



(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2013년08월05일
(11) 등록번호 10-1293219
(24) 등록일자 2013년07월30일

(51) 국제특허분류(Int. Cl.)
G06F 11/28 (2006.01) G06F 11/36 (2006.01)
G06F 9/46 (2006.01)
(21) 출원번호 10-2012-0021230
(22) 출원일자 2012년02월29일
심사청구일자 2012년02월29일
(56) 선행기술조사문헌
김정사 외 1인, '요약보고 방법에 의해 병목현상을 개선한 최초 경합의 수행중 탐지기법', 정보과학회논문지 제26권 제9호, pp.1042-1054 (1999.9)*
obert H.B. Netzer외 1인, 'EFFICIENT RACE CONDITION DETECTION FOR SHARED-MEMORY PROGRAMS WITH POST/WAIT SYNCHRONIZATION', In 1992 Intl Conference on Parallel Processing, St. Charles, IL(1992.8)*
KR1020040027556 A*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
경상대학교산학협력단
경상남도 진주시 진주대로 501 (가좌동)
(72) 발명자
전용기
경상남도 진주시 하대동 현대아파트 102-1506
박길준
대전광역시 유성구 하기동 524-2번지
강문혜
경상남도 진주시 호탄동 대경아르떼빌 305호
(74) 대리인
오세준, 권혁수, 송윤호

전체 청구항 수 : 총 15 항

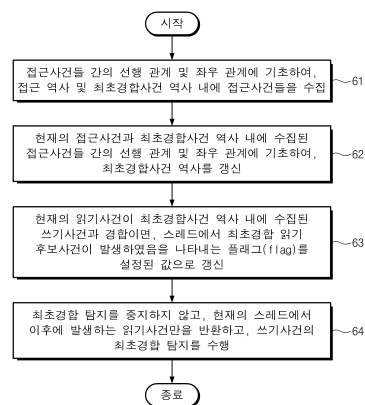
심사관 : 이정은

(54) 발명의 명칭 공유메모리 프로그램의 최초경합 탐지 방법 및 장치

(57) 요약

본 발명은 공유메모리 프로그램의 최초경합 탐지 장치 및 방법에 관한 것으로, 공유메모리 프로그램에서 공유 변수에 대한 최초경합을 탐지하는 장치에 있어서, 읽기사건이 상기 최초경합의 읽기 후보사건으로 결정되면, 상기 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 플래그(flag)를 설정된 값으로 갱신하고, 상기 플래그 값을 이용하여 이후에 발생되는 접근사건에 대한 경합 탐지를 수행하는 최초경합 탐지부를 포함함을 특징으로 하는 공유메모리 프로그램의 최초경합 탐지 장치, 및 이에 의하여 수행되는 공유메모리 프로그램의 최초경합 탐지 방법을 개시한다.

대표도 - 도6



특허청구의 범위

청구항 1

공유메모리 프로그램에서 공유 변수에 대한 최초경합을 탐지하는 장치에 있어서,

읽기사건이 상기 최초경합의 읽기 후보사건으로 결정되면, 상기 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 플래그(flag)를 설정된 값으로 갱신하고, 상기 플래그 값을 이용하여 이후에 발생하는 접근사건에 대한 경합 탐지를 수행하는 최초경합 탐지부를 포함하며,

상기 최초경합 탐지부는,

상기 플래그가 상기 설정된 값으로 갱신되면, 상기 스레드에서의 최초경합 탐지를 중지하지 않고, 상기 스레드에서 상기 읽기사건 이후에 발생하는 읽기사건만을 반환하여, 상기 스레드에서 상기 읽기사건 이후에 발생하는 쓰기사건에 대하여 최초경합 탐지를 수행함을 특징으로 하는 공유메모리 프로그램의 최초경합 탐지 장치.

청구항 2

삭제

청구항 3

공유메모리 프로그램에서 공유 변수에 대한 최초경합을 탐지하는 장치에 있어서,

읽기사건이 상기 최초경합의 읽기 후보사건으로 결정되면, 상기 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 플래그(flag)를 설정된 값으로 갱신하고, 상기 플래그 값을 이용하여 이후에 발생하는 접근사건에 대한 경합 탐지를 수행하는 최초경합 탐지부를 포함하며,

상기 최초경합 탐지부는,

상기 공유 변수에 대한 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 접근 역사 및 최초경합사건 역사 내에 접근사건들을 저장하는 제1 경합탐지부;

현재의 접근사건과 상기 최초경합사건 역사 내에 수집된 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 상기 최초경합사건 역사를 갱신하는 제2 경합탐지부; 및

상기 접근 역사 및 최초경합사건 역사 내에 접근사건들을 저장하는 저장부를 포함하고,

상기 제2 경합탐지부는 현재의 읽기사건이 상기 최초경합사건 역사 내에 저장된 쓰기사건과 경합을 구성하면, 상기 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 상기 플래그를 설정된 값으로 갱신하며,

상기 제2 경합탐지부는,

상기 플래그가 상기 설정된 값으로 갱신되면, 상기 스레드에서의 최초경합 탐지를 중지하지 않고, 상기 스레드에서 상기 읽기사건 이후에 발생하는 읽기사건만을 반환하여, 상기 스레드에서 상기 읽기사건 이후에 발생하는 쓰기사건에 대하여 최초경합 탐지를 수행함을 특징으로 하는 공유메모리 프로그램의 최초경합 탐지 장치.

청구항 4

공유메모리 프로그램에서 공유 변수에 대한 최초경합을 탐지하는 장치에 있어서,

읽기사건이 상기 최초경합의 읽기 후보사건으로 결정되면, 상기 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 플래그(flag)를 설정된 값으로 갱신하고, 상기 플래그 값을 이용하여 이후에 발생하는 접근사건에 대한 경합 탐지를 수행하는 최초경합 탐지부를 포함하며,

상기 최초경합 탐지부는,

상기 공유 변수에 대한 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 접근 역사 및 최초경합사건 역사 내에 접근사건들을 저장하는 제1 경합탐지부;

현재의 접근사건과 상기 최초경합사건 역사 내에 수집된 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 상기 최초경합사건 역사를 갱신하는 제2 경합탐지부; 및

상기 접근 역사 및 최초경합사건 역사 내에 접근사건들을 저장하는 저장부를 포함하고,

상기 제2 경합탐지부는 현재의 읽기사건이 상기 최초경합사건 역사 내에 저장된 쓰기사건과 경합을 구성하면, 상기 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 상기 플래그를 설정된 값으로 갱신하며,

상기 제1 경합탐지부는,

상기 공유 변수에 대한 읽기사건과 상기 저장부에 저장된 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 상기 접근 역사 및 최초경합사건 역사 내에 읽기사건을 저장하는 제1 읽기 경합탐지부; 및

상기 공유 변수에 대한 쓰기사건과 상기 저장부에 저장된 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 상기 접근 역사 및 최초경합사건 역사 내에 쓰기사건을 저장하는 제1 쓰기 경합탐지부를 포함함을 특징으로 하는 공유메모리 프로그램의 최초경합 탐지 장치.

청구항 5

제4 항에 있어서,

상기 제1 읽기 경합탐지부는,

현재의 읽기사건과 상기 접근 역사 내의 쓰기사건 간의 선행 관계에 기초하여 경합 여부를 판단하는 제1 읽기 경합판단부;

상기 현재의 읽기사건과 상기 접근 역사 내의 읽기사건 간의 좌우 관계에 기초하여 읽기 접근역사를 갱신하는 제1 읽기 접근역사 갱신부; 및

상기 현재의 읽기사건이 상기 접근 역사 내의 쓰기사건과 경합이 아니면 반환하고, 경합이면 상기 최초경합사건 역사 내의 읽기사건들과의 좌우 관계에 기초하여 읽기 최초경합사건 역사를 갱신하는 제1 읽기 최초경합사건 역사 갱신부를 포함함을 특징으로 하는 공유메모리 프로그램의 최초경합 탐지 장치.

청구항 6

제4 항에 있어서,

상기 제1 쓰기 경합탐지부는,

현재의 쓰기사건과 상기 접근 역사 내의 접근사건들 간의 선행 관계에 기초하여 경합 여부를 판단하는 제1 쓰기 경합판단부;

쓰기 접근역사를 상기 현재의 쓰기사건으로 갱신하는 제1 쓰기 접근역사 갱신부; 및

상기 현재의 쓰기사건이 경합을 구성하지 않으면 반환하고, 경합을 구성하면 상기 최초경합사건 역사 내의 쓰기 사건들과의 좌우 관계에 기초하여 쓰기 최초경합사건 역사를 갱신한 후 상기 현재의 쓰기사건이 발생한 스레드에서의 경합탐지를 중지하는 제1 쓰기 최초경합사건 역사 갱신부를 포함함을 특징으로 하는 공유메모리 프로그램의 최초경합 탐지 장치.

청구항 7

공유메모리 프로그램에서 공유 변수에 대한 최초경합을 탐지하는 장치에 있어서,

읽기사건이 상기 최초경합의 읽기 후보사건으로 결정되면, 상기 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 플래그(flag)를 설정된 값으로 갱신하고, 상기 플래그 값을 이용하여 이후에 발생하는 접근사건에 대한 경합 탐지를 수행하는 최초경합 탐지부를 포함하며,

상기 최초경합 탐지부는,

상기 공유 변수에 대한 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 접근 역사 및 최초경합사건 역사 내에 접근사건들을 저장하는 제1 경합탐지부;

현재의 접근사건과 상기 최초경합사건 역사 내에 수집된 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 상기 최초경합사건 역사를 갱신하는 제2 경합탐지부; 및

상기 접근 역사 및 최초경합사건 역사 내에 접근사건들을 저장하는 저장부를 포함하고,

상기 제2 경합탐지부는 현재의 읽기사건이 상기 최초경합사건 역사 내에 저장된 쓰기사건과 경합을 구성하면, 상기 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 상기 플래그를 설정된 값으로 갱신하며,

상기 제2 경합탐지부는,

현재의 읽기사건과 상기 최초경합사건 역사 내에 저장된 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 상기 최초경합사건 역사를 갱신하는 제2 읽기 경합탐지부; 및

현재의 쓰기사건과 상기 최초경합사건 역사 내에 수집된 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 상기 최초경합사건 역사를 갱신하는 제2 쓰기 경합탐지부를 포함함을 특징으로 하는 공유메모리 프로그램의 최초경합 탐지 장치.

청구항 8

제7 항에 있어서,

상기 제2 읽기 경합탐지부는,

현재의 스레드에서 읽기 후보사건이 발생하였는지 판단하는 읽기후보 발생 판단부;

상기 현재의 스레드에서 읽기 후보사건이 발생하였으면, 상기 현재의 읽기사건이 상기 최초경합사건 역사 내에 수집된 쓰기사건과 경합을 구성하는지 판단하는 제2 읽기 경합판단부;

상기 현재의 읽기사건이 경합을 구성하지 않으면 반환하고, 경합을 구성하면 상기 현재의 읽기사건과 상기 최초경합사건 역사 내에 수집된 접근사건들 간의 좌우 관계에 기초하여 읽기 최초경합사건 역사를 갱신하는 제2 읽기 최초경합사건 역사 갱신부;

상기 현재의 읽기사건을 최초경합 후보사건으로 백업하는 읽기 후보사건 백업부; 및

상기 현재의 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 상기 플래그를 설정된 값으로 갱신하는 플래그 갱신부를 포함함을 특징으로 하는 공유메모리 프로그램의 최초경합 탐지 장치.

청구항 9

제7 항에 있어서,

상기 제2 쓰기 경합탐지부는,

상기 현재의 쓰기사건과 상기 최초경합사건 역사 내의 접근사건들 간의 선행 관계에 기초하여 경합 여부를 판단하는 제2 쓰기 경합판단부;

상기 현재의 쓰기사건이 경합을 구성하지 않으면 반환하고, 경합을 구성하면 상기 최초경합사건 역사 내의 쓰기사건들과의 좌우 관계에 기초하여 쓰기 최초경합사건 역사를 갱신하는 제2 쓰기 최초경합사건 역사 갱신부; 및

상기 현재의 쓰기사건을 최초경합 후보사건으로 백업하는 쓰기 후보사건 백업부를 포함함을 특징으로 하는 공유메모리 프로그램의 최초경합 탐지 장치.

청구항 10

공유메모리 프로그램에서 공유 변수에 대한 최초경합을 탐지하는 방법에 있어서,

읽기사건이 상기 최초경합에 대한 읽기 후보사건으로 결정되면, 상기 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 플래그(flag)를 설정된 값으로 갱신하고, 상기 플래그 값을 이용하여 이후에 발생하는 접근사건에 대한 최초경합 탐지를 수행하는 단계를 포함하며,

상기 최초경합 탐지를 수행하는 단계는,

상기 플래그가 상기 설정된 값으로 갱신되면, 상기 스레드에서의 최초경합 탐지를 중지하지 않고, 상기 스레드

에서 상기 읽기사건 이후에 발생하는 읽기사건만을 반환하여, 상기 스레드에서 상기 읽기사건 이후에 발생하는 쓰기사건에 대하여 최초경합 탐지를 수행함을 특징으로 하는 공유메모리 프로그램의 최초경합 탐지 방법.

청구항 11

삭제

청구항 12

공유메모리 프로그램에서 공유 변수에 대한 최초경합을 탐지하는 방법에 있어서,

읽기사건이 상기 최초경합에 대한 읽기 후보사건으로 결정되면, 상기 읽기사건이 발생한 스레드에서 읽기 후보 사건이 발생하였음을 나타내는 플래그(flag)를 설정된 값으로 갱신하고, 상기 플래그 값을 이용하여 이후에 발생하는 접근사건에 대한 최초경합 탐지를 수행하는 단계를 포함하며,

상기 최초경합 탐지를 수행하는 단계는,

상기 공유 변수에 대한 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 접근 역사 및 최초경합사건 역사 내에 접근사건들을 수집하는 제1 단계; 및

현재의 접근사건과 상기 최초경합사건 역사 내에 수집된 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 상기 최초경합사건 역사를 갱신하는 제2 단계를 포함하고,

상기 제2 단계는,

현재의 읽기사건이 상기 최초경합사건 역사 내에 수집된 쓰기사건과 경합을 구성하면, 상기 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 상기 플래그를 설정된 값으로 갱신하는 단계; 및

상기 플래그가 상기 설정된 값으로 갱신되면, 상기 스레드에서의 최초경합 탐지를 중지하지 않고, 상기 스레드에서 상기 읽기사건 이후에 발생하는 읽기사건만을 반환하여, 상기 스레드에서 상기 읽기사건 이후에 발생하는 쓰기사건에 대하여 최초경합 탐지를 수행하는 단계를 포함함을 특징으로 하는 공유메모리 프로그램의 최초경합 탐지 방법.

청구항 13

공유메모리 프로그램에서 공유 변수에 대한 최초경합을 탐지하는 방법에 있어서,

읽기사건이 상기 최초경합에 대한 읽기 후보사건으로 결정되면, 상기 읽기사건이 발생한 스레드에서 읽기 후보 사건이 발생하였음을 나타내는 플래그(flag)를 설정된 값으로 갱신하고, 상기 플래그 값을 이용하여 이후에 발생하는 접근사건에 대한 최초경합 탐지를 수행하는 단계를 포함하며,

상기 최초경합 탐지를 수행하는 단계는,

상기 공유 변수에 대한 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 접근 역사 및 최초경합사건 역사 내에 접근사건들을 수집하는 제1 단계; 및

현재의 접근사건과 상기 최초경합사건 역사 내에 수집된 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 상기 최초경합사건 역사를 갱신하는 제2 단계를 포함하고,

상기 제2 단계는,

현재의 읽기사건이 상기 최초경합사건 역사 내에 수집된 쓰기사건과 경합을 구성하면, 상기 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 상기 플래그를 설정된 값으로 갱신하며,

상기 제1 단계는,

현재의 읽기사건과 접근 역사 내의 쓰기사건 간의 선행 관계에 기초하여 경합 여부를 판단하는 단계;

상기 현재의 읽기사건과 상기 접근 역사 내의 읽기사건 간의 좌우 관계에 기초하여 읽기 접근역사를 갱신하는 단계; 및

상기 현재의 읽기사건이 상기 접근 역사 내의 쓰기사건과 경합을 구성하지 않으면 반환하고, 경합을 구성하면 상기 최초경합사건 역사 내의 읽기사건들과의 좌우 관계에 기초하여 읽기 최초경합사건 역사를 갱신하는 단계를

포함함을 특징으로 하는 공유메모리 프로그램의 최초경합 탐지 방법.

청구항 14

공유메모리 프로그램에서 공유 변수에 대한 최초경합을 탐지하는 방법에 있어서,

읽기사건이 상기 최초경합에 대한 읽기 후보사건으로 결정되면, 상기 읽기사건이 발생한 스레드에서 읽기 후보 사건이 발생하였음을 나타내는 플래그(flag)를 설정된 값으로 갱신하고, 상기 플래그 값을 이용하여 이후에 발생하는 접근사건에 대한 최초경합 탐지를 수행하는 단계를 포함하며,

상기 최초경합 탐지를 수행하는 단계는,

상기 공유 변수에 대한 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 접근 역사 및 최초경합사건 역사 내에 접근사건들을 수집하는 제1 단계; 및

현재의 접근사건과 상기 최초경합사건 역사 내에 수집된 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 상기 최초경합사건 역사를 갱신하는 제2 단계를 포함하고,

상기 제2 단계는,

현재의 읽기사건이 상기 최초경합사건 역사 내에 수집된 쓰기사건과 경합을 구성하면, 상기 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 상기 플래그를 설정된 값으로 갱신하며,

상기 제1 단계는,

현재의 쓰기사건과 접근 역사 내의 접근사건들 간의 선행 관계에 기초하여 경합 여부를 판단하고, 쓰기 접근 역사를 갱신하는 단계;

상기 현재의 쓰기사건이 경합을 구성하지 않으면 반환하고, 경합을 구성하면 상기 최초경합사건 역사 내의 쓰기 사건들과의 좌우 관계에 기초하여 쓰기 최초경합사건 역사를 갱신하는 단계; 및

상기 현재의 쓰기사건이 발생한 스레드에서의 경합탐지를 중지하는 단계를 포함함을 특징으로 하는 공유메모리 프로그램의 최초경합 탐지 방법.

청구항 15

공유메모리 프로그램에서 공유 변수에 대한 최초경합을 탐지하는 방법에 있어서,

읽기사건이 상기 최초경합에 대한 읽기 후보사건으로 결정되면, 상기 읽기사건이 발생한 스레드에서 읽기 후보 사건이 발생하였음을 나타내는 플래그(flag)를 설정된 값으로 갱신하고, 상기 플래그 값을 이용하여 이후에 발생하는 접근사건에 대한 최초경합 탐지를 수행하는 단계를 포함하며,

상기 최초경합 탐지를 수행하는 단계는,

상기 공유 변수에 대한 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 접근 역사 및 최초경합사건 역사 내에 접근사건들을 수집하는 제1 단계; 및

현재의 접근사건과 상기 최초경합사건 역사 내에 수집된 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 상기 최초경합사건 역사를 갱신하는 제2 단계를 포함하고,

상기 제2 단계는,

현재의 읽기사건이 상기 최초경합사건 역사 내에 수집된 쓰기사건과 경합을 구성하면, 상기 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 상기 플래그를 설정된 값으로 갱신하며,

상기 제2 단계는,

현재의 스레드에서 읽기 후보사건이 발생하였는지 판단하는 단계;

상기 현재의 스레드에서 읽기 후보사건이 발생하였으면, 현재의 읽기사건이 상기 최초경합사건 역사 내에 수집된 쓰기사건과 경합을 구성하는지 판단하는 단계;

상기 현재의 읽기사건이 경합을 구성하지 않으면 반환하고, 경합을 구성하면 상기 현재의 읽기사건과 상기 최초경합사건 역사 내에 수집된 읽기사건들 간의 좌우 관계에 기초하여 상기 최초경합사건 역사를 갱신하는 단계;

상기 현재의 읽기사건을 최초경합 후보사건으로 백업하는 단계; 및

상기 현재의 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 상기 플래그를 설정된 값으로 갱신하는 단계를 포함함을 특징으로 하는 공유메모리 프로그램의 최초경합 탐지 방법.

청구항 16

공유메모리 프로그램에서 공유 변수에 대한 최초경합을 탐지하는 방법에 있어서,

읽기사건이 상기 최초경합에 대한 읽기 후보사건으로 결정되면, 상기 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 플래그(flag)를 설정된 값으로 갱신하고, 상기 플래그 값을 이용하여 이후에 발생하는 접근사건에 대한 최초경합 탐지를 수행하는 단계를 포함하며,

상기 최초경합 탐지를 수행하는 단계는,

상기 공유 변수에 대한 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 접근 역사 및 최초경합사건 역사 내에 접근사건들을 수집하는 제1 단계; 및

현재의 접근사건과 상기 최초경합사건 역사 내에 수집된 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 상기 최초경합사건 역사를 갱신하는 제2 단계를 포함하고,

상기 제2 단계는,

현재의 읽기사건이 상기 최초경합사건 역사 내에 수집된 쓰기사건과 경합을 구성하면, 상기 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 상기 플래그를 설정된 값으로 갱신하며,

상기 제2 단계는,

현재의 쓰기사건과 상기 최초경합사건 역사 내의 접근사건들 간의 선행 관계에 기초하여 경합 여부를 판단하는 단계;

상기 현재의 쓰기사건이 경합을 구성하지 않으면 반환하고, 경합을 구성하면 상기 최초경합사건 역사 내의 쓰기사건들과의 좌우 관계에 기초하여 쓰기 최초경합사건 역사를 갱신하는 단계;

상기 현재의 쓰기사건을 최초경합 후보사건으로 백업하는 단계; 및

상기 현재의 쓰기사건이 발생한 스레드에서의 최초경합 탐지를 중지하는 단계를 포함함을 특징으로 하는 공유메모리 프로그램의 최초경합 탐지 방법.

청구항 17

제10 항, 제12 항 내지 제16 항 중 어느 한 항에 기재된 최초경합 탐지 방법을 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체.

명세서

기술분야

[0001] 본 발명은 공유메모리 프로그램의 최초경합을 탐지하는 방법 및 이를 수행하는 공유메모리 프로그램의 최초경합 탐지 장치에 관한 것이다.

배경기술

[0002] 멀티코어(Multi-Core)의 보편화로 인해, 병렬프로그래밍이 일반화되고 있다. 내포병렬성을 갖는 공유메모리 프로그램에서 경합(race)은 프로그램상에 의도하지 않은 비결정적인 수행 결과를 초래하는 오류 중의 하나로서, 프로그램의 신뢰성을 높이기 위한 디버깅(debugging)을 위해 이러한 경합을 탐지할 필요가 있다. 병렬프로그램에서 발생할 수 있는 경합 중에서 가장 먼저 발생하고 다른 경합으로부터 영향을 받지 않은 경합을 최초경합(first race)이라고 한다. 이러한 최초경합의 탐지는 최초경합으로부터 영향받은 다른 경합들을 사라지게 하거나 새로이 발생하게 할 수 있으므로 중요하다.

[0003] 공유메모리 병렬프로그램의 최초경합을 탐지하기 위해서 최초경합의 가능성이 있는 후보사건들을 이용하는 기존의 방법들은 두 번의 프로그램 수행중 후보사건을 수집하고 이 후보사건들 간의 경합을 최초경합으로 보고한다.

기존의 방법들 중 순서적 동기화를 지원하는 방법은 현재의 접근사건과 접근 역사에 저장된 이전 사건들의 논리적 병행성 관계(logical concurrency relation)를 확인하여 최초경합을 탐지한다. 이 방법은 최초 얽힘을 포함한 최초경합의 탐지를 보장하는 잇점을 갖지만, 프로그램 수행중 각 공유 변수마다 프로그램의 최대병렬성(Maximum Parallelism)에 의존적인 크기의 접근 역사(Access History)를 유지하므로, 비효율적인 수행시간(execution time)과 기억공간(memory space)을 요구하게 된다.

[0004] 기존의 방법들 중 접근 역사를 상수적 크기로 유지하는 방법은 최초 얽힘에 포함된 경합들(first tangled races) 중의 일부를 최초경합으로 탐지하지 못할 수 있으며, 최악의 경우 최초 얽힘에 포함된 단 하나의 경합만을 탐지할 수 있다. 따라서, 이 방법은 최초 얽힘의 검증을 보장할 수 없으므로, 얽힌 경합들(tangled races)을 갖는 프로그램의 경합 없는 위치(race-free portion)를 결정하기 위해 예측할 수 없는 많은 수행들의 반복을 요구하게 된다.

발명의 내용

해결하려는 과제

[0005] 본 발명이 해결하고자 하는 과제는 공유메모리 프로그램의 최초경합을 효율적이면서도 정확하게 탐지하는 방법 및 이를 수행하는 장치를 제공하는데 있다.

[0006] 본 발명이 해결하고자 하는 다른 과제는 공유메모리 프로그램에서 접근 역사를 최대 병렬성에 의존하지 않는 상수적 크기로 유지하면서도, 최초 얽힘에 포함된 최초경합들을 정확하게 탐지할 수 있는 최초경합 탐지 방법 및 이를 수행하는 장치를 제공하는데 있다.

[0007] 본 발명이 해결하고자 하는 다른 과제는 최초경합을 효율적이면서도 정확하게 탐지하는 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체를 제공하는데 있다.

[0008] 본 발명이 해결하고자 하는 과제는 이상에서 언급된 과제로 제한되지 않는다. 언급되지 않은 다른 기술적 과제들은 이하의 기재로부터 당업자에게 명확하게 이해될 수 있을 것이다.

과제의 해결 수단

[0009] 상기 과제를 해결하기 위한 본 발명에 따른 공유메모리 프로그램의 최초경합 탐지 장치는 공유메모리 프로그램에서 공유 변수에 대한 최초경합을 탐지하는 장치에 있어서, 읽기사건이 상기 최초경합의 읽기 후보사건으로 결정되면, 상기 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 플래그(flag)를 설정된 값으로 갱신하고, 상기 플래그 값을 이용하여 이후에 발생하는 접근사건에 대한 경합 탐지를 수행하는 최초경합 탐지부를 포함함을 특징으로 한다.

[0010] 상기 과제를 해결하기 위한 본 발명의 다른 양상에 따른 공유메모리 프로그램의 최초경합 탐지 방법은 공유메모리 프로그램에서 공유 변수에 대한 최초경합을 탐지하는 방법에 있어서, 읽기사건이 상기 최초경합에 대한 읽기 후보사건으로 결정되면, 상기 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 플래그(flag)를 설정된 값으로 갱신하고, 상기 플래그 값을 이용하여 이후에 발생하는 접근사건에 대한 경합 탐지를 수행함을 특징으로 한다.

발명의 효과

[0011] 본 발명의 일 실시예에 따른 최초경합 탐지 방법 및 장치에 의하면, 이하의 효과를 얻을 수 있다.

[0012] 첫째, 공유메모리 프로그램의 최초경합을 효율적이면서도 정확하게 탐지할 수 있다.

[0013] 둘째, 본 발명의 일 실시예에 따른 최초경합 탐지 방법 및 장치에 의하면, 최초 얽힘에 포함된 최초경합들을 정확하게 탐지할 수 있다.

[0014] 셋째, 본 발명의 일 실시예에 따른 최초경합 탐지 방법 및 장치에 의하면, 공유메모리 프로그램에서 접근 역사 및 최초경합사건 역사를 최대 병렬성에 의존하지 않는 상수적 크기로 유지하여 수행시간(execution time) 및 기억공간(memory space)의 측면에서 효율적이다.

도면의 간단한 설명

[0015] 도 1은 부분순서 수행 그래프(Partial Order Execution Graph; POEG)의 예시도이다.

도 2는 본 발명의 일 실시예에 따른 최초경합 탐지 장치의 구성도이다.

도 3은 도 2에 도시된 최초경합 탐지부의 예시적인 구성도이다.

도 4는 도 3에 도시된 제1 경합탐지부의 예시적인 구성도이다.

도 5는 도 3에 도시된 제2 경합탐지부의 예시적인 구성도이다.

도 6은 본 발명의 일 실시예에 따른 최초경합 탐지 방법의 흐름도이다.

도 7은 도 6에 도시된 단계 61의 읽기사건에 대한 예시적인 흐름도이다.

도 8은 도 6에 도시된 단계 61의 쓰기사건에 대한 예시적인 흐름도이다.

도 9는 도 6에 도시된 단계 62 내지 64의 읽기사건에 대한 예시적인 흐름도이다.

도 10은 도 6에 도시된 단계 62 내지 64의 쓰기사건에 대한 예시적인 흐름도이다.

도 11은 또 다른 부분순서 수행 그래프(Partial Order Execution Graph; POEG)의 예시도이다.

발명을 실시하기 위한 구체적인 내용

[0016] 본 발명의 다른 이점 및 특징, 그리고 그것들을 달성하는 방법은 첨부되는 도면과 함께 상세하게 후술 되는 실시 예를 참조하면 명확해질 것이다. 그러나 본 발명은 이하에서 개시되는 실시 예에 한정되는 것이 아니라 서로 다른 다양한 형태로 구현될 수 있으며, 단지 본 실시 예는 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.

[0017] 만일 정의되지 않더라도, 여기서 사용되는 모든 용어들(기술 혹은 과학 용어들을 포함)은 이 발명이 속한 종래 기술에서 보편적 기술에 의해 일반적으로 수용되는 것과 동일한 의미를 가진다. 일반적인 사전들에 의해 정의된 용어들은 관련된 기술 그리고/혹은 본 출원의 본문에 의미하는 것과 동일한 의미를 갖는 것으로 해석될 수 있고, 그리고 여기서 명확하게 정의된 표현이 아니더라도 개념화되거나 혹은 과도하게 형식적으로 해석되지 않을 것이다.

[0018] 본 명세서에서 사용된 용어는 실시 예들을 설명하기 위한 것이며 본 발명을 제한하고자 하는 것은 아니다. 본 명세서에서, 단수형은 문구에서 특별히 언급하지 않는 한 복수형도 포함한다. 명세서에서 사용되는 '포함한다' 및/또는 이 동사의 다양한 활용형들 예를 들어, '포함', '포함하는', '포함하고', '포함하며' 등은 언급된 조성, 성분, 구성요소, 단계, 동작 및/또는 소자는 하나 이상의 다른 조성, 성분, 구성요소, 단계, 동작 및/또는 소자의 존재 또는 추가를 배제하지 않는다.

[0019] 한편, 본 명세서 전체에서 사용되는 '~부', '~기', '~블록', '~모듈' 등의 용어는 적어도 하나의 기능이나 동작을 처리하는 단위를 의미할 수 있다. 예를 들어 소프트웨어, FPGA 또는 ASIC과 같은 하드웨어 구성요소를 의미할 수 있다. 그렇지만 '~부', '~기', '~블록', '~모듈' 등이 소프트웨어 또는 하드웨어에 한정되는 의미는 아니다. '~부', '~기', '~블록', '~모듈'은 어드레싱할 수 있는 저장 매체에 있도록 구성될 수도 있고 하나 또는 그 이상의 프로세서들을 재생시키도록 구성될 수도 있다.

[0020] 따라서, 일 예로서 '~부', '~기', '~블록', '~모듈'은 소프트웨어 구성요소들, 객체지향 소프트웨어 구성요소들, 클래스 구성요소들 및 태스크 구성요소들과 같은 구성요소들과, 프로세스들, 함수들, 속성들, 프로시저들, 서브루틴들, 프로그램 코드의 세그먼트들, 드라이버들, 펌웨어, 마이크로 코드, 회로, 데이터, 데이터베이스, 데이터 구조들, 테이블들, 어레이들 및 변수들을 포함할 수 있다. 구성요소들과 '~부', '~기', '~블록', '~모듈'들 안에서 제공되는 기능은 더 작은 수의 구성요소들 및 '~부', '~기', '~블록', '~모듈'들로 결합되거나 추가적인 구성요소들과 '~부', '~기', '~블록', '~모듈'들로 더 분리될 수 있다.

[0021] 본 발명의 일 실시예에 따른 최초경합 탐지 장치는 읽기사건이 최초경합에 대한 읽기 후보사건으로 결정되면, 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 플래그(flag)를 설정된 값으로 갱신하고, 플래그 값을 이용하여 이후에 발생하는 접근사건에 대한 경합 탐지를 수행하는 최초경합 탐지부를 포함함으로써, 내포병렬성을 가진 공유메모리 프로그램에서 최종 얽힘에 속하는 최초경합을 효율적이면서 정확하게 탐지할 수 있다.

[0022] 본 발명의 일 실시예에 따른 최초경합 탐지 방법은 공유 변수에 대한 접근사건들 간의 선행 관계 및 좌우 관계

에 기초하여, 접근 사건 및 최초경합사건 역사 내에 접근사건들을 수집하는 제1 단계와, 현재의 접근사건과 제1 단계를 통해 최초경합사건 역사 내에 수집된 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 최초경합사건 역사를 갱신하는 제2 단계를 포함하되, 제2 단계는 현재의 읽기사건이 최초경합사건 역사 내에 수집된 쓰기 사건과 경합을 구성하면 현재의 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 플래그(flag)를 설정된 값으로 갱신하는 단계를 포함함으로써, 내포병렬성을 가진 공유메모리 프로그램의 최초 얽힘에 속하는 최초경합을 효율적이면서 정확하게 탐지할 수 있다.

[0023] 본 발명의 일 실시예에 따른 최초경합 탐지 방법은 제2 단계에서 플래그가 설정된 값으로 갱신되면, 현재의 스레드에서의 최초경합 탐지를 중지하지 않고, 스레드에서 해당 읽기사건 이후에 발생하는 읽기사건만을 반환하여, 해당 읽기사건 이후에 발생하는 쓰기사건에 대하여 제2 단계 최초경합 탐지를 수행함으로써, 내포병렬성을 가진 공유메모리 프로그램의 최초얽힘에 속하는 최초경합을 효율적이면서 정확하게 탐지할 수 있다.

[0024] 먼저, 본 발명의 실시예에서 사용되는 용어에 대하여 설명한다. 공유메모리 병렬프로그램에서 경합(race)은 병행적으로 수행되는 스레드들이 공유 변수를 사용하고, 적절한 동기화 없이 적어도 하나 이상의 쓰기사건으로 접근할 때 일어날 수 있는 접근 오류이다. 공유메모리 프로그램의 수행이나, 스레드들 사이의 동시 관계(concurrency relation)는 방향성이 있는 비순환 그래프인 부분순서 수행 그래프(Partial Order Execution Graph)(이하, 'POEG'로 칭함)로 나타낼 수 있다. POEG 는 병렬프로그램의 수행 시에 스레드들 간의 부분적 순서(partial order) 관계를 나타낸다.

[0025] 도 1은 예시적인 POEG 를 나타낸다. 도 1을 참조하면, POEG 에서 정점(vertex)은 병렬 스레드(thread)의 생성(fork)과 합류(join) 동작을 나타내며, 정점들 사이의 간선(arc)은 정점에서 생성되거나 합류된 스레드를 나타낸다. 동일한 공유 변수에 접근하는 접근사건(이하, '사건'으로 칭해질 수 있음)은 해당 스레드의 간선 위에 점으로 표시된다. 각 접근사건은 읽기사건의 경우 'r', 쓰기사건의 경우 'w' 로 표기되어 그 유형이 구분될 수 있다. 'r' 또는 'w' 뒤의 숫자는 프로그램 수행 시에 해당 접근사건이 관찰되는 순서, 즉 접근사건들의 발생 순서를 나타낸다.

[0026] 만약, POEG 에서 사건 e_i 로부터 사건 e_j 에 이르기까지 경로가 존재하면, 두 사건은 순서화(ordered)되었다고 하며, 이러한 경우 사건 e_i 는 사건 e_j 에 선행(happened-before)한다. 만약, 두 사건 중 어느 한 사건도 다른 사건을 선행하지 않으면, 즉 두 사건 사이에 경로가 존재하지 않으면, 이 두 사건은 병행(concurrent)하다. 그리고 임의의 사건 e_i 가 e_j 보다 선행하고 사건 e_i , e_j 의 각 내포수준이 m , n 일 때, $m < n$ 을 만족하면 e_j 가 e_i 에 내포(nested)된다고 한다.

[0027] 도 1에 도시된 POEG 를 예로 들면, r_0 로부터 w_6 를 향해 경로가 존재하므로, r_0 는 w_6 에 선행한다. 그리고, r_0 와 w_7 는 이들 사이에 경로가 존재하므로 r_0 는 w_7 에 선행하고, w_7 의 내포수준이 '2'로서 r_0 의 내포수준 '1'보다 크기 때문에 w_7 은 r_0 에 내포된다. 그러나, 예를 들어 r_0 와 w_{10} 의 사이, r_8 과 w_7 의 사이에는 경로가 없으므로, r_0 와 w_{10} 는 병행하고, r_8 과 w_7 도 병행하다.

[0028] 하나의 공유 변수에 대한 두 사건 중에서 적어도 하나가 쓰기사건이면, 이들 두 사건은 '대립하는(conflicting) 사건'이다. 만약, 두 개의 접근사건이 대립하는 사건이고, 서로 병행하면 두 접근사건은 '경합(race)'에 해당한다. 즉, 병행 관계에 있는 두 개의 접근사건 e_i , e_j 중 적어도 하나 이상이 쓰기사건이고, 하나의 공유 변수에 대해 두 개의 접근사건이 적절한 동기화 없이 나타날 때 발생하는 오류를 '경합(race)'이라 할 수 있으며, 이를 ' e_i-e_j '로 표시할 수 있다.

[0029] 여기에서 임의의 사건 e_i 가 e_j 에 선행하고, e_i 가 경합 R_i 에 포함되면, e_j 는 e_i 나 R_i 에 의해 '영향받은(affected) 사건'이다. 임의의 경합 ' e_i-e_j '을 구성하는 두 사건 e_i , e_j 중에서 어느 하나의 사건도 다른 사건에 의해 영향받은 사건이 아니면, 이러한 경합 ' e_i-e_j '은 '영향받지 않은(unaffected)' 경합이다. 만약, 경합 ' e_i-e_j '을 구성하는 두 사건 e_i , e_j 중에서 하나의 사건만 다른 사건에 의해 영향받으면, 경합 ' e_i-e_j '은 '부분적으로 영향받은(partially affected)' 경합이다.

[0030] 얽힘(tangle)은 부분적으로 영향받은 경합들의 집합으로서, 얽힘에 속하는 경합들 중 경합 ' e_i-e_j '을 구성하는 두 사건 e_i , e_j 중에서 어느 하나의 사건만이 동일한 얽힘에 속하는 경합을 구성하는 사건 e_k 에 의해 영향받는 조건을 만족하는 경합들의 집합을 의미할 수 있다. 얽힘에 속한 임의의 부분적으로 영향받은 경합을 '얽힌 경합

(tangled race)'이라 한다. 그리고, 모든 접근사건들이 기껏해야 하나의 얽힌 경합에 영향받은 경합들의 집합인 얽힘을 '최초얽힘(first tangle)'이라 한다. 이러한 영향받지 않은 경합이거나, 최초얽힘에 속하는 경합을 '최초경합(first race)'이라 한다.

[0031] 도 1을 참조하면, 도 1에 도시된 POEG 는 점선으로 나타내어진 다수의 경합들을 포함하지만, $\{r9-w6, r9-w7\}$ 는 $\{r8-w6, r0-w10\}$ 으로부터 영향받은 경합이므로, 실선으로 표시된 $\{r0-w10, r8-w7, r8-w6\}$ 만이 최초경합이며 이들은 얽힘을 이루고 있다. 최초경합은 이에 의하여 영향받은 다른 경합들이 사라질 수 있거나 새로이 발생할 수 있으므로, 효과적인 디버깅(debugging)을 위해서는 이러한 최초경합을 탐지하여 제거할 필요가 있다.

[0032] 최초경합 접근사건은 최초경합을 구성하는 두 개의 접근사건들 중의 임의의 하나의 사건을 의미할 수 있으며, 읽기 최초경합 접근사건, 또는 쓰기 최초경합 접근사건, 또는 읽기-쓰기 최초경합 접근사건 중의 어느 하나에 해당할 수 있다. 만약, 경합을 구성하는 접근사건 e_i 에 선행하는 다른 접근사건이 존재하지 않거나 선행하는 다른 접근사건이 존재하더라도 경합에 속하지 않을 때, 사건 e_i 는 '최초경합 접근사건' 이다. 만약, 경합을 구성하는 쓰기사건 e_i 에 선행하는 쓰기사건이 존재하지 않고, e_i 에 선행하는 읽기사건이 존재하고, e_i 와 병행 관계를 갖는 쓰기 최초경합 접근사건이 존재하지 않으면, 사건 e_i 은 '읽기-쓰기 최초경합 접근사건' 이다.

[0033] 도 1에 도시된 POEG 를 예로 들면, 읽기사건 $r0$ 는 쓰기사건 $w10$ 과 경합을 구성하고, $r0$ 에 선행하는 경합을 구성하는 다른 접근사건이 존재하지 않으므로, 사건 $r0$ 는 읽기 최초경합 접근사건이다. 쓰기사건 $w6$ 은 선행하는 쓰기 접근사건이 존재하지 않고, $w6$ 에 선행하는 읽기사건 $r0$ 가 존재하고, $w6$ 와 병행 관계를 갖는 다른 쓰기 최초경합 접근사건이 존재하지 않으므로, 사건 $w6$ 는 읽기-쓰기 최초경합 접근사건이다. $w6$ 와 병행 관계를 갖는 두 개의 쓰기사건 $w7$ 과 $w10$ 은 읽기-쓰기 최초경합 접근사건이다.

[0034] 공유메모리 프로그램의 최초경합을 탐지하기 위해 최초경합의 가능성이 있는 후보사건들을 이용하는 기존의 최초경합 탐지 방법들은 두 번의 프로그램 수행중 후보사건을 수집하고 이 후보사건들 간의 경합을 최초경합으로 보고한다. 그러나, 기존의 방법들은 두 번의 프로그램 수행 중에 각 공유 변수마다 프로그램의 최대병렬성에 의존적인 크기의 접근 역사를 유지하여 비효율적인 수행시간과 기억공간을 요구하거나, 최초 얽힘이 포함된 경우, 최초 얽힘에 속하는 모든 최초경합들의 존재를 검증하지 못할 수 있다.

[0035] 도 1의 예에 기존의 방법을 적용하면, ' $r0$ ' 과 ' $w7$ ' 이 동일한 스레드로 지정될 때, ' $r0-w10$ ' 은 최초경합으로 탐지되지만, ' $r0$ ' 가 속한 스레드에 대한 이후의 사건들의 최초경합 탐지가 중지(halt)되어, 경합 ' $r0-w10$ ' 을 포함하는 최초 얽힘에 포함되는 다른 최초경합 ' $w7-r8$ ' 을 탐지하지는 못할 수 있다. 만약, 최초 얽힘의 다른 모든 경합들을 탐지하려면, 예측할 수 없는 프로그램 수행의 무한정한 반복이 요구될 수 있다. 이에 따라, 본 발명의 실시예는 내포병렬성을 갖는 공유메모리 프로그램에서 최초 얽힘에 포함된 최초경합들을 효율적이면서 정확하게 검증할 수 있는 최초경합 탐지 장치를 제시한다.

[0036] 도 2는 본 발명의 일 실시예에 따른 최초경합 탐지 장치의 구성도이다. 도 2에 도시된 바와 같이, 본 발명의 일 실시예에 따른 최초경합 탐지 장치는 레이블링 엔진부(1)와, 최초경합 탐지부(2)를 포함한다. 레이블링 엔진부(1)는 자료 구조의 생성이나 자료 초기화, 스레드의 분기 및 종료를 위한 라이브러리, 경합탐지를 위한 읽기 및 쓰기 라이브러리(checkread, checkwrite)를 지원할 수 있다. 레이블링 엔진부(1)는 프로그램 수행중 생성된 스레드에 논리적 병행성 정보를 생성하여 스레드마다 고유한 식별자를 부여할 수 있다. 레이블링 엔진부(1)는 예를 들어, 내포병렬성을 갖는 프로그램의 경합 탐지에 적합한 NR(Nest Region) 레이블링 기법을 사용할 수 있다. 본 발명의 실시예가 속하는 기술분야에서 NR 레이블링 기법은 주지되어 있으므로, 이에 대한 구체적인 설명은 생략하기로 한다.

[0037] 최초경합 탐지부(2)는 접근사건들 간의 선행 관계 및 좌우 관계를 검사하여, 접근 역사 및 최초경합사건 역사 내에 접근사건들을 수집한 다음, 현재의 접근사건과 최초경합사건 역사 내에 수집된 접근사건들의 선행 관계 및 좌우 관계를 검사하여 최초경합사건 역사를 갱신할 수 있으며, 이 과정에서 최초경합 후보사건을 결정함으로써 최초경합을 탐지할 수 있다. 여기서, 접근사건들 간의 좌우 관계는, 예를 들어 접근사건 e_i 와 다른 사건 e_j 가 불 방식의 관계(boolean relation) ' $Leftof(e_i, e_j)$ ' 를 만족하는지 여부로부터 검사할 수 있다. 만약, 접근사건 e_i 와 e_j 가 ' $Leftof(e_i, e_j)$ ' 를 만족하면, 접근사건 e_i 는 사건 e_j 의 왼쪽에 존재한다고 할 수 있다.

[0038] 본 발명의 일 실시예에 따른 최초경합 탐지 장치는 상수적 크기의 접근 역사 및 최초경합사건 역사를 유지하여 효율적으로 최초경합을 탐지하기 위하여, '외부사건'의 정의를 활용할 수 있다. 만약, 접근사건 e_i 의 왼쪽 또

는 오른쪽에 다른 사건이 존재하지 않으면, 사건 e_i 은 다른 사건들보다 외측에서 수행되는 '외부사건' 이다. 예를 들어, 도 1에 도시된 POEG 에서, 접근사건 r_2 는 오른쪽에 임의의 사건이 존재하지 않고, r_8 은 왼쪽에 임의의 사건이 존재하지 않으므로 두 사건 r_2 , r_8 은 외부사건이다.

[0039] 또한, 본 발명의 일 실시예에 따른 최초경합 탐지 장치는 엄힘에 속하는 모든 경합들을 탐지하기 위해, '마지막 외부사건' 의 정의를 활용할 수 있다. 만약, 접근사건 e_i 에 후행(happened-after)하는 다른 접근사건이 존재하지 않을 때, 사건 e_i 는 마지막 접근사건이다. 만약, 접근사건 e_i 가 외부사건이고, 마지막 접근사건이면, 사건 e_i 는 '마지막 외부사건' 이다.

[0040] 이 때, 최초경합 탐지부(2)는 읽기사건에 해당하는 접근사건이 최초경합의 읽기 후보사건으로 결정되면, 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 플래그(flag)를 설정된 값으로 갱신할 수 있다. 플래그가 설정된 값으로 갱신되면, 최초경합 탐지부(2)는 읽기사건이 발생한 스레드에서의 최초경합 탐지를 중지하는 대신, 해당 스레드에서 이후에 발생하는 읽기사건만을 반환하고, 해당 스레드에서 이후에 발생하는 쓰기사건에 대하여는 최초경합사건 역사 내에 수집된 접근사건들과의 선행 관계 및 좌우 관계를 검사함으로써, 최초경합 탐지를 수행할 수 있다. 최초경합 탐지부(2)에 대하여는 도 3을 참조하여 보다 상세히 설명한다.

[0041] 도 3은 도 2에 도시된 최초경합 탐지부의 예시적인 구성도이다. 도 3에 도시된 바와 같이, 최초경합 탐지부(2)는 제1 경합탐지부(21), 제2 경합탐지부(22), 및 저장부(23)를 포함한다. 최초 엄힘을 포함하는 최초 경합을 검증하기 위해, 본 발명의 실시예에 따른 최초경합 탐지 장치는 두 개의 공유 데이터 구조(shared data structure)를 유지할 수 있다. 첫 번째 공유 데이터 구조는 공유 변수에 대해 세 개의 접근사건을 저장할 수 있는 접근 역사(Access History)(231)이고, 두 번째 공유 데이터 구조는 공유 변수에 대해 외부사건으로 수행되는 네 개의 접근사건을 저장할 수 있는 최초경합사건 역사(First race access History)(232)일 수 있다.

[0042] 제1 경합탐지부(21)는 공유 변수에 대한 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 저장부(23)의 접근 역사(231) 및 최초경합사건 역사(232) 내에 접근사건들을 저장할 수 있다. 제2 경합탐지부(22)는 제1 경합탐지부(21)를 통해 저장부(23)의 최초경합사건 역사(232) 내에 수집된 접근사건들을 활용하여 최초경합 탐지를 수행할 수 있다. 제2 경합탐지부(22)는 현재의 접근사건과 최초경합사건 역사(232) 내에 수집된 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 최초경합사건 역사(232)를 갱신할 수 있다. 이 때, 제2 경합탐지부(22)는 현재의 읽기사건이 최초경합사건 역사(232) 내에 저장된 쓰기사건과 경합을 구성하면, 읽기사건이 발생한 스레드에서 읽기후보사건이 발생하였음을 나타내는 플래그(flag)를 설정된 값으로 갱신한다.

[0043] 제1 경합탐지부(21)는 제1 읽기 경합탐지부(211)와, 제1 쓰기 경합탐지부(212)를 포함한다. 제1 읽기 경합탐지부(211)는 공유 변수에 대한 읽기사건과 저장부(23)의 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 접근 역사(231) 및 최초경합사건 역사(232) 내에 읽기사건을 새로 저장하거나 갱신할 수 있다. 제1 쓰기 경합탐지부(212)는 공유 변수에 대한 쓰기사건과 저장부(23)의 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 접근 역사(231) 및 최초경합사건 역사(232) 내에 쓰기사건을 새로 저장하거나 갱신할 수 있다.

[0044] 제2 경합탐지부(22)는 제2 읽기 경합탐지부(221)와, 제2 쓰기 경합탐지부(222)를 포함한다. 제2 읽기 경합탐지부(221)는 제1 읽기 경합탐지부(21)에 의해 최초경합사건 역사(232) 내에 접근사건들이 수집되면, 현재의 읽기사건과 최초경합사건 역사(232) 내에 수집된 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 최초경합사건 역사(232)를 갱신할 수 있다. 제2 쓰기 경합탐지부(222)는 현재의 쓰기사건과 최초경합사건 역사(232) 내에 수집된 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 최초경합사건 역사를 갱신할 수 있다.

[0045] 도 4는 도 3에 도시된 제1 경합탐지부의 예시적인 구성도이다. 도 4를 참조하여 제1 읽기 및 쓰기 경합 탐지부(211,212)에 대하여 보다 상세히 설명한다. 도 4에 도시된 바와 같이, 제1 읽기 경합탐지부(211)는 제1 읽기 경합판단부(2111), 제1 읽기 접근역사 갱신부(2112), 및 제1 읽기 최초경합사건역사 갱신부(2113)를 포함한다. 제1 읽기 경합판단부(2111)는 현재의 읽기사건과 접근 역사(231) 내의 쓰기사건 간의 선행 관계에 기초하여 경합 여부를 판단할 수 있다. 제1 읽기 접근역사 갱신부(2112)는 현재의 읽기사건과 접근 역사(231) 내의 접근사건 간의 좌우 관계에 기초하여 읽기 접근역사를 갱신할 수 있다. 제1 읽기 최초경합사건역사 갱신부(2113)는 현재의 읽기사건이 접근 역사(231) 내의 쓰기사건과 경합이면 반환하고, 경합이 아니면 최초경합사건 역사(232) 내의 접근사건들과의 좌우 관계에 기초하여 읽기 최초경합사건 역사를 갱신할 수 있다.

[0046] 제1 쓰기 경합탐지부(212)는 제1 쓰기 경합판단부(2121), 제1 쓰기 접근역사 갱신부(2122), 및 제1 쓰기 최초경합사건역사 갱신부(2123)를 포함한다. 제1 쓰기 경합판단부(2121)는 현재의 쓰기사건과 접근 역사(231) 내의 접근

근사건들 간의 선행 관계에 기초하여 경합 여부를 판단할 수 있다. 제1 쓰기 접근역사 갱신부(2122)는 쓰기 접근역사를 마지막 쓰기사건으로 갱신할 수 있다. 제1 쓰기 최초경합사건역사 갱신부(2123)는 현재의 쓰기사건이 경합을 구성하지 않으면 반환하고, 경합을 구성하면 최초경합사건 역사 내의 쓰기사건들과의 좌우 관계에 기초하여 쓰기 최초경합사건 역사를 갱신한 후, 현재의 쓰기사건이 발생한 스레드에서의 경합탐지를 중지할 수 있다.

[0047] 도 5는 도 3에 도시된 제2 경합탐지부의 예시적인 구성도이다. 도 5를 참조하여 제2 읽기 및 쓰기 경합 탐지부(221,222)에 대하여 보다 상세히 설명한다. 도 5에 도시된 바와 같이, 제2 읽기 경합탐지부(221)는 읽기후보 발생 판단부(2211), 제2 읽기 경합판단부(2212), 제2 읽기 최초경합사건역사 갱신부(2213), 읽기 후보사건 백업부(2214), 및 플래그 갱신부(2215)를 포함한다.

[0048] 읽기후보 발생 판단부(2211)는 현재의 스레드에서 읽기 후보사건이 발생하였는지를 판단한다. 제2 읽기 경합판단부(2212)는 현재의 스레드에서 읽기 후보사건이 발생하였으면, 현재의 읽기사건이 최초경합사건 역사(232) 내에 수집된 쓰기사건과 경합을 구성하는지를 판단할 수 있다. 제2 읽기 최초경합사건역사 갱신부(2213)는 현재의 읽기사건이 경합을 구성하지 않으면 반환하고, 경합을 구성하면 현재의 읽기 사건과 최초경합사건 역사(232) 내에 수집된 읽기사건들 간의 좌우 관계에 기초하여 읽기 최초경합사건 역사를 갱신할 수 있다.

[0049] 읽기 후보사건 백업부(2214)는 현재의 읽기사건을 최초경합 읽기 후보사건으로 백업(back-up)할 수 있다. 플래그 갱신부(2215)는 현재의 읽기사건이 발생한 스레드에서 읽기 후보사건이 발생하였음을 나타내는 플래그(flag)를 설정된 값으로 갱신할 수 있다. 이 때, 플래그는 예를 들어 '0' 또는 '1' 로 설정되거나, 'true' 또는 'faulse' 로 설정될 수 있으나, 이에 제한되지 않는다. 플래그 갱신부(2215)에 의해 플래그가 설정된 값으로 갱신되면, 읽기후보 발생 판단부(2211)는 이를 참조하여, 해당 읽기사건이 발생한 스레드에서의 최초경합 탐지를 중지하는 대신, 해당 스레드에서 이후에 발생하는 읽기사건만을 반환한다. 이에 따라, 읽기 후보사건 발생 이후에 동일한 스레드에서 쓰기사건이 발생하였을 때, 해당 쓰기사건은 제2 쓰기 경합탐지부(222)에 의해 최초경합 탐지가 수행될 수 있다.

[0050] 제2 쓰기 경합 탐지부(222)는 제2 쓰기 경합판단부(2221), 제2 쓰기 최초경합사건역사 갱신부(2222), 및 쓰기 후보사건 백업부(2223)를 포함한다. 제2 쓰기 경합판단부(2221)는 현재의 쓰기사건과 최초경합사건 역사(232) 내의 접근사건들 간의 선행 관계에 기초하여 경합 여부를 판단할 수 있다. 제2 쓰기 최초경합사건역사 갱신부(2222)는 현재의 쓰기사건이 경합을 구성하지 않으면 반환하고, 경합을 구성하면 최초경합사건 역사(232) 내의 쓰기사건들과의 좌우 관계에 기초하여 쓰기 최초경합사건 역사를 갱신할 수 있다. 쓰기 후보사건 백업부(2223)는 현재의 쓰기사건을 최초경합 후보사건으로 백업할 수 있다. 최초경합 탐지부(2)는 최초경합 후보사건으로 백업된 쓰기 및 읽기 접근사건들을 최종적으로 최초경합 사건으로 보고할 수 있다.

[0051] 도 6은 본 발명의 일 실시예에 따른 최초경합 탐지 방법의 흐름도이다. 도 2 내지 도 6을 참조하면, 본 발명의 일 실시예에 따른 최초경합 탐지 방법은 첫 번째 수행단계(단계 61)(이하, '1 단계' 로 칭할 수 있음)에서 임의의 접근사건이 접근 역사(231) 내의 접근사건들과 병행한지 판단하여 경합 존재 여부를 판단하고, 접근 역사(231)를 사용하여 최초경합사건 역사(232) 내에 접근사건들을 수집할 수 있다. 두 번째 수행단계(단계 62 내지 64)(이하, '2 단계' 로 칭할 수 있음)에서 최초경합사건 역사(232)를 완성하여, 현재의 접근사건이 최초경합 사건(first-race access)인지 결정하고, 이를 최초경합을 보고하는데 사용할 수 있다. 첫 번째 및 두 번째 수행단계는 각각 읽기사건 프로시저(procedure)와, 쓰기사건 프로시저로 구성될 수 있다.

[0052] 도 6을 참조하면, 본 발명의 일 실시예에 따른 최초경합 탐지 방법은 단계 61에서, 프로그램 수행중(on-the-fly) 임의의 공유 변수에 대한 접근사건들 간의 선행 관계(happened-before relation)와, 좌우 관계(left-of relation)를 조사함으로써, 접근유형에 따라 공유 변수에 대한 공유 데이터 구조, 즉 접근 역사(access history) 및 최초경합사건 역사(first-race access history) 내에 접근사건들을 수집할 수 있다.

[0053] 본 발명의 일 실시예에 따른 최초경합 탐지 방법은 공유 데이터 구조를 유지하여 수행중(on-the-fly) 각 공유 변수에 대한 임의의 접근사건들을 모니터링함으로써, 최초경합 후보사건들을 실시간으로 탐지할 수 있다. 다시 말해, 임의의 접근사건이 공유 변수에 접근할 때마다, 실시간 모니터링 기술은 공유 변수에 대한 모든 두 개의 접근사건들 사이의 선행 관계와 좌우 관계를 조사함으로써, 현재 접근 사건과 이전 접근 사건들 사이에 경합이 존재하는지 여부를 실시간으로 확인할 수 있다.

[0054] 본 발명의 일 실시예에 따른 최초경합 탐지 방법은 비효율적 수행시간과 기억공간을 방지하고 최초경합을 효율적으로 탐지하기 위해, 접근 역사의 크기를 항상 상수개로 유지할 수 있다. 예를 들어, 공유 변수 'X' 에 대한

접근 역사 ' AH_x ' 는 가장 최근의 접근사건의 두 개의 부분집합인 읽기 접근역사 ' $AH_x[R]$ ' 와, 쓰기 접근역사 ' $AH_x[W]$ ' 로 구성될 수 있다. 그리고, 읽기 접근역사 ' $AH_x[R]$ ' 은 가장 최근에 발생한 읽기사건들의 좌우 관계를 고려하여 마지막 외부사건에 해당하는 두 개의 읽기사건 $AH_x[R_L]$, $AH_x[R_R]$ 을 저장할 수 있다. 쓰기 접근역사 ' $AH_x[W]$ ' 는 마지막 외부사건에 해당하는 하나의 쓰기사건을 저장할 수 있다. 이러한 경우, 접근 역사의 크기는 세 개로 유지될 수 있다.

[0055] 본 발명의 일 실시예에 따른 최초경합 탐지 방법은 최초경합에 포함될 가능성이 있는 사건들만을 수집하기 위한 최초경합사건 역사 ' FH_x ' 를 사용할 수 있다. 최초경합의 효율적 탐지를 위해, 최초경합사건 역사는 상수적 크기로 유지될 수 있다. 공유 변수 ' X ' 에 대한 최초경합사건 역사 ' FH_x ' 는 어떠한 접근사건에 의해서도 영향받지 않은 접근사건의 두 개의 부분집합인 읽기 최초경합사건 역사 ' $FH_x[R]$ ' 와, 쓰기 최초경합사건 역사 ' $FH_x[W]$ ' 로 구성될 수 있다. 읽기 최초경합사건 역사 ' $FH_x[R]$ ' 는 외부사건으로 수행되고, 경합에 해당하는 두 개의 읽기사건 $FH_x[R_L]$ 및 $FH_x[R_R]$ 을 저장할 수 있다. 쓰기 최초경합사건 역사 ' $FH_x[W]$ ' 는 외부사건으로 수행되고, 경합에 해당하는 두 개의 쓰기사건 $FH_x[W_L]$ 및 $FH_x[W_R]$ 을 저장할 수 있다.

[0056] 단계 62 내지 64에서는, 단계 61에서 수집된 최초경합사건 역사 내의 접근사건들을 활용하여 얻힌 경합에 포함된 모든 최초경합들을 탐지할 수 있다. 단계 62에서는, 현재의 접근사건과 최초경합사건 역사 내에 수집된 접근사건들 간의 선행 관계 및 좌우 관계에 기초하여, 최초경합사건 역사를 갱신할 수 있다. 단계 63에서는, 만약 현재의 읽기사건이 최초경합사건 역사 내에 수집된 쓰기사건과 경합이면, 읽기사건이 발생한 스레드에서 읽기후보사건이 발생하였음을 나타내는 플래그(flag)를 설정된 값으로 갱신할 수 있다. 단계 64에서는, 스레드에서의 최초경합 탐지를 중지하지 않고, 스레드에서 읽기사건 이후에 발생하는 읽기사건만을 반환하여, 스레드에서 읽기사건 이후에 발생하는 쓰기사건에 대하여 최초경합을 탐지할 수 있다.

[0057] 도 7은 도 6에 도시된 단계 61의 읽기사건에 대한 예시적인 흐름도이다. 단계 71에서는 현재의 읽기사건과 접근역사 내의 쓰기사건 간의 선행 관계에 기초하여 경합 여부를 판단할 수 있다. 단계 72에서는 현재의 읽기사건과 접근역사 내의 읽기사건 간의 좌우 관계에 기초하여 읽기 접근역사를 갱신할 수 있다. 단계 73에서는 현재의 읽기사건이 접근역사 내의 쓰기사건과 경합이 아니면 반환하고, 경합이면 최초경합사건 역사 내의 읽기사건들과의 좌우 관계에 기초하여 읽기 최초경합사건 역사를 갱신할 수 있다.

[0058] 도 8은 도 6에 도시된 단계 61의 쓰기사건에 대한 예시적인 흐름도이다. 단계 81에서는 현재의 쓰기사건과 접근역사 내의 접근사건들 간의 선행 관계에 기초하여 경합 여부를 판단하고, 쓰기 접근역사를 갱신할 수 있다. 단계 82에서는 현재의 쓰기사건이 경합에 포함되지 않으면 반환하고, 경합에 포함되면 최초경합사건 역사 내의 쓰기사건들과의 좌우 관계에 기초하여 쓰기 최초경합사건 역사를 갱신할 수 있다. 단계 83에서는 현재의 쓰기사건이 발생한 스레드에서의 경합탐지를 중지할 수 있다.

[0059] 도 9는 도 6에 도시된 단계 62 내지 64의 읽기사건에 대한 예시적인 흐름도이다. 단계 91에서는 현재의 스레드에서 읽기 후보사건이 발생하였는지를 판단할 수 있다. 단계 92에서는 현재의 스레드에서 읽기후보사건이 발생하였으면, 현재의 읽기사건이 최초경합사건 역사 내에 수집된 쓰기사건과 경합을 구성하는지를 판단할 수 있다. 단계 93에서는 현재의 읽기사건이 경합에 포함되지 않으면 반환하고, 경합에 포함되면 현재의 읽기사건과 최초경합사건 역사 내에 수집된 읽기사건들 간의 좌우 관계에 기초하여 읽기 최초경합사건 역사를 갱신할 수 있다. 단계 94에서는 현재의 읽기사건을 최초경합 후보사건으로 백업할 수 있다. 단계 95에서는 현재의 읽기사건이 발생한 스레드에서 읽기후보사건이 발생하였음을 나타내는 플래그(flag)를 설정된 값으로 갱신할 수 있다.

[0060] 도 10은 도 6에 도시된 단계 62 내지 64의 쓰기사건에 대한 예시적인 흐름도이다. 단계 101에서는 현재의 쓰기사건과 최초경합사건 역사 내의 접근사건들 간의 선행 관계에 기초하여 경합 여부를 판단할 수 있다. 단계 102에서는 현재의 쓰기사건이 경합에 포함되지 않으면 반환하고, 경합에 포함되면 최초경합사건 역사 내의 접근사건들과의 좌우 관계에 기초하여 쓰기 최초경합사건 역사를 갱신할 수 있다. 단계 103에서는 현재의 쓰기사건을 최초경합 후보사건으로 백업할 수 있다. 단계 104에서는 현재의 쓰기사건이 발생한 스레드에서의 경합탐지를 중지할 수 있다.

[0061] 이상에서 설명한 도 6 내지 도 10에 도시된 실시예에 따른 최초경합 탐지 방법은 컴퓨터에서 실행될 수 있는 프로그램으로 작성 가능하고, 컴퓨터로 읽을 수 있는 기록매체를 이용하여 상기 프로그램을 동작시키는 범용 디지털 컴퓨터에서 구현될 수 있다. 상기 컴퓨터로 읽을 수 있는 기록매체는 마그네틱 저장매체(예를 들면, 롬, 플

로피 디스크, 하드 디스크 등), 광학적 판독 매체(예를 들면, 시디롬, 디브이디 등)와 같은 저장매체를 포함할 수 있다.

[0062] 아래의 표 1은 예시적인 1 단계 읽기사건과의 경합탐지 알고리즘으로, 도면 7의 단계들에 대응한다.

표 1

```

0 Checkread_1st( $X$ ,  $current$ )
1 if  $\neg$  Ordered( $AH_X[W]$ ,  $current$ ) then
2    $racing\_current := true$ ;
3 if  $\neg$  Leftof( $AH_X[R_L]$ ,  $current$ ) then
4    $AH_X[R_L] := current$ ;
5 if  $\neg$  Leftof( $current$ ,  $AH_X[R_R]$ ) then
6    $AH_X[R_R] := current$ ;
7 if  $\neg$   $racing\_current$  then
8   return;
9 if  $FH_X[R_L] = \emptyset$  or
10  Leftof( $current$ ,  $FH_X[R_L]$ ) then
11    $FH_X[R_L] := current$ ;
12 if  $FH_X[R_R] = \emptyset$  or
13  Leftof( $FH_X[R_R]$ ,  $current$ ) then
14    $FH_X[R_R] := current$ ;
15 end Checkread_1st

```

[0064] 표 1에서, ' $current$ '는 공유 변수 ' X '에 대한 현재의 접근사건을 나타낸다. 0번 줄에서, 공유 변수 ' X '와 공유 변수에 대한 현재의 접근사건 ' $current$ '이 입력된다. 1-2번 줄에서는, 이전에 쓰기 접근역사 $AH_X[W]$ 내에 저장된 쓰기사건과 선행 관계(happened-before relation)가 비교되어, 현재의 읽기사건 ' $current$ '가 경합에 포함되는지 검사된다. 3-6번 줄에서는, 현재의 읽기사건과 읽기 접근역사 ' $AH_X[R]$ ' 내에 저장된 모든 읽기 사건들 $AH_X[R_L]$, $AH_X[R_R]$ 과의 좌우 관계(Leftof relation)가 검사된다. 이에 따라, 읽기 접근역사 ' $AH_X[R]$ '에는 외부사건에 해당하는 읽기 접근사건이 저장되어 갱신된다.

[0065] 만약, 1번 줄에서 현재의 접근사건이 경합이 아니라면, 7-8번 줄에서 현재의 읽기사건은 반환되고 프로시저가 종료된다. 만약, 1번 줄에서 현재의 접근사건이 경합이면, 9-14번 줄에서 현재의 접근사건과 최초경합사건 역사 ' $FH_X[R]$ ' 내에 저장된 이전 접근사건들 $FH_X[R_L]$, $FH_X[R_R]$ 사이의 좌우 관계가 검사된다. 만약, $FH_X[R_L]$, 또는 $FH_X[R_R]$ 가 비어 있거나, 좌우 관계 검사 결과 현재의 접근사건이 이전 접근사건을 교체하는 새로운 외부사건으로 결정되면, 현재의 접근사건이 $FH_X[R_L]$, $FH_X[R_R]$ 으로 저장된다. 이에 따라, 읽기 최초경합사건 역사 ' $FH_X[R]$ '는 영향받지 않은 경합사건으로 갱신될 수 있다.

[0066] 아래의 표 2는 예시적인 1 단계 쓰기사건과의 경합탐지 알고리즘으로, 도면 8의 단계들에 대응한다.

표 2

```

0 Checkwrite_1st( $X$ ,  $current$ )
1 for all  $c$  in  $AH_X$  do
2   if  $\neg Ordered(c, current)$  then
3      $racing\_current := true$ ;
4    $AH_X[W] := current$ ;
5   if  $\neg racing\_current$  then
6     return;
7   if  $FH_X[W_L] = \emptyset$  or
8     Leftof( $current, FH_X[W_L]$ ) then
9      $FH_X[W_L] := current$ ;
10  if  $FH_X[W_R] = \emptyset$  or
11    Leftof( $FH_X[W_R], current$ ) then
12     $FH_X[W_R] := current$ ;
13 halt;
14 end Checkwrite_1st

```

[0067]

[0068] 표 2를 참조하면, 1-3번 줄에서 현재의 쓰기사건 ' $current$ ' 와, 접근 역사 ' AH_X ' 내에 저장된 세 개의 이전 접근사건들의 선행관계가 검사되어, 현재의 접근사건이 경합을 구성하는지가 결정된다. 쓰기 접근역사 ' $AH_X[W]$ ' 에 저장된 쓰기사건은 이후의 읽기사건 또는 쓰기사건과 경합을 구성할 수 있으므로, 4번 줄에서는 현재의 쓰기사건이 쓰기 접근역사 ' $AH_X[W]$ ' 에 저장되어 ' $AH_X[W]$ ' 가 갱신된다. 경합을 구성하지 않는 접근사건은 최초경합에 해당하지 않으므로, 만약 1-3번 줄에서 현재의 접근사건이 경합에 포함되지 않으면, 5-6번 줄에서 현재의 쓰기사건이 반환되고 프로시저가 중단된다.

[0069] 만약 1-3번 줄에서 현재의 접근사건이 경합에 포함되면, 7-12번 줄에서 현재의 접근사건과 쓰기 최초경합사건 역사 ' $FH_X[W]$ ' 내에 저장된 이전 접근사건들 ' $FH_X[W_L]$, $FH_X[W_R]$ ' 간 좌우관계가 검사된다. 만약, ' $FH_X[W_L]$ ' , 또는 ' $FH_X[W_R]$ ' 가 비어 있거나, 좌우 관계 검사 결과 현재의 접근사건이 이전 접근사건을 교체하는 새로운 외부사건으로 결정되면, 현재의 접근사건이 ' $FH_X[W_L]$, $FH_X[W_R]$ ' 으로 저장된다. 이에 따라, 쓰기 최초경합사건 역사 ' $FH_X[R]$ ' 는 경합을 구성하는 영향받지 않은 외부사건으로 갱신될 수 있다.

[0070] 만약 1-3번 줄에서 현재의 쓰기사건이 경합에 포함되면, 현재의 스레드에서 이후에 발생하는 쓰기사건은 더 이상 최초경합 사건일 수 없으므로 불필요한 감시시간을 제거하기 위해 13번 줄에서 현재의 스레드에서의 1 단계 경합탐지가 중단된다. 만약, 도 1의 예에 전술한 1 단계 경합탐지 알고리즘이 적용되면, 접근 역사 ' AH_X ' 에는 $r9$, $r5$, $w10$ 이 저장되고, 최초경합사건 역사 ' FH_X ' 에는 $r8$, $w10$ 이 저장된다.

[0071] 아래의 표 3은 예시적인 2 단계 읽기사건과의 경합탐지 알고리즘으로, 도면 9의 단계들에 대응한다.

표 3

```

0 Checkread_2nd( $X$ ,  $current$ )
1 if  $read\_first\_candidate$  then
2   return;
3 for all  $c$  in  $FH_X[W]$  do
4   if  $\neg Ordered(c, current)$  then
5      $racing\_current := true$ ;
6   if  $\neg racing\_current$  then
7     return;
8 if  $FH_X[R_L] = \emptyset$  or
9   Leftof( $current, FH_X[R_L]$ ) then
10    $FH_X[R_L] := current$ ;
11 if  $FH_X[R_R] = \emptyset$  or
12   Leftof( $FH_X[R_R], current$ ) then
13    $FH_X[R_R] := current$ ;
14 add  $current$  to  $FS_X[R]$ ;
15  $read\_first\_candidate := true$ ;
16 end Checkread_2nd

```

[0072]

[0073] 표 3을 참조하면, 현재의 읽기사건 이전에 발생한 최초경합 읽기 후보사건이 존재하면 'read_first_candidate' 플래그(flag)가 'true' 로 갱신되고, 그 이후에 발생한 읽기사건은 유효하지 않다. 따라서, 1-2번 줄에서 'read_first_candidate' 플래그가 'true' 인지 판단하여, 현재 스레드에서 최초경합 읽기 후보사건이 존재하면 현재의 읽기사건을 반환하고, 현재의 읽기사건에 대한 2 단계 경합탐지 프로시저를 중단한다. 만약, 'read_first_candidate' 플래그가 'true' 가 아니면, 3-5번 줄에서 현재의 읽기사건과 1 단계 알고리즘에서 수집된 쓰기 최초경합사건 역사 ' $FH_X[W]$ ' 에 수집된 모든 쓰기사건들 간의 선행 관계를 검사하여 현재의 읽기사건이 경합에 포함되는지 판단한다.

[0074] 만약, 3-5번 줄에서 현재의 읽기사건이 경합이 아니면, 현재의 읽기사건은 최초경합 후보사건이 될 수 없으므로, 6-7번 줄에서 현재의 읽기사건이 반환되고, 현재의 2 단계 읽기사건과의 경합탐지 프로시저가 중단된다. 현재의 읽기사건이 경합에 포함되면, 8-13번 줄에서 현재의 읽기사건과 읽기 최초경합사건 역사 ' $FH_X[R]$ ' 내에 저장된 읽기사건들 사이의 좌우 관계가 검사된다. 만약, $FH_X[R_L]$, 또는 $FH_X[R_R]$ 가 비어 있거나, 좌우 관계 검사 결과 현재의 읽기사건이 이전 역사를 교체할 새로운 외부사건으로 결정되면, 현재의 읽기사건이 $FH_X[R_L]$, $FH_X[R_R]$ 으로 저장된다. 이에 따라, 읽기 최초경합사건 역사 ' $FH_X[R]$ ' 는 경합을 구성하는 영향받지 않은 외부사건으로 갱신될 수 있다.

[0075] 그리고, 현재의 읽기사건 이전에 발생한 최초경합 읽기 후보사건이 존재하지 않으면서 현재의 읽기사건이 경합에 포함되면, 최초경합사건이지만 외부사건이 아니어서 최초경합으로 보고되지 않는 것을 막기 위해 14번 줄에서 현재의 읽기사건은 최초경합 후보사건으로 보고될 ' $FS_X[R]$ ' 에 추가되어 백업된다. 그리고, 15번 줄에서, 현재의 읽기사건이 발생한 스레드에서 최초경합 읽기 후보사건이 발생하였음을 나타내는 'read_first_candidate' 플래그(flag)가 'true' 로 갱신된다. 플래그가 갱신된 이후에 동일한 스레드에서 발생하는 읽기사건은 표 3의 1-2번 줄에서 반환되지만, 동일한 스레드에서 발생한 쓰기사건에 대하여는 2 단계 쓰기사건과의 경합탐지가 수행될 수 있다. 이에 따라, 최초업힘에 포함되는 모든 최초경합들에 대한 탐지가 가능해질 수 있다.

[0076] 아래의 표 4는 예시적인 2 단계 쓰기사건과의 경합탐지 알고리즘으로, 도면 10의 단계들에 대응한다.

표 4

```

0 Checkwrite_2nd( $X$ ,  $current$ )
1 for all  $c$  in  $FH_X$  do
2   if  $\neg \text{Ordered}(c, current)$  then
3      $racing\_current := true$ ;
4   if  $\neg racing\_current$  then
5     return;
6   if  $FH_X[W_L] = \emptyset$  or
7     Leftof( $current, FH_X[W_L]$ ) then
8      $FH_X[W_L] := current$ ;
9   if  $FH_X[W_R] = \emptyset$  or
10    Leftof( $FH_X[W_R], current$ ) then
11     $FH_X[W_R] := current$ ;
12 add  $current$  to  $FS_X[W]$ ;
13 halt;
14 end Checkwrite_2nd

```

[0077]

[0078] 표 4를 참조하면, 1-3번 줄에서 현재의 쓰기사건과 최초경합사건 역사 FH_X 내에 저장된 네 개의 접근사건의 선행관계가 검사되어 현재의 쓰기사건이 경합에 포함되는지 판단된다. 현재의 쓰기사건이 경합에 포함되지 않으면, 4-5번 줄에서 현재의 쓰기사건이 반환된다. 6-11번 줄에서는, 현재의 쓰기사건과 쓰기 최초경합사건 역사 ' $FH_X[W]$ ' 내에 저장된 쓰기사건들 사이의 좌우 관계가 검사된다. 만약, $FH_X[W_L]$, 또는 $FH_X[W_R]$ 가 비어 있거나, 좌우 관계 검사 결과 현재의 쓰기사건이 이전 역사를 교체할 새로운 외부사건으로 결정되면, 현재의 쓰기사건이 $FH_X[W_L]$, $FH_X[W_R]$ 으로 저장된다. 이에 따라, 쓰기 최초경합사건 역사 ' $FH_X[W]$ ' 는 경합을 구성하는 영향받지 않은 외부사건으로 갱신될 수 있다.

[0079] 그리고, 현재의 쓰기사건이 경합에 포함되면, 12번 줄에서 현재의 쓰기사건은 최초경합 후보사건으로 보고될 ' $FS_X[W]$ ' 에 추가되어 백업된다. 마지막으로 13번 줄에서 감시시간의 효율성을 위해서 최초경합 탐지를 중지하고, 불필요한 감시시간을 제거하기 위해 현재의 스레드는 중단된다. 만약, 도 1의 예에 전술한 1 단계 및 2 단계 경합탐지 알고리즘이 적용되면, 최초경합사건 역사 ' FH_X ' 에는 사건 $r8$, $r0$, $w10$ 이 저장되고, 최초경합 후보사건 집합 ' FS_X ' 에는 사건 $r0$, $w10$ 이 저장된다.

[0080] 도 11은 또 하나의 예시적인 POEG 를 나타낸 도면이다. 표 1과 표 2에 나타낸 1 단계 경합탐지 알고리즘을 도 11에 도시된 POEG 에 적용한 경우의 접근 역사 및 최초경합사건 역사를 나타내면 아래의 표 5와 같다.

표 5

[0081]

	접근 역사 (AH)			최초경합사건 역사 (FH)			
	R_L	R_R	W	R_L	R_R	W_L	W_R
$r0$	$r0$	$r0$					
$r1$	$r1$	$r1$					
$r2$	$r1$	$r2$					
$r3$	$r3$	$r2$					
$r4$	$r3$	$r4$					
$r5$	$r5$	$r5$					
$r6$	$r6$	$r6$					
$w7$	$r6$	$r6$	$w7$				
$r8$	$r8$	$r6$	$w7$	$r8$	$r8$		

r9	r9	r6	w7	r8	r8		
w10	r9	r6	w10	r8	r8	w10	w10
w11	r9	r6	w11	r8	r8	w11	w10

[0082] 이 때, 최초경합 후보사건으로는 w10, w11 이 백업된다.

[0083] 다음으로, 표 3과 표 4에 나타난 2 단계 경합탐지 알고리즘을 도 11에 도시된 POEG 의 예에 적용한 경우의 최초 경합사건 역사를 나타내면 아래의 표 6과 같다.

표 6

[0084]

	최초경합사건 역사 (FH)			
	R _L	R _R	W _L	W _R
First	r8	r8	w11	w10
r0	r8	r0	w11	w10
r1	r8	r0	w11	w10
r2	r8	r0	w11	w10
r3	r8	r0	w11	w10
r4	r8	r0	w11	w10
r5	r8	r0	w11	w10
r6	r8	r0	w11	w10
w7	r8	r0	w11	w7
r8	r8	r0	w11	w7
r9	r8	r0	w11	w7
w10	r8	r0	w11	w7
w11	r8	r0	w11	w7

[0085] 이 때, 최초경합 후보사건으로는 r0, r8, w10, w11, w7 이 백업된다.

[0086] 기존의 최초경합 탐지 방법의 경우, 읽기사건 r0 가 읽기 후보사건으로 결정되면, r0 가 발생한 스레드의 이후 경합탐지가 중지되므로, r0 이후에 동일한 스레드에서 발생한 쓰기사건 w7 이 최초경합 후보사건으로 결정되지 않을 수 있다. 이와 달리, 본 발명의 일 실시예에 따른 최초경합 탐지 방법에 의하면, 읽기사건 r0 가 읽기 후보사건으로 결정되더라도, 단지 r0 발생 이후에 동일한 스레드에서 발생하는 다른 읽기사건을 반환할 뿐, r0 가 발생한 스레드의 이후 경합탐지가 중지되지 않고 해당 스레드에서의 경합탐지가 수행되며, 그 결과 쓰기사건 w7 이 최초경합 후보사건으로 결정될 수 있다.

[0087] 본 발명의 실시예의 최초경합 탐지 성능을 검증하기 위해, 모의실험을 수행하였다. 이 때, C 언어로 쓰여진 실시간 라이브러리를 구현하고, 프로그램을 내포병렬성을 지원하는 OpenMP 3.0 프로그램을 위한 gcc 4.4.4로 컴파일하고, 리눅스 운영시스템의 Kernel 2.6 체제하에서 Intel Quad-core 2Xeon CPU, 8GB RAM 을 갖는 시스템에서 실행하였다. 최초경합 탐지의 정확성 및 효율성 실험을 위한 합성 프로그램은 내포깊이, 스레드 수, 접근사건의 수, 그리고 접근사건의 배치 등을 달리하여 작성되었다.

[0088] 현재의 접근사건과 접근 역사에 저장된 이전 접근사건들의 논리적 병행 관계 검사와, 스레드의 병행성 정보 생성은, 내포병렬성을 갖는 프로그램에 효율적인 NR-레이블링(Nest Region Labeling) 기법을 사용하여 수행되었다. 그리고, 최초 얽힘 탐지의 정확성을 실험하기 위해서 내포가 없는 모델과 내포수준이 2 인 프로그램 모델로 합성프로그램을 작성하였다. 표 7은 12번의 수행 결과 탐지된 경합을 나타낸 것으로, 본 발명의 실시예에 따른 최초경합 탐지 장치는 모든 최초경합을 정확하게 탐지하였음을 확인할 수 있다.

표 7

[0089]

번호	Synthetic			탐지된 경합	번호	Synthetic			탐지된 경합
1	r1	r2		r1-w4	7	r1	r2	r3	w4-r2
	r3	w4				w4	r5	r6	
2	r1	r2		r1-w5 r1-w4	8	r1	r2	r3	r1-w6, r2-w4 r2-w6, r3-w4
	r3	w4	w5			w4	r5	w6	

3	r1	w2		r1-w2	9	r1	r2		r1-w4, r1-w5 w4-w5
	r3	r4	w5			r3	w4	w5	
4	w1	r2		w1-r2	10	r1	r2		r1-w4, r1-w5 r2-w3
	r3	r4	w5			w3	w4	w5	
5	r1	r2		r1-w5 w5-r6	11	r1	r2		r1-w5 r4-w5
	r3	r4				r3	r4	w5	
	r7	w5	r6			r6	r7		
6	r1	r2		r1-w4	12	r1	r2		r1-w7
	r3	w4				r3	r4	r5	
	r7	w5	w6			r6	w7		

표 8

유형	합성 프로그램			업힌 경합의 수	탐지된 업힌 경합의 수					
					종래 기술			본 발명		
					최소	최대	평균	최소	최대	평균
1	r	r	r	6	6	6	6	6	6	6
				12	9	12	11.1	12	12	12
				20	20	20	20	20	20	20
				30	25	30	28	30	30	30
	w	w	w	42	36	42	41.4	42	42	42
				56	56	56	56	56	56	56
				72	40	40	40	72	72	72
				90	36	45	44.1	90	90	90
2	r	r	r	3	3	3	3	3	3	3
				10	6	10	9.6	12	12	12
				21	20	21	20.4	21	21	21
				36	28	36	32.8	36	36	36
	w	w	w	55	46	55	54.1	55	55	55
				78	78	78	78	78	78	78
				105	105	105	105	105	105	105
				136	30	76	66.7	136	136	136

[0090]

[0091] 표 8은 종래 기술 및 본 발명의 실시예에 따른 최초경합 탐지 수행 결과를 나타낸 결과이다. 종래 기술로는 기존의 최초경합 탐지 방법 중 접근 역사를 상수적 크기로 유지하는 방법이 사용되었다. 이 때, 프로그램은 두 가지 유형의 병렬 스레드에서 수행되었다.

[0092] 유형 1은 비내포된 프로그램으로, 동시에 세 개의 스레드를 수행하고, 업힌 경합을 발생시키기 위해서 각 스레드마다 읽기사건 발생 후에 쓰기사건을 발생시켰으며, 같은 패턴의 사건을 발생시키는 스레드의 수를 증가시켜 업힌의 수가 6, 12, 20, ..., 90개를 포함하도록 작성하였다.

[0093] 유형 2는 업힌 경합의 수를 증가시키기 위해, 마지막 스레드의 수를 증가시켰으며, 내포가 없는 스레드와 2개의 스레드를 내포하는 스레드를 병행하게 수행시켰다. 이 때, 두 개의 스레드를 동시에 수행하였으며, 두 번째 스레드는 두 개의 Child 스레드를 갖도록 하였다. 또한, 유형 1과 마찬가지로 업힌 경합을 발생시키기 위해서 각 스레드마다 읽기사건 발생 후에 쓰기사건을 발생시켰다. 그리고, 내포된 스레드의 수를 증가시켜 업힌의 수가 3, 10, 21, 36, ..., 136이 되도록 하였다. 각 프로그램을 10번씩 수행하고, 최대값과 최소값, 및 전체 검출된 경합의 수의 평균값을 구하였다.

[0094] 표 8에 나타낸 바와 같이, 본 발명의 실시예에 따른 최초경합 탐지 방법은 대상 프로그램에 포함된 업힌의 수를 모두 탐지한 반면, 종래 기법은 전체 경합들 중에서 일부만을 탐지하였다. 종래 기법은 최악의 경우, 136개의 업힌이 포함된 경우 30개의 업힌만을 탐지하였음을 알 수 있다. 종래 기법의 경우, 업힌의 개수가 증가할수록,

탐지하지 못하는 경합의 수도 증가하였다. 따라서, 종래의 최초경합 탐지 기법은 최초경합의 얽힘의 검증을 보장할 수 없으나, 본 발명의 실시예에 따른 최초경합 탐지 방법은 내포병렬성을 갖는 공유메모리 프로그램에서 최초 얽힘을 검증할 수 있음이 실험적으로 증명되었다.

공유 변수의 수를 V 로, 네포 깊이(nesting depth)를 N 으로, 병렬프로그램의 최대 병렬성(maximum parallelism)을 T 로 나타내면, 본 발명의 실시예에서 최악의 경우의 복잡도는, 일곱 개의 상수적 크기의 역사의 등록을 위한 공간 $O(V)$ 과, 접근 사건들의 선행 관계 및 좌우 관계를 비교하기 위한 시간 $O(\log_2 N)$ 만을 포함하므로, 최대 병렬성에 의존하지 않는다. 따라서, 본 발명의 실시예는 프로그램 수행중 각 공유 변수에 대한 역사를 상수적 크기로 유지하여 각 접근사건의 수행시에 상수적 복잡도의 사건비교 횟수와 기억공간만을 요구하므로 효율적이면서도 정확하게 최적경합을 탐지할 수 있다.

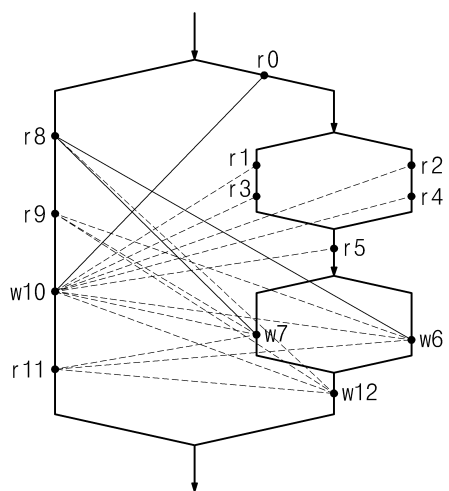
이상의 실시예들은 본 발명의 이해를 돕기 위하여 제시된 것으로, 본 발명의 범위를 제한하지 않으며, 이로부터 다양한 변형 가능한 실시예들도 본 발명의 범위에 속할 수 있음을 이해하여야 한다. 예를 들어, 본 발명의 실시예에 도시된 각 구성 요소는 분산되어 실시될 수도 있으며, 반대로 여러 개로 분산된 구성 요소들은 결합되어 실시될 수 있다. 따라서, 본 발명의 기술적 보호범위는 특허청구범위의 기술적 사상에 의해 정해져야 할 것이며, 본 발명의 기술적 보호범위는 특허청구범위의 문언적 기재 그 자체로 한정되는 것이 아니라 실질적으로는 기술적 가치가 균등한 범주의 발명에 대하여까지 미치는 것임을 이해하여야 한다.

부호의 설명

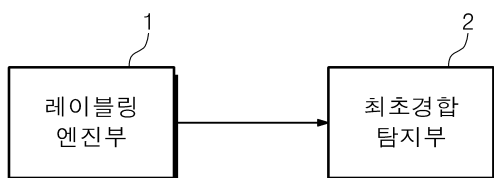
1: 레이블링 엔진부	2: 최초경합 탐지부
21: 제1 경합탐지부	211: 제1 읽기 경합탐지부
2111: 제1 읽기 경합 판단부	2112: 제1 읽기 접근역사 갱신부
2113: 제1 읽기 최초경합사건역사 갱신부	212: 제1 쓰기 경합탐지부
2121: 제1 쓰기 경합 판단부	2122: 제1 쓰기 접근역사 갱신부
2123: 제1 쓰기 최초경합사건역사 갱신부	22: 제2 경합탐지부
221: 제2 읽기 경합탐지부	2211: 읽기 후보 발생 판단부
2212: 제2 읽기 경합 판단부	
2213: 제2 읽기 최초경합사건역사 갱신부	2214: 읽기 후보사건 백업부
2215: 플래그 갱신부	222: 제2 쓰기 경합탐지부
2221: 제2 쓰기 경합 판단부	
2222: 제2 쓰기 최초경합사건역사 갱신부	2223: 쓰기 후보사건 백업부
23: 저장부	231: 접근 역사
232: 최초경합사건 역사	

도면

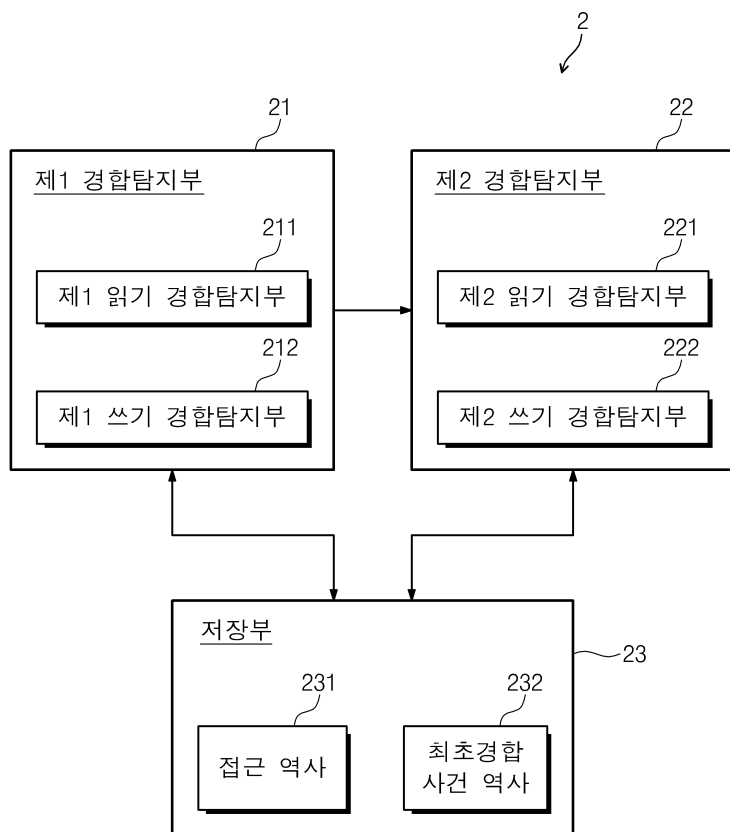
도면1



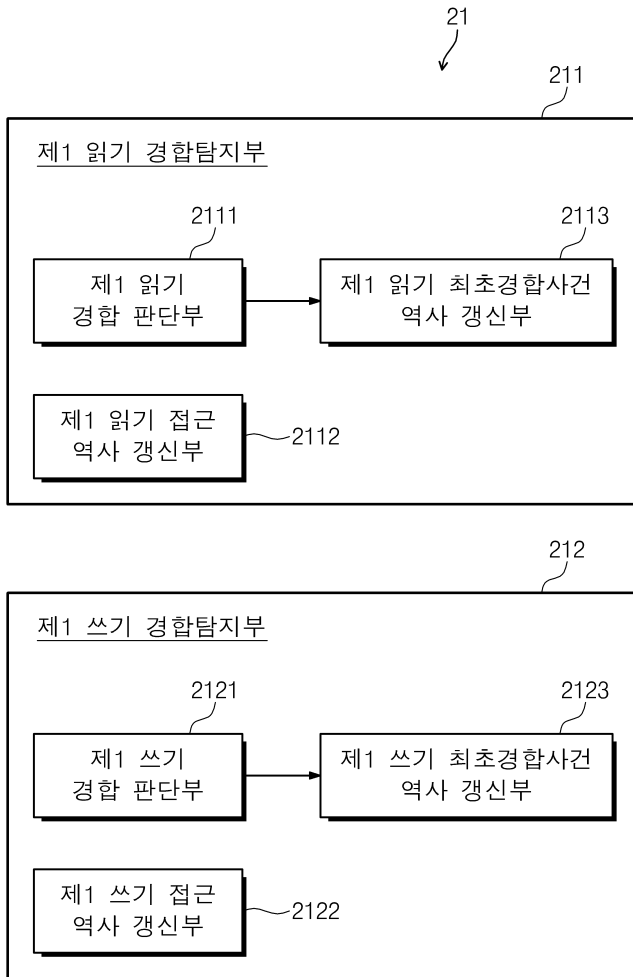
도면2



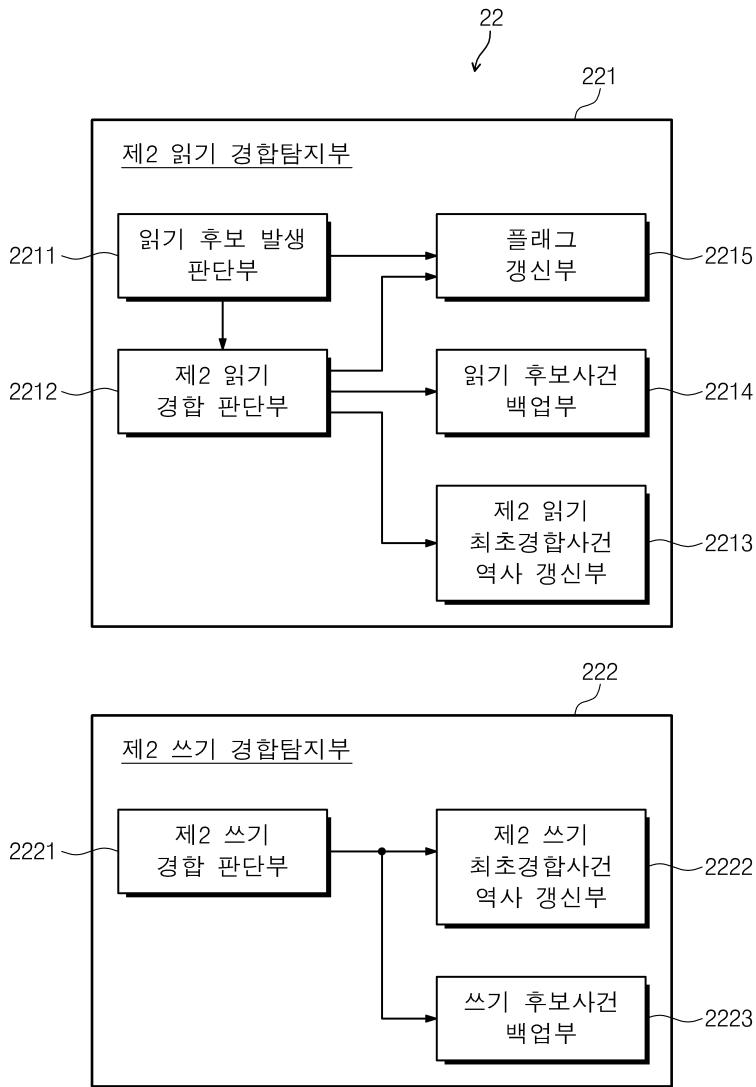
도면3



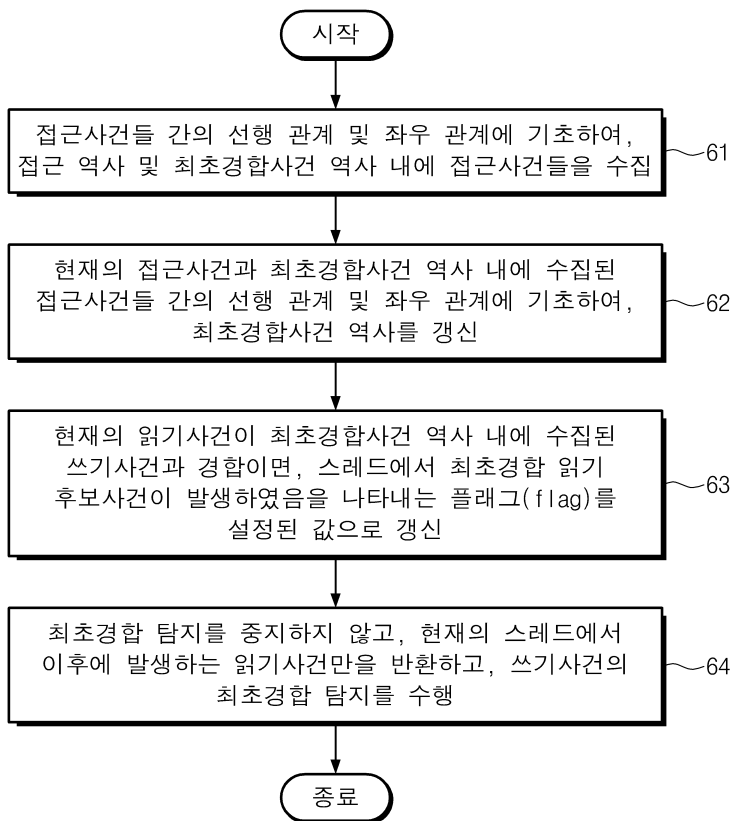
도면4



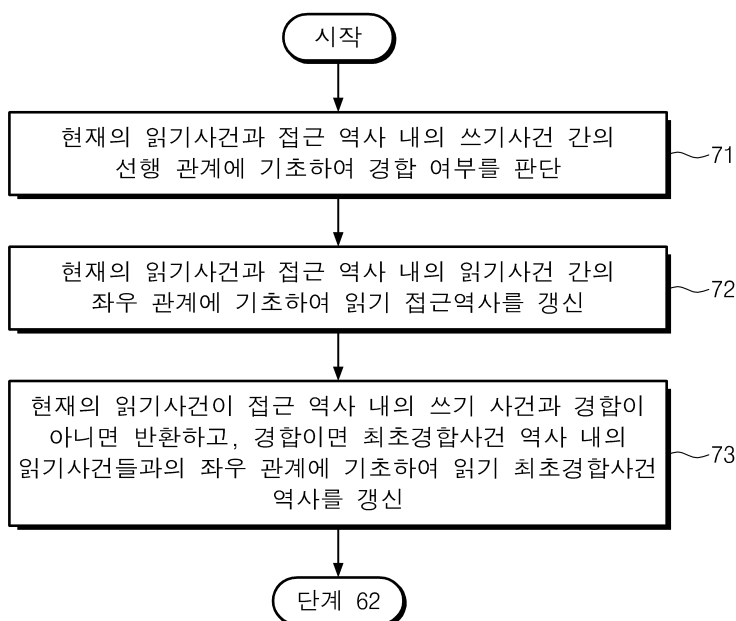
도면5



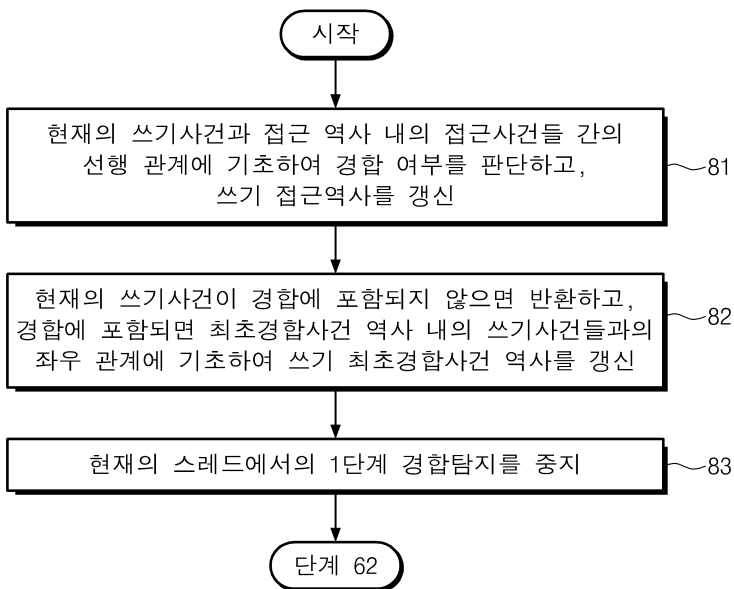
도면6



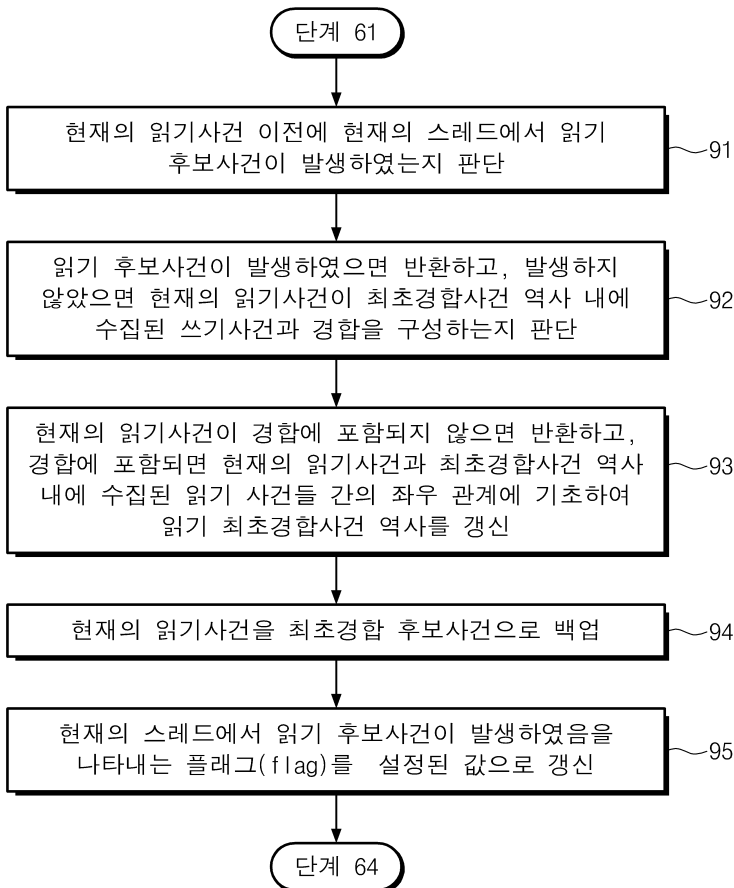
도면7



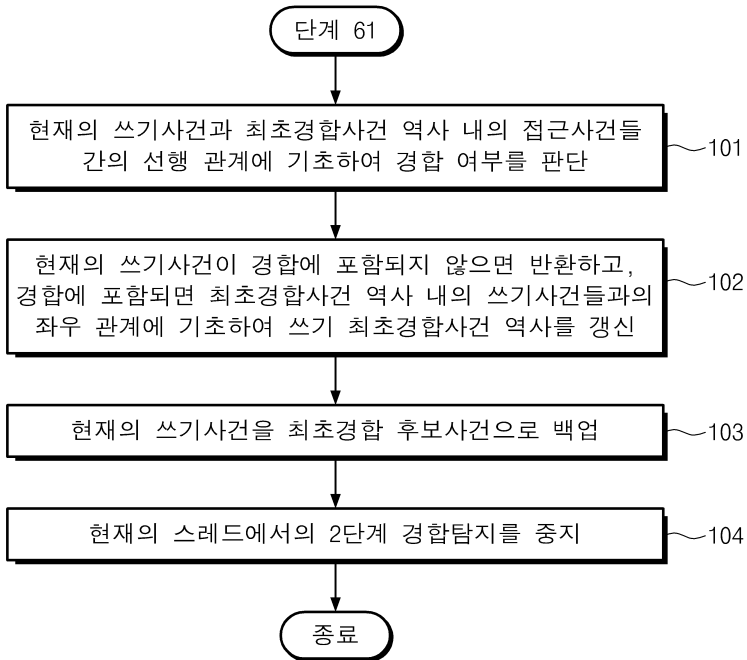
도면8



도면9



도면10



도면11

