 식별자는 당업자에게 공지되어 있는 종래의 방식으로 생성될 수 있다.

[0058] 본원에서 사용되는 용어 "스레드 식별자"는 멀티스레디드 프로세서에서 다수의 스레드 세트 또는 특정 스레드를 식별하기에 적합한 정보를 포함하도록 의도된다. 제한받지 않는 예를 들자면, n-비트 스레드 식별자는 멀티스레디드 프로세서에 의해 지원되는 $N=2^n$개의 스레드 중 하나를 고유하게 식별하기 위해 사용될 수 있다. 본 발명에서 사용하기에 적합한 다양한 스레드 식별자 구조는 당업자에게 분명할 것이다.

[0059] 앞서 언급된 바와 같이, 본 발명의 토큰 트리거 방식 스레딩 기술은 종래의 기술에 비해 상당한 개선점을 제공한다. 예를 들어, 상기 기술은 스레드 스톨링의 가능성을 상당히 감소시킬 수 있다. 더욱이, 이러한 향상은 프로세서 동시성 또는 프로세서의 성능에 영향을 주지 않으면서 제공된다.

도면의 간단한 설명

[0011] 도 1은 본 발명이 구현된 예시적 프로세싱 시스템의 블록도이다.

[0012] 도 2는 도 1의 프로세싱 시스템의 멀티스레디드 프로세서의 보다 상세한 블록도이다.

[0013] 도 3은 본 발명의 기술에 따라 도 2의 멀티스레디드 프로세서에서 사용하기 에 적합한 토큰 트리거 방식 스레딩의 예이다.

[0014] 도 4는 본 발명의 기술에 따라 예시적 명령어 함수가 도 2의 멀티스레디드 프로세서에서 파이프라인될 수 있는

방식을 보여준다.

[0015] 도 5는 도 2의 프로세서의 각 스레드가 사이클 당 하나의 명령어를 발생시키는 단일 발생 파이프라인을 보여준다.

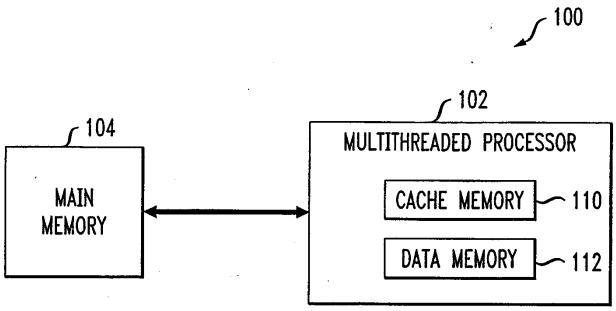
[0016] 도 6은 도 2의 프로세서의 각 스레드가 사이클 당 두 개의 명령어를 발생시키는 다중 발생 파이프라인을 보여준다.

[0017] 도 7은 도 2의 프로세서에서 사용하기에 적합하고 본 발명에 따르도록 구성된 하드웨어 스레드 유닛을 보여준다.

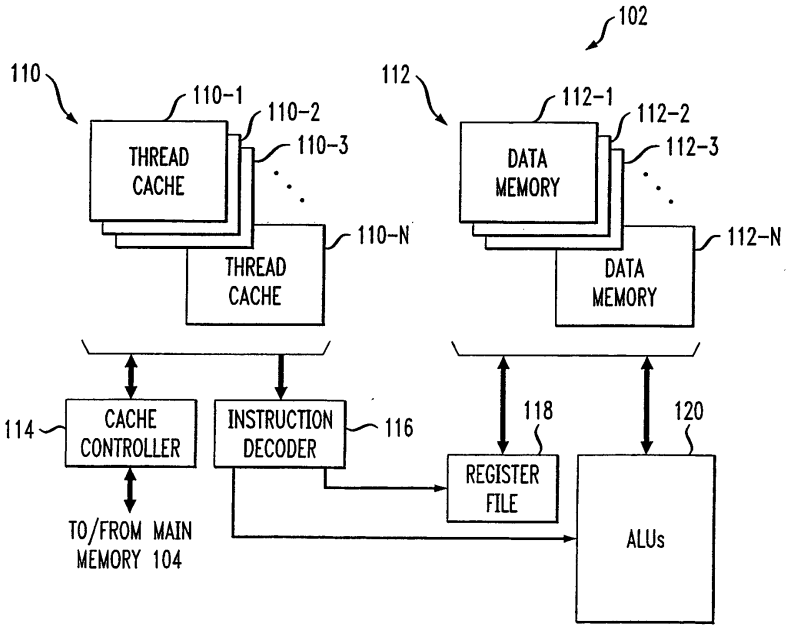
[0018] 도 8은 도 2의 프로세서의 실시예에서 구현된 토큰 트리거 방식 스레딩 기술을 보여준다.

도면

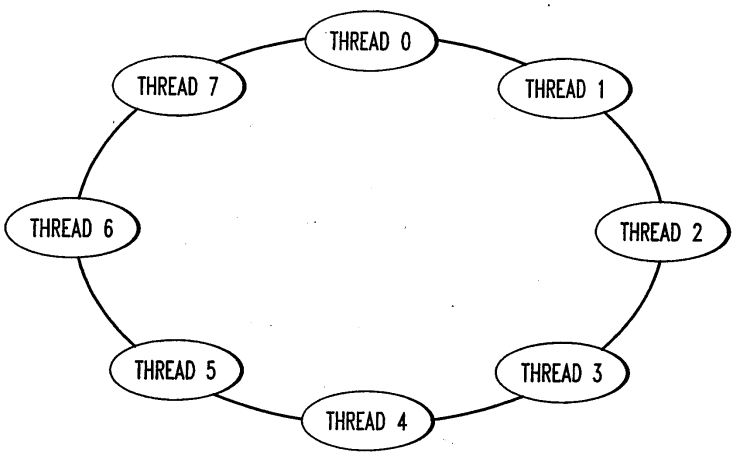
도면1



도면2



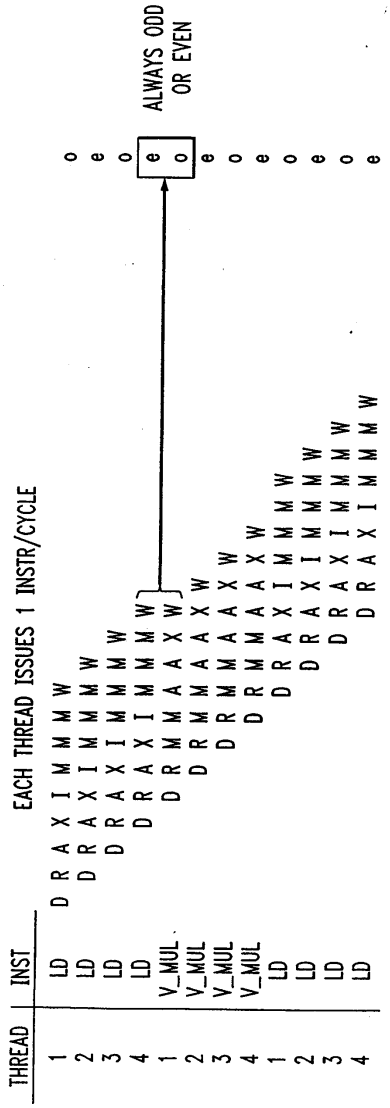
도면3



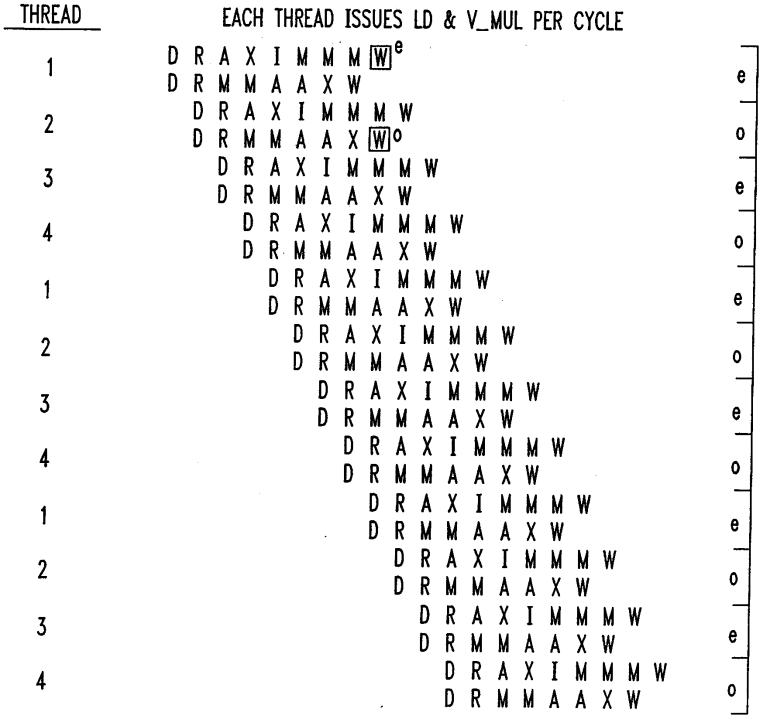
도면4

Ld/St	INST DEC	RF READ	AGEN	XFER	INT EXT	MEM 0	MEM 1	MEM 2	WB
ALU	INST DEC	RF READ	EXEC1	EXEC2	XFER	WB			
I_Mul	INST DEC	RF READ	EXEC1	EXEC2	EXEC3	XFER	WB		
V_Mul	INST DEC	RF READ	MPY1	MPY2	ADD1	ADD2	XFER	WB	

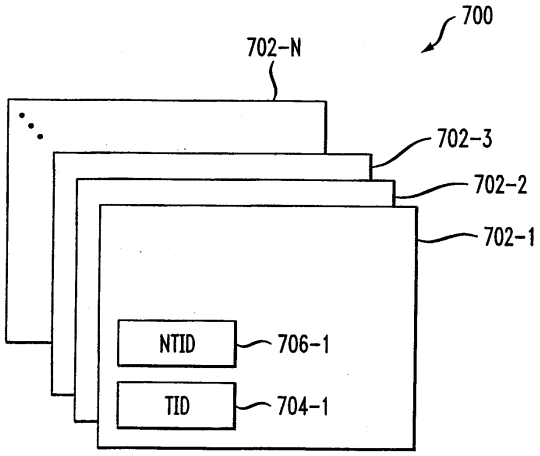
도면5


EACH THREAD ISSUES 1 INSTR/CYCLE
THREAD
INST
LD
V_MUL
D R A X I M M M W
ALWAYS ODD OR EVEN
o
e

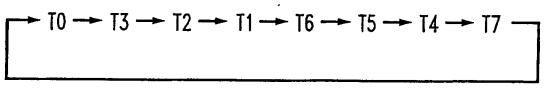
도면6


THREAD
EACH THREAD ISSUES LD & V_MUL PER CYCLE
1
2
3
4
1
2
3
4
1
2
3
4
e
o
e
o
e
o
e
o
e
o
e
o

도면7


700
702-N
702-3
702-2
702-1
NTID
706-1
TID
704-1

도면8


T0
T3
T2
T1
T6
T5
T4
T7