(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2010-0032778
(43) 공개일자 2010년03월26일

(51) Int. Cl.
G06F 9/46 (2006.01) *G06F 9/44* (2006.01)
G06F 13/00 (2006.01) *G06F 17/00* (2006.01)
(21) 출원번호 10-2008-0091806
(22) 출원일자 2008년09월18일
심사청구일자 2008년09월18일

(71) 출원인
고려대학교 산학협력단
서울 성북구 안암동5가1 고려대학교 내
(72) 발명자
김선욱
경기 남양주시 도농동 부영아파트 2단지 214동 2003호
정동하
서울 성북구 안암동2가 94-1 601호
오재근
전북 전주시 덕진구 덕진동1가 649번지 1통 4반
(74) 대리인
전종학

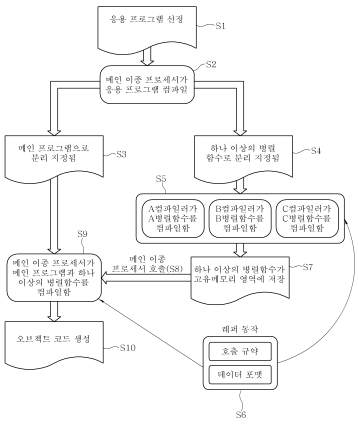
전체 청구항 수 : 총 12 항

(54) 다수의 이종 프로세서를 구비하는 멀티 프로세싱 시스템 및그 구동 방법

(57) 요 약

본 발명은 다수의 이종 프로세서 간에 처리되는 멀티 프로세싱 기술을 개시한다. 즉, 다수의 이종 프로세서를 구비하는 멀티 프로세싱 시스템 및 그 구동 방법은 기선정된 응응 프로그램이 메인 프로그램과 하나 이상의 병렬 함수로 분리 지정된 후, 서브 이종 프로세서에 의해 컴파일된 하나 이상의 병렬 함수가 상기 서브 이종 프로세서에 기구비된 레퍼를 통과함에 따라, 상기 컴파일된 하나 이상의 병렬 함수가 메인 이종 프로세서의 고유 호출 규약과 데이터 포맷에 맞도록 변경 또는 정합(整合)되게 하는 시스템 및 그 구동 방법을 구현함으로써, 다수의 이종 프로세서 간의 호출 규약 및 데이터 포맷에 대한 차이를 레퍼를 통해 극복하여 기존 개발 툴의 큰 수정 없이도 병렬적으로 처리하고자 하는 응용 프로그램을 할당 및 처리할 수 있어 시스템 성능 향상 및 고집적화로 인한 매출 증대를 높일 수 있다.

대 표 도 - 도4



이 발명을 지원한 국가연구개발사업

과제고유번호	10030565-2007-01
부처명	한국 산업 기술 평가원
연구사업명	시스템집적반도체기반기술개발사업
연구과제명	MPSoc 설계 기술 개발
주관기관	(주)자람테크놀러지
연구기간	2007년 09월 01일 ~ 2008년 08월 31일

특허청구의 범위

청구항 1

레퍼가 1:1 매칭하며 각각 삽입된 다수의 이종 프로세서를 구비하며, 상기 다수의 이종 프로세서 중 메인 이종 프로세서와 서브 이종 프로세서를 선택한 멀티 프로세싱 시스템으로,

기선정된 응응 프로그램이 메인 프로그램과 하나 이상의 병렬 함수로 분리 지정된 후, 상기 서브 이종 프로세서에 의해 컴파일된 하나 이상의 병렬 함수가 상기 서브 이종 프로세서에 기구비된 레퍼를 통과함에 따라, 상기 컴파일된 하나 이상의 병렬 함수가 상기 메인 이종 프로세서의 고유 호출 규약과 데이터 포맷에 맞도록 변경 또는 정합(整合)되게 하는 것을 특징으로 하는 다수의 이종 프로세서를 구비하는 멀티 프로세싱 시스템.

청구항 2

레퍼가 1:1 매칭하며 각각 삽입된 다수의 이종 프로세서를 구비하는 멀티 프로세싱 시스템으로,

상기 다수의 이종 프로세서 중 선택된 하나이며, 소정의 기선정된 응용 프로그램을 컴파일하여 메인 프로그램과 하나 이상의 병렬 함수로 분리 지정하는 메인 이종 프로세서; 및

상기 다수의 이종 프로세서 중 선택된 다른 하나이며, 상기 메인 이종 프로세서로부터 할당된 하나 이상의 병렬 함수를 자체 구비된 전용 메모리에 매핑한 후, 상기 매핑된 하나 이상의 병렬 함수를 컴파일하는 서브 이종 프로세서;를 포함하며,

상기 서브 이종 프로세서에 기구비된 레퍼가,

상기 컴파일된 하나 이상의 병렬 함수를 통과시켜 상기 컴파일된 하나 이상의 병렬 함수를 상기 메인 이종 프로세서의 고유 호출 규약과 데이터 포맷과 일치한 데이터로 변경 또는 정합(整合)하는 것을 특징으로 하는 다수의 이종 프로세서를 구비하는 멀티 프로세싱 시스템.

청구항 3

제 2 항에 있어서,

상기 메인 이종 프로세서와 서브 이종 프로세서 간에 연결되며, 상기 서브 이종 프로세서의 레퍼에 의해 변경 또는 정합된 하나 이상의 병렬 함수를 저장하고 있다가, 상기 메인 이종 프로세서 호출시 상기 메인 이종 프로세서에 기구비된 전용 메모리에 전달케 하는 공유 메모리 영역;를 더 포함하는 다수의 이종 프로세서를 구비하는 멀티 프로세싱 시스템.

청구항 4

제 2 항에 있어서,

상기 메인 이종 프로세서는,

상기 서브 이종 프로세서에 의해 컴파일된 하나 이상의 병렬 함수의 프로그램 카운터 값을 모아 상기 메인 프로그램과 함께 재 컴파일하여 생성된 오브젝트 코드를 기구비된 전용 메모리의 어드레스에 저장하는 것을 특징으로 하는 다수의 이종 프로세서를 구비하는 멀티 프로세싱 시스템.

청구항 5

제 4 항에 있어서,

상기 메인 이종 프로세서는,

상기 전용 메모리의 어드레스에 위치한 오브젝트 코드를 조회 가능케 하여 상기 응용 프로그램 시동시, 상기 오브젝트 코드를 즉각 실행시키는 것을 특징으로 하는 다수의 이종 프로세서를 구비하는 멀티 프로세싱 시스템.

청구항 6

제 2 항에 있어서,

상기 레퍼는,

상기 메인 프로그램에 fork 함수를 삽입시킴에 의해 생성된 가상의 매칭 필터로, 상기 컴파일된 하나 이상의 병렬 함수를 상기 메인 이종 프로세서만의 고유 호출 규약 및 데이터 포맷에 맞게 변경 또는 정합시켜 주는 것을 특징으로 하는 다수의 이종 프로세서를 구비하는 멀티 프로세싱 시스템.

청구항 7

레퍼가 1:1 매칭하며 각각 삽입된 다수의 이종 프로세서를 구비하며, 상기 다수의 이종 프로세서 중 메인 이종 프로세서와 서브 이종 프로세서를 선택한 멀티 프로세싱 시스템 상의 구동 방법으로,

기선정된 응응 프로그램이 메인 프로그램과 하나 이상의 병렬 함수로 분리 지정되는 단계;

상기 서브 이종 프로세서에 의해 컴파일된 하나 이상의 병렬 함수가 상기 서브 이종 프로세서에 기구비된 레퍼를 통과하는 단계; 및

상기 레퍼를 통과한 하나 이상의 병렬 함수가 상기 메인 이종 프로세서의 고유 호출 규약과 데이터 포맷에 맞게 변경 또는 정합(整合)하는 단계;를 포함하는 멀티 프로세싱 시스템 상의 구동 방법.

청구항 8

레퍼가 1:1 매칭하며 각각 삽입된 다수의 이종 프로세서를 구비하는 멀티 프로세싱 시스템 상의 구동 방법으로,

상기 다수의 이종 프로세서 중 선택된 메인 이종 프로세서가 기선정된 응용 프로그램을 컴파일하여 메인 프로그램과 하나 이상의 병렬 함수로 분리 지정하는 단계;

상기 다수의 이종 프로세서 중 선택된 서브 이종 프로세서가 상기 메인 이종 프로세서로부터 할당된 하나 이상의 병렬 함수를 자체 구비된 전용 메모리에 매핑하는 단계;

상기 서브 이종 프로세서가 상기 매핑된 하나 이상의 병렬 함수를 컴파일하는 단계;

상기 서브 이종 프로세서에 기구비된 레퍼가 상기 컴파일된 하나 이상의 병렬 함수를 통과하는 단계; 및

상기 레퍼가 상기 컴파일된 하나 이상의 병렬 함수를 상기 메인 이종 프로세서의 고유 호출 규약과 데이터 포맷과 일치한 데이터로 변경 또는 정합(整合)하는 단계;를 포함하는 멀티 프로세싱 시스템 상의 구동 방법.

청구항 9

제 8 항에 있어서,

상기 메인 이종 프로세서와 서브 이종 프로세서 간에 연결된 공유 메모리 영역이 상기 서브 이종 프로세서의 레퍼에 의해 변경 또는 정합된 하나 이상의 병렬 함수를 저장하는 단계; 및

상기 메인 이종 프로세서가 하나 이상의 병렬 함수 호출시, 상기 공유 메모리 영역이 상기 하나 이상의 병렬 함수를 기구비된 전용 메모리에 전달하는 단계;를 더 포함하는 멀티 프로세싱 시스템 상의 구동 방법.

청구항 10

제 8 항에 있어서,

상기 메인 이종 프로세서가,

상기 서브 이종 프로세서에 의해 컴파일된 하나 이상의 병렬 함수의 프로그램 카운터 값을 모아 상기 메인 프로그램과 함께 재 컴파일하는 단계; 및

상기 재 컴파일에 의해 생성된 오브젝트 코드를 기구비된 전용 메모리의 어드레스에 저장하는 단계;를 더 포함하는 멀티 프로세싱 시스템 상의 구동 방법.

청구항 11

제 10 항에 있어서,

상기 메인 이종 프로세서가,

상기 전용 메모리의 어드레스에 위치한 오브젝트 코드를 조회 가능케 하여 상기 응용 프로그램 시동시, 상기 오

브젝트 코드를 즉각 실행시키는 단계;를 더 포함하는 멀티 프로세싱 시스템 상의 구동 방법.

청구항 12

제 8 항에 있어서,

상기 레퍼가,

상기 컴파일된 하나 이상의 병렬 함수를 상기 메인 이종 프로세서만의 고유 호출 규약 및 데이터 포맷에 맞게 매칭시켜 주는 단계;를 더 포함하는 멀티 프로세싱 시스템 상의 구동 방법.

명 세 서

발명의 상세한 설명

기 술 분 야

[0001] 본 발명은 다수의 이종 프로세서 간에 처리되는 멀티 프로세싱 기술에 관한 것으로, 더욱 상세하게는 기선정된 응응 프로그램이 메인 프로그램과 하나 이상의 병렬 함수로 분리 지정된 후, 서브 이종 프로세서에 의해 컴파일된 하나 이상의 병렬 함수가 상기 서브 이종 프로세서에 기구비된 레퍼를 통과함에 따라, 상기 컴파일된 하나 이상의 병렬 함수가 메인 이종 프로세서의 고유 호출 규약과 데이터 포맷에 맞도록 변경 또는 정합(整合)되게 하는 시스템 및 그 구동 방법에 관한 것이다.

배 경 기 술

[0002] 일반적으로, 임베디드 시스템은 다수의 프로세서를 시스템에 포함시킨 후 소프트웨어를 이용하여 어플리케이션을 처리하여 필요한 성능을 발휘하면서 동시에 시스템 자원의 유연성을 가지게 한다. 이는 어플리케이션의 병렬성을 분석하여 다수의 프로세서에게 나누어 할당하고 병렬 처리한다.

[0003] 병렬처리를 위한 멀티 프로세싱 시스템은 하나의 범용 프로세서와 다수의 DSP(Digital Signal Processor)와 같은 보조 프로세서로 구성되며, 범용 프로세서는 운영체제를 처리하거나 어플리케이션 전체 흐름을 제어하는 역할을 하며, 보조프로세서는 많은 양의 데이터를 연산 처리한다.

[0004] 이와 같이, 용도가 적합한 이종의 프로세서들로 구성된 병렬처리를 위한 멀티 프로세싱 시스템은 처리 성능의 향상 및 시스템 자원의 유연성을 높일 수 있다.

[0005] 또한, 병렬처리를 위한 멀티 프로세싱 시스템의 구현 방법은 공유 메모리 방식(Shared Memory Model)과 분산 메모리 방식(Distributed Memory Model)이 있다. 공유 메모리 방식은 둘 이상의 프로세서들이 메모리를 공유하여 사용한다.

[0006] 분산 메모리 방식은 프로세서들이 독립된 메모리 영역을 가지고 계산을 수행하게 된다. 병렬 프로세서상에서의 소프트웨어 개발은 단일 프로세서와 달리 프로세서간의 통신을 고려해주어야 하며, 이는 시스템의 구현방법에 따라 달라진다. 프로그래머가 리를 고려해가며 소프트웨어를 만드는 것은 부담스러운 일이다.

[0007] 또한, 시스템에 의존적으로 만들어진 프로그램은 이식성 역시 떨어지게 된다. 이를 위해, 병렬 프로그래밍에서 표준 API가 필요하며 현재 OPENMP 와 MPI 가 많이 사용되고 있다. OPENMP는 공유 메모리 모델에서 사용되는 API로 멀티쓰레딩(Multi-Threading)을 구현하는 방법 중에 하나이다.

[0008] OPENMP는 마스트 쓰레드가 슬레이브 쓰레드를 필요한 수만큼 생성하고, 생성된 쓰레드들이 개별 프로세서에 할당되어 병렬 처리하게 된다.

[0009] MPI는 분산 메모리 환경을 위한 API로, 독립된 컴퓨터 시스템 간의 통신을 위해 메시지를 통해 추상화된 형식으로 데이터를 송수신하고 있어 이종의 컴퓨터 시스템이 충돌없이 데이터를 주고 받을 수 있으나, 현재의 API 들에선 이종 프로세서들로 구성된 공유 메모리 시스템에 사용하기에 적합하지 않다.

[0010] 또한, OPENMP의 경우에는 멀티 프로세싱 시스템에서 멀티쓰레딩을 구현하에는 훌륭한 표준 API 이긴 하지만, 이종 프로세서들이 아닌 동종 프로세서들로 구성된 공유 메모리 시스템에 사용되도록 만들어졌다.

발명의 내용

해결 하고자하는 과제

[0011] 종래의 임베디드 시스템은 대부분 하나의 마이크로 프로세서로 구성되어 왔으나, 이는 멀티미디어 어플리케이션을 처리하기엔 성능상 충분하지 못하다. 멀티미디어 어플리케이션은 많은 양의 데이터를 빠르게 처리할 수 있어야 하는데 하나의 프로세서만으로 처리할 경우, 고속의 고가 프로세서를 사용하여야 하므로 전력 소모와 비용면에 문제가 있다.

과제 해결수단

[0012] 다수의 이종 프로세서를 구비하는 멀티 프로세싱 시스템 및 그 구동 방법은 기선정된 응응 프로그램이 메인 프로그램과 하나 이상의 병렬 함수로 분리 지정된 후, 서브 이종 프로세서에 의해 컴파일된 하나 이상의 병렬 함수가 상기 서브 이종 프로세서에 기구비된 레퍼를 통과함에 따라, 상기 컴파일된 하나 이상의 병렬 함수가 메인 이종 프로세서의 고유 호출 규약과 데이터 포맷에 맞도록 변경 또는 정합(整合)되게 하는 시스템 및 그 구동 방법을 구현함으로써, 다수의 이종 프로세서 간의 호출 규약 및 데이터 포맷에 대한 차이를 레퍼를 통해 극복하여 기존 개발 툴의 큰 수정없이도 병렬적으로 처리하고자 하는 응용 프로그램을 할당 및 처리할 수 있어 시스템 성능 향상 및 고집적화로 인한 매출 증대를 높이기 위함이다.

[0013] 상기의 과제를 달성하기 위한 본 발명은 다음과 같은 구성을 포함한다.

[0014] 즉, 다수의 이종 프로세서를 구비하는 멀티 프로세싱 시스템은, 레퍼가 1:1 매칭하며 각각 삽입된 다수의 이종 프로세서를 구비하며, 상기 다수의 이종 프로세서 중 메인 이종 프로세서와 서브 이종 프로세서를 선택한 멀티 프로세싱 시스템으로,기선정된 응응 프로그램이 메인 프로그램과 하나 이상의 병렬 함수로 분리 지정된 후, 상기 서브 이종 프로세서에 의해 컴파일된 하나 이상의 병렬 함수가 상기 서브 이종 프로세서에 기구비된 레퍼를 통과함에 따라, 상기 컴파일된 하나 이상의 병렬 함수가 상기 메인 이종 프로세서의 고유 호출 규약과 데이터 포맷에 맞도록 변경 또는 정합(整合)되게 하는 것을 특징으로 한다.

[0015] 본 발명에 따른 다수의 이종 프로세서를 구비하는 멀티 프로세싱 시스템은, 레퍼가 1:1 매칭하며 각각 삽입된 다수의 이종 프로세서를 구비하는 멀티 프로세싱 시스템으로, 상기 다수의 이종 프로세서 중 선택된 하나이며, 소정의 기선정된 응용 프로그램을 컴파일하여 메인 프로그램과 하나 이상의 병렬 함수로 분리 지정하는 메인 이종 프로세서; 및 상기 다수의 이종 프로세서 중 선택된 다른 하나이며, 상기 메인 이종 프로세서로부터 할당된 하나 이상의 병렬 함수를 자체 구비된 전용 메모리에 매핑한 후, 상기 매핑된 하나 이상의 병렬 함수를 컴파일하는 서브 이종 프로세서;를 포함하며, 상기 서브 이종 프로세서에 기구비된 레퍼가 상기 컴파일된 하나 이상의 병렬 함수를 통과시켜 상기 컴파일된 하나 이상의 병렬 함수를 상기 메인 이종 프로세서의 고유 호출 규약과 데이터 포맷과 일치한 데이터로 변경 또는 정합(整合)하는 것을 특징으로 한다.

[0016] 본 발명에 따른 멀티 프로세싱 시스템 상의 구동 방법은, 레퍼가 1:1 매칭하며 각각 삽입된 다수의 이종 프로세서를 구비하며, 상기 다수의 이종 프로세서 중 메인 이종 프로세서와 서브 이종 프로세서를 선택한 멀티 프로세싱 시스템 상의 구동 방법으로, 기선정된 응응 프로그램이 메인 프로그램과 하나 이상의 병렬 함수로 분리 지정되는 단계; 상기 서브 이종 프로세서에 의해 컴파일된 하나 이상의 병렬 함수가 상기 서브 이종 프로세서에 기구비된 레퍼를 통과하는 단계; 및 상기 레퍼를 통과한 하나 이상의 병렬 함수가 상기 메인 이종 프로세서의 고유 호출 규약과 데이터 포맷에 맞게 변경 또는 정합(整合)하는 단계;를 포함한다.

[0017] 본 발명에 따른 멀티 프로세싱 시스템 상의 구동 방법은, 레퍼가 1:1 매칭하며 각각 삽입된 다수의 이종 프로세서를 구비하는 멀티 프로세싱 시스템 상의 구동 방법으로, 상기 다수의 이종 프로세서 중 선택된 메인 이종 프로세서가 기선정된 응용 프로그램을 컴파일하여 메인 프로그램과 하나 이상의 병렬 함수로 분리 지정하는 단계; 상기 다수의 이종 프로세서 중 선택된 서브 이종 프로세서가 상기 메인 이종 프로세서로부터 할당된 하나 이상의 병렬 함수를 자체 구비된 전용 메모리에 매핑하는 단계; 상기 서브 이종 프로세서가 상기 매핑된 하나 이상의 병렬 함수를 컴파일하는 단계; 상기 서브 이종 프로세서에 기구비된 레퍼가 상기 컴파일된 하나 이상의 병렬 함수를 통과하는 단계; 및 상기 레퍼가 상기 컴파일된 하나 이상의 병렬 함수를 상기 메인 이종 프로세서의 고유 호출 규약과 데이터 포맷과 일치한 데이터로 변경 또는 정합(整合)하는 단계;를 포함한다.

효 과

[0018] 본 발명에서의 다수의 이종 프로세서를 구비하는 멀티 프로세싱 시스템 및 그 구동 방법은 기선정된 응응 프로그램이 메인 프로그램과 하나 이상의 병렬 함수로 분리 지정된 후, 서브 이종 프로세서에 의해 컴파일된 하나

이상의 병렬 함수가 상기 서브 이종 프로세서에 기구비된 레퍼를 통과함에 따라, 상기 컴파일된 하나 이상의 병렬 함수가 메인 이종 프로세서의 고유 호출 규약과 데이터 포맷에 맞도록 변경 또는 정합(整合)되게 하는 시스템 및 그 구동 방법을 구현함으로써, 다수의 이종 프로세서 간의 호출 규약 및 데이터 포맷에 대한 차이를 레퍼를 통해 극복하여 기존 개발 툴의 큰 수정없이도 병렬적으로 처리하고자 하는 응용 프로그램을 할당 및 처리할 수 있어 시스템 성능 향상 및 고집적화로 인한 매출 증대를 높이는 효과를 준다.

발명의 실시를 위한 구체적인 내용

[0019] [실시예]

[0020] 이하, 본 발명의 실시예를 첨부된 도면을 참조하여 상세히 설명하면 다음과 같다.

[0021] 도 1은 본 발명의 실시예에 따른 이종의 프로세서들로 구성된 멀티 프로세싱 시스템을 도시한 도면이다.

[0022] 도 1을 참조하면, 멀티 프로세싱 시스템(1000)은 N개의 이종 프로세서들의 집합으로 이루어진 프로세서 그룹부(100), N개의 이종 프로세서들마다 각각 1:1 매칭으로 연결된 N개의 전용 메모리의 집합으로 이루어진 전용 메모리 그룹부(200), 버스, 공유 메모리(300) 및 직접 메모리 엑세스(400)를 포함한다.

[0023] 일반적으로, 프로세서 그룹부(100)는 N개의 동종 프로세서들의 집합으로 구성될 수 있고, N개의 이종 프로세서들의 집합으로도 구성될 수 있다.

[0024] 본 발명에 따른 프로세서 그룹부(100)에 형성된 N개의 이종 프로세서들의 데이터 병렬처리 기술은 N개의 동종 프로세서들에선 당연히 처리 가능함으로, N개의 동종 프로세서들에 대한 데이터 병렬처리에 대해선 이하 설명에서 추가 언급하지 않더라도 당연히 병렬처리 가능함을 전제로 하고 상세 기술하고자 한다.

[0025] 다시 말해, N개의 동종 프로세서들의 데이터 병렬 처리는 N개의 이종 프로세서들에서보다 그 처리 과정이 쉽고 간단하게 실시됨으로, N개의 이종 프로세서들이 데이터를 병렬 처리 가능하다는 것은 N개의 동종 프로세서들의 데이터 병렬 처리에선 더 이상 언급할 필요도 없이 당연히 실시 가능함을 뜻한다 하겠다.

[0026] 따라서, N개의 동종 프로세서들의 데이터 병렬 처리는 N개의 이종 프로세서들의 데이터 병렬 처리 과정을 통해 설명되는바 별도 추가 설명은 생략함을 미리 알려둔다.

[0027] 먼저, 프로세서 그룹부(100)는 버스와 연결되며, 프로세서 그룹부(100)에 형성된 각각의 N개 이종 프로세서들은 전용 메모리 그룹부(200)에 형성된 N개의 각 전용 메모리들과 1:1 매칭 연결된다.

[0028] 프로세서 그룹부(100)의 N개의 이종 프로세서들은 직접 메모리 엑세스(400)의 메모리 접근을 통해 공유 메모리(300)와 연결되거나 혹은 공유 메모리에 바로 엑세스하는 방법을 선택하여 N개의 이종 프로세서들 중 어느 하나 이상이 공유 메모리(300)에 접근하여 기저장된 데이터를 요청할 시에 기저장된 데이터의 읽기/쓰기 모두를 제공받는다.

[0029] N개의 이종 프로세서들은 N개의 전용 메모리들과 각각 1:1 매칭하며 독립적으로 연결되어 있어, 비밀이 요구되거나 단독처리가 요구되는 데이터를 즉각 처리한다.

[0030] 멀티 프로세싱 시스템(1000)상의 N 개의 이종 프로세서들 간 연결 관계는 태스크 할당, 프로세서 간 통신 및 동기화에 영향을 미친다.

[0031] N 개의 이종 프로세서들 간 연결 관계는 플랫폼이 핵심임으로, 플랫폼 설계는 멀티 프로세싱 시스템(1000)의 성능향상, 프로그램 이식 및 프로그램 개발을 용이하게 하는데 중요한 요소이다.

[0032] 플랫폼 설계는 N 개의 이종 프로세서들 간 밀접성에 따라 긴밀 결합(tightly coupled)과 비긴밀 결합(loosely coupled)으로 나눌 수 있다.

[0033] 즉, 긴밀 결합인 경우, N 개의 이종 프로세서들이 서로 긴밀히 연결되어 있어, N 개의 이종 프로세서들 중 메인 프로세서로 선택된 하나는 전체 멀티 프로세싱 시스템(1000)의 성능향상을 위해 나머지 N-1개의 이종 프로세서들을 긴밀하게 통제 및 제어한다.

[0034] 이와 반대로 비긴밀 결합에 경우, N 개의 이종 프로세서들 간 통신 및 동기화는 반드시 별도의 장치 드라이브의 API(Application Program Interface)를 거쳐야 하고, N 개의 이종 프로세서들 간 데이터 공유는 공유 메모리(300)의 사용이 불가능함으로 명시적 데이터 복사를 통해서만 추가적인 데이터를 발생시킨다.

[0035] 멀티 프로세싱 시스템(1000)은 긴밀 결합이 성립된 상태에서 데이터 병렬 처리함으로, N 개의 이종 프로세서들

간 연결은 특정 장치에 해당하는 장치 드라이브나 API가 없어도 바로 접근 용이하다.

[0036] 이와 동시에, 멀티 프로세싱 시스템(1000)은 어셈블리 언어(일명 : 데이터 병렬처리 가상 머신) 또는 별도의 라이브러리를 사용하지 않더라도 소스 코딩 작업만으로 프로그램의 이식성을 높일 수 있고, 디렉티브의 추가만으로 프로그램을 병렬 처리 환경으로 가져오게끔 유도한다.

[0037] 멀티 프로세싱 시스템(1000)은 N 개의 이종 프로세서들마다 레퍼(Wrapper)를 각각 구비시킴으로써, 신호 처리가 각각 다른 N 개의 이종 프로세서들 간의 연결 관계를 극복하고, 상기에서 언급된 N 개의 이종 프로세서들 간 긴밀 결합을 구현할 수 있다.

[0038] 본 발명에 따른 멀티 프로세싱 시스템(1000)은 기형성된 N 개의 이종 프로세서들마다 각각 서로 다른 호출 규약(Calling Convention)을 가지고 있어, N 개의 이종 프로세서들마다 구비된 레퍼를 이용하여 서로 다른 호출 규약을 매칭 조율함으로써, 매개 변수에 따른 해당 함수를 호출시킨다.

[0039] 즉, 하나의 프로세서에 구비된 레퍼는 자신의 호출 규약을 다른 하나의 프로세서의 호출 규약에 맞춤으로써, 다른 하나의 이종 프로세서로 하여금 할당하고자 하는 테스크(Task)에 대한 해당 함수를 호출토록 유도한다.

[0040] 여기서, 호출 규약은 매개 변수를 함수에 전달하여 이에 따른 결과값을 함수로부터 가져오게 하는 프로그램 표준운영 약속으로, 첫째, 레지스터 또는 스택 중 적어도 하나에 매개 변수와 그 결과값을 당해 어드레스에 위치시키고 둘째, 해당 함수를 호출하는 하나의 이종 프로세서의 호출 규약을 따를 것인지 아니면 해당 함수를 호출받는 다른 하나의 이종 프로세서의 호출 규약을 따를 것인지 결정짓는 것이 필요하다 하겠다.

[0041] 하나 더, 본 발명에 따른 멀티 프로세싱 시스템(1000)은 기형성된 N 개의 이종 프로세서들마다 각각 서로 다른 데이터 포맷을 형성하고 있어, 래퍼는 N 개의 이종 프로세서들마다 이미 다르게 정해진 엔디안 방식(Endianness), 워드 사이즈(Word Size) 및 데이터 타입(Data Type) 등의 데이터 포맷을 약속 규정에 따라 정함으로써, 에러 발생이 제거된 데이터 연산을 실시하여 양단 간 데이터 통신을 유도한다.

[0042] 도 2는 본 발명의 실시예에 따른 서로 다른 호출 규약을 갖는 이종 프로세서 간 차이를 극복하기 위한 래퍼 활용 방법을 도시한 도면이다.

[0043] 도 2를 참조하면, 멀티 프로세싱 시스템은 기구비된 복수의 이종 프로세서 간의 서로 다른 호출 규약을 래퍼를 이용하여 매칭시켜 줌으로써, 할당하고자 하는 테스크(Task)에 대한 해당 함수를 호출한다.

[0044] 즉, 레퍼는 복수의 이종 프로세서 간에 기형성된 서로 다른 호출 규약을 상호 조율해주는 역할을 한다.

[0045] 본 발명에 따른 서로 다른 호출 규약을 갖는 이종 프로세서 간 차이를 극복하기 위한 래퍼 활용 방법의 일예로, 먼저, 멀티 프로세싱 시스템 내에 X86 프로세서(110)와 파워PC 프로세서(120)가 구비된 상태이다.

[0046] 우선, 메인 프로세서로 설정된 X86 프로세서(110)가 보조 프로세서로 설정된 파워PC 프로세서(120)로 해당 함수를 호출토록 매개 변수를 기구비된 자신의 x86 레지스터(111)를 사용하는 대신에 X86 스택(112)에 넘겨준다 ①.

[0047] 그러나, 파워PC 프로세서(120)의 호출 규약과 X86 프로세서(110)의 호출 규약이 서로 다르게 형성되어 있는 까닭에 파워PC 프로세서(120)는 X86 프로세서(110)의 X86 레지스터(111)를 통해 매개 변수가 넘어오는 것으로 인식한다.

[0048] 이러한 이유로, 메인 프로세서의 X86 프로세서(110)는 자신의 X86 스택(112)과 공유 스택(130) 간에 형성된 X86 레퍼(140)를 가상의 매칭 필터로 이용하여 X86 스택(112)에 저장된 매개 변수를 공유 스택(130)에 전달한다 ②.

[0049] X86 레퍼(140)가 이종 프로세서 간의 서로 다른 호출 규약을 맞춰 주는데 사용되는 가상의 매칭 필터로 동작됨에 따라, X86 레퍼(140)는 X86 프로세서(110)와 파워PC 프로세서(120) 간 서로 다른 호출 규약을 매칭 조율하여 X86 프로세서(110)의 고유 호출 규약을 파워PC 프로세서(120)의 고유 호출 규약으로 변경 또는 정합시켜 준다.

[0050] 즉, X86 래퍼(140)는 공유 스택(130)에 쌓여진 해당 함수의 주소와 매개 변수를 확인하여 파워PC 프로세서(120)에 구비된 파워PC 래퍼(150)로 해당 함수가 호출됨을 공지시켜 준다 ③.

[0051] 양단 간의 호출 규약이 정합된 파워PC 프로세서(120)의 파워PC 레지스터(121)는 공유 스택(130)으로부터 전달된 매개 변수를 순차적으로 수신한 후, 자신만의 고유 호출 규약에 맞게 공유 스택(130)에 쌓인 하나 이상의 매개 변수를 복사하여 매개 변수에 따른 결과값이 나오도록 호출된 해당 함수를 실행시킨다 ④.

[0052] 결과적으로, x86 레퍼(140)는 X86 프로세서(110)와 파워PC 프로세서(120) 간에 매개 변수를 저장하는 메모리 공

간이 서로 다르더라도 매개 변수로의 해당 함수실행에 대한 결과값이 산출되도록 지원해 준다.

[0053] 서로 다른 호출 규약을 갖는 이종 프로세서 간 차이를 극복하기 위한 래퍼 활용 방법에 대해 좀 더 추가적으로 설명하자면 다음과 같다.

[0054] 메인 프로세서로 설정된 X86 프로세서(110)는 자신만의 고유 호출 규약에 따라 자신의 x86 스택(112)에 매개 변수를 쌓고 fork 함수를 호출한다 ①.

[0055] fork 함수는 X86 프로세서(110)에 구비된 X86 래퍼(140)를 호출한다.

[0056] x86 레퍼(140)는 보조 프로세서인 파워PC 프로세서(120)에 의해 추후 호출될 해당 함수의 주소와 매개 변수를 공유 스택(Shared Stack : 130)에 미리 쌓아둔다 ②.

[0057] 여기서, 공유 스택(130)으로 사용될 수 있는 저장 장치로선 다양한 메모리 장치 중에 선택된 하나를 탑재시킬 수도 있고, X86 프로세서(110)와 파워PC 프로세서(120) 모두에 접근 가능한 어느 특정 레지스터 장치를 탑재시킬 수도 있다.

[0058] X86 프로세서(110)의 X86 래퍼(140)는 공유 스택(130)에 쌓여진 해당 함수의 주소와 매개 변수를 확인하여 파워PC 프로세서(120)의 파워PC 래퍼(150)에게 해당 함수가 호출됨을 수행토록 하는 함수호출 알림 방법을 통해 공지시켜 준다 ③.

[0059] 여기서, 함수호출 알림 방법은 인터럽트(Interrupt)를 발생시켜 파워PC 프로세서(120)에게 전달해 주거나, 파워PC 프로세서(120) 내의 특정 메모리 또는 레지스터를 폴링(Polling)하는 방식이 있다.

[0060] 이에 따라, 파워PC 프로세서(120)의 파워PC 래퍼(140)는 자신만의 고유 호출 규약에 맞게 공유 스택(130)에 쌓인 하나 이상의 매개 변수를 파워PC 스택(122)이 아닌 파워PC 레지스터(121)에 복사하여 매개 변수에 따른 결과값이 나오도록 해당 함수를 실행시킨다 ④.

[0061] 도 3은 본 발명의 실시예에 따른 서로 다른 데이터 포맷을 형성하는 이종 프로세서 간 차이를 극복하기 위한 래퍼 활용 방법을 도시한 도면이다.

[0062] 도 3을 참조하면, 멀티 프로세싱 시스템은 기구비된 이종 프로세서 양단 간에 서로 다른 호출 규약이 레퍼를 통해 해결되었다 하더라도, 이종 프로세서마다 메모리 접근 방법이 다를 경우엔 양방향의 데이터 통신은 불가능하다.

[0063] 래퍼는 이종 프로세서마다 이미 다르게 정해진 엔디안 방식(Endianness), 워드 사이즈(Word Size) 및 데이터 타입(Data Type) 등의 데이터 포맷을 약속 규정에 따라 정함으로써, 에러 발생이 제거된 데이터 연산을 실시하여 양단 간 데이터 통신을 유도한다.

[0064] 서로 다른 데이터 포맷을 형성하는 이종 프로세서 간 차이를 극복하기 위한 래퍼 활용 방법의 일 예를 설명하자면 다음과 같다.

[0065] 즉, 워드 사이즈는 16비트이고 리틀 엔디안 방식(Little Endianness)을 사용하는 A 프로세서(110)가 공유 메모리 영역(130)에 접근하여 32 비트의 데이터를 읽어드릴 경우, A 프로세서(110)는 0,1,2,3으로 되어 있는 32 비트 데이터를 0x00 번지와 0X02 번지 나누고 왼쪽부터 낮은 주소 순으로 두 차례 걸쳐 읽힌 1,0,3,2를 A 전용 메모리(Dedicated Area for A : 210)에 저장한다.

[0066] 또한, 워드 사이즈는 32 비트이고 빅 엔디안 방식(Big Endianness)을 사용하는 B 프로세서(120)가 공유 메모리 영역(130)에 접근하여 32 비트의 데이터를 읽어드릴 경우, B 프로세서(120)는 0x00 번지에서부터 32 비트의 데이터를 한번에 모두 읽어 1,2,3,4를 B 전용 메모리(Dedicated Area for B : 220)에 저장한다.

[0067] 이때, 이종의 A 프로세서(110)와 B 프로세서(120)가 서로 데이터 공유하기 위해선 데이터 형식을 일치시켜 주기 위한 약속 규정이 필요하다.

[0068] 만약, 양단(이종의 A 프로세서와 B 프로세서 : 110, 120)간 B 프로세서(120)의 데이터 포맷을 약속 규정으로 삼을 경우, A 프로세서(110)는 공유 메모리 영역(130) 접근시 B 프로세서(120)의 데이터 포맷을 따라야 한다.

[0069] 이에 따라, 본 발명의 레퍼 활용 방법은 A 프로세서(110)가 공유 메모리 영역(130)에 존재하는 공유 데이터에 접근할 시, 레퍼는 접근하려는 주소가 공유 메모리 영역(130)을 지칭하는 것인지 여부를 판단한다.

[0070] 접근하려는 주소가 공유 메모리 영역(130)을 지칭하는 것으로 판단될 경우, 레퍼는 현재의 A 프로세서(110)의

데이터 포맷을 B 프로세서(120)의 데이터 포맷 형식으로 변환하여 32 비트 데이터를 읽어 오게끔 한다.

[0071] 허나, 접근하려는 주소가 공유 메모리 영역(130)을 지칭하지 않는 것으로 판단될 경우, 레퍼는 현재의 A 프로세서(110)의 데이터 포맷을 그대로 유지시켜 32 비트 데이터를 읽어 오게끔 한다.

[0072] 반대의 경우에 있어서도 마찬가지로, 양단(이종의 A 프로세서와 B 프로세서 : 110, 120)간 A 프로세서(110)의 데이터 포맷을 약속 규정으로 삼을 경우엔, B 프로세서(120)는 공유 메모리 영역(130) 접근시 A 프로세서(110)의 데이터 포맷을 따라야 한다.

[0073] B 프로세서(120)가 공유 메모리 영역(130)에 존재하는 공유 데이터에 접근할 시, 레퍼는 접근하려는 주소가 공유 메모리 영역(130)을 지칭하는 것인지 여부를 판단한다.

[0074] 접근하려는 주소가 공유 메모리 영역(130)을 지칭하는 것으로 판단될 경우, 레퍼는 현재의 B 프로세서(120)의 데이터 포맷을 A 프로세서(110)의 데이터 포맷 형식에 맞게 변환 및 정합시켜 32 비트 데이터를 읽어 오게끔 한다.

[0075] 허나, 접근하려는 주소가 공유 메모리 영역을 지칭하지 않는 것으로 판단될 경우, 레퍼는 현재의 B 프로세서(120)의 데이터 포맷을 그대로 유지시켜 32 비트 데이터를 읽어 오게끔 한다.

[0076] 본 일 예에선, A 프로세서(110)의 데이터 포맷이 B 프로세서(120)의 데이터 포맷보다 워드 사이즈가 작으므로, B 프로세서(120)의 성능이 우수함을 바로 인식할 수 있어 B 프로세서(120)의 데이터 포맷을 약속 규정하는 것이 바람직함을 일러둔다.

[0077] 도 4는 본 발명의 실시예에 따른 래퍼를 이용한 소프트웨어 빌드 방법을 도시한 도면이다.

[0078] 도 4를 참조하면, 래퍼를 이용한 소프트웨어 빌드 방법은 기선정된 응응 프로그램이 메인 프로그램과 하나 이상의 병렬 함수로 분리 지정된 후, 서브 이종 프로세서에 의해 컴파일된 하나 이상의 병렬 함수가 상기 서브 이종 프로세서에 기구비된 레퍼를 통과함에 따라, 상기 컴파일된 하나 이상의 병렬 함수가 메인 이종 프로세서의 고유 호출 규약과 데이터 포맷에 맞도록 변경 또는 정합(整合)되게 하는 멀티 프로세싱 시스템 상의 구동 방법이다.

[0079] 먼저, 메인 이종 프로세서는 병렬적 디렉티브를 적용할 수 있는 프로그램 언어로 코딩된 응용 프로그램을 선정한다(S1).

[0080] 단, 프로그램머는 응용 프로그램의 소스 코드를 분석하여 코드의 병렬성을 파악하고, 병렬화된 응용 프로그램에서 인식할 수 있는 디렉티브를 사용하여 병렬적으로 실행시킬 응용 프로그램의 소스 코드를 공유 메모리 영역에 존재케 한 후, 키워드를 이용하여 단락별로 주석 표시해 둔다.

[0081] 메인 이종 프로세서는 응용 프로그램의 소스 코드를 컴파일러를 통해 컴파일함에 따라(S2), 메인 프로그램과 메인 프로그램이 아닌 구조체 형태를 갖는 하나 이상의 병렬 함수로 나눠 분리 지정한다(S3, S4).

[0082] 메인 이종 프로세서는 메인 프로그램을 순차 처리할 준비를 개시하고, 서브 이종 프로세서는 하나 이상의 병렬 함수를 할당받아 쓰레드(Thread) 처리할 준비를 개시한다.

[0083] 여기서, 쓰레드는 응용 프로그램 수행 시 프로세스 내부에 존재하는 수행 경로 실제 작업을 담당한다. 프로세스 생성시 하나의 주 스레드가 생성되어 대부분의 작업을 처리하고 주 스레드가 종료되면 프로세스도 종료된다. 하나의 프로세스 내에서 다수의 스레드가 동시에 수행되기도 한다.

[0084] 서브 이종 프로세서가 하나 이상의 병렬 함수를 쓰레드(Thread) 처리할 준비를 개시하기 위해선, 미리 자신의 현 상태를 초기화하여 메인 이종 프로세서 호출시 즉각 응답할 수 있도록 대기 상태로 만들어 주고, 하나 이상의 병렬 함수를 자체 구비된 전용 메모리에 매핑시켜 준다.

[0085] 서브 이종 프로세서는 쓰레드(Thread) 처리할 준비를 완료함에 따라, 전용 메모리에 매핑된 하나 이상의 병렬 함수(A, B, C)를 다수의 컴파일러(A, B, C)를 이용하여 각각 컴파일한다(S5).

[0086] 서브 이종 프로세서는 컴파일된 하나 이상의 병렬 함수를 서브 이종 프로세서에 기구비된 레퍼를 통과시켜 컴파일된 하나 이상의 병렬 함수가 메인 이종 프로세서의 고유 호출 규약과 데이터 포맷에 맞도록 변경 또는 정합(整合)을 실시한다(S6).

[0087] 이에 따라, 메인 이종 프로세서의 고유 호출 규약과 데이터 포맷에 맞도록 변경 또는 정합(整合)된 하나 이상의 병렬 함수는 공유 메모리 영역에 위치하고 있다가 메인 이종 프로세서 호출시 메인 이종 프로세서에 기구비된

전용 메모리에 전달된다(S7, S8).

[0088] 여기서, 소프트웨어 빌드시 사용되는 래퍼는 컴파일된 하나 이상의 병렬 함수를 메인 이종 프로세서만의 고유 호출 규약 및 데이터 포맷에 맞도록 구현하기 위해 메인 프로그램에 fork 함수를 삽입하여 호출한 가상의 매칭 필터이다.

[0089] 메인 이종 프로세서는 서브 이종 프로세서에 의해 컴파일된 하나 이상의 병렬 함수의 PC(Program Counter, 일명 : 포인터 지정, 스퀀스 지정으로도 호칭됨)값을 모아 메인 프로그램과 함께 다시 컴파일한다(S9).

[0090] 이 때도, 메인 이종 프로세서에 구비된 래퍼는 메인 이종 프로세서의 전용 메모리에 위치한 상태이고, 재 컴파일 실시할 예정인 메인 프로그램과 하나 이상의 병렬 함수를 대상으로 자신의 고유 호출 규약과 데이터 포맷이 일치하는지를 재검토 및 재확인한다(S6).

[0091] 메인 이종 프로세서는 재 컴파일을 실시하기 전에 다음과 같은 작업을 선행하는데, 메인 이종 프로세서는 서브 이종 프로세서와 공동으로 사용하고 있는 공유 메모리 영역에 저장된 컴파일된 하나 이상의 병렬함수를 자체 구비된 전용 메모리에 할당하고, 링커(Linker)를 이용하여 컴파일된 하나 이상의 병렬 함수를 모아 메인 프로그램과 서로 링크(Link)시켜 주는 작업을 먼저 선행한다.

[0092] 이어서, 메인 이종 프로세서는 서브 이종 프로세서에 의해 컴파일된 하나 이상의 병렬 함수로부터 추출한 PC(Program Counter, 일명 : 포인터 지정, 스퀀스 지정으로도 호칭됨)값을 모아 메인 프로그램과 함께 재 컴파일함에 따라 일괄적으로 생성된 오브젝트 코드(Object Code)를 전용 메모리의 어드레스(address)에 저장한다(S10).

[0093] 메인 이종 프로세서는 전용 메모리의 어드레스에 위치한 오브젝트 코드를 조회 가능케 함으로써 응용 프로그램 시동(始動)시, 즉각적으로 오브젝트 코드가 실행되게끔 한다.

[0094] 상기에서는 본 발명의 바람직한 실시예를 참조하여 설명하였지만, 해당 기술 분야의 숙련된 당업자는 하기의 특허청구범위에 기재된 본 발명의 사상 및 영역으로부터 벗어나지 않는 범위 내에서 본 발명을 다양하게 수정 및 변경시킬 수 있음을 이해할 수 있을 것이다.

도면의 간단한 설명

[0095] 도 1은 본 발명의 실시예에 따른 이종의 프로세서들로 구성된 멀티 프로세싱 시스템을 도시한 도면이다.

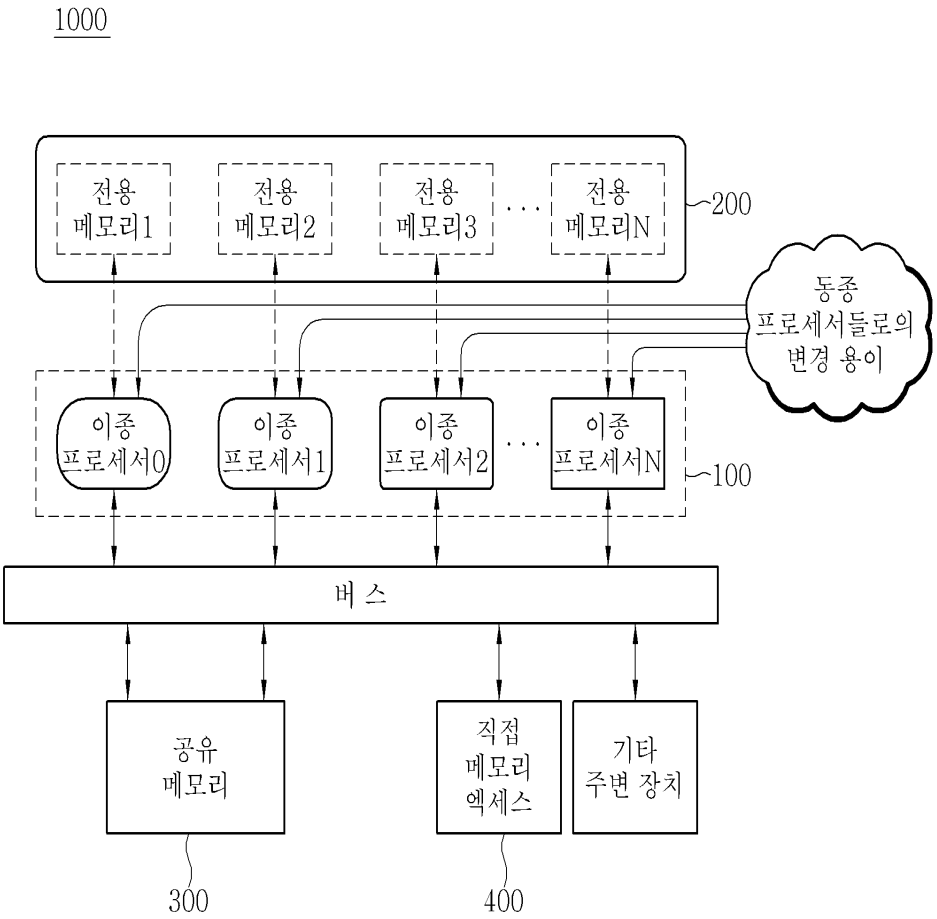
[0096] 도 2는 본 발명의 실시예에 따른 서로 다른 호출 규약을 갖는 이종 프로세서 간 차이를 극복하기 위한 래퍼 활용 방법을 도시한 도면이다.

[0097] 도 3은 본 발명의 실시예에 따른 서로 다른 데이터 포맷을 형성하는 이종 프로세서 간 차이를 극복하기 위한 래퍼 활용 방법을 도시한 도면이다.

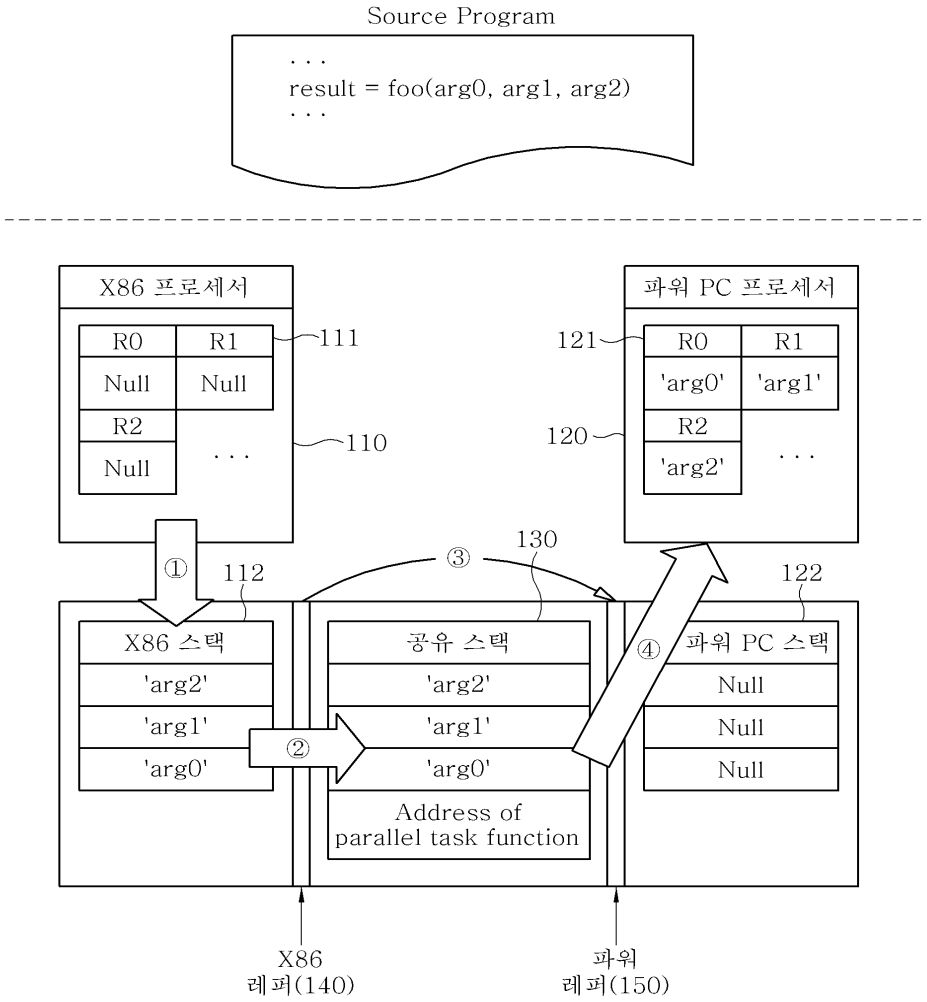
[0098] 도 4는 본 발명의 실시예에 따른 래퍼를 이용한 소프트웨어 빌드 방법을 도시한 도면이다.

도면

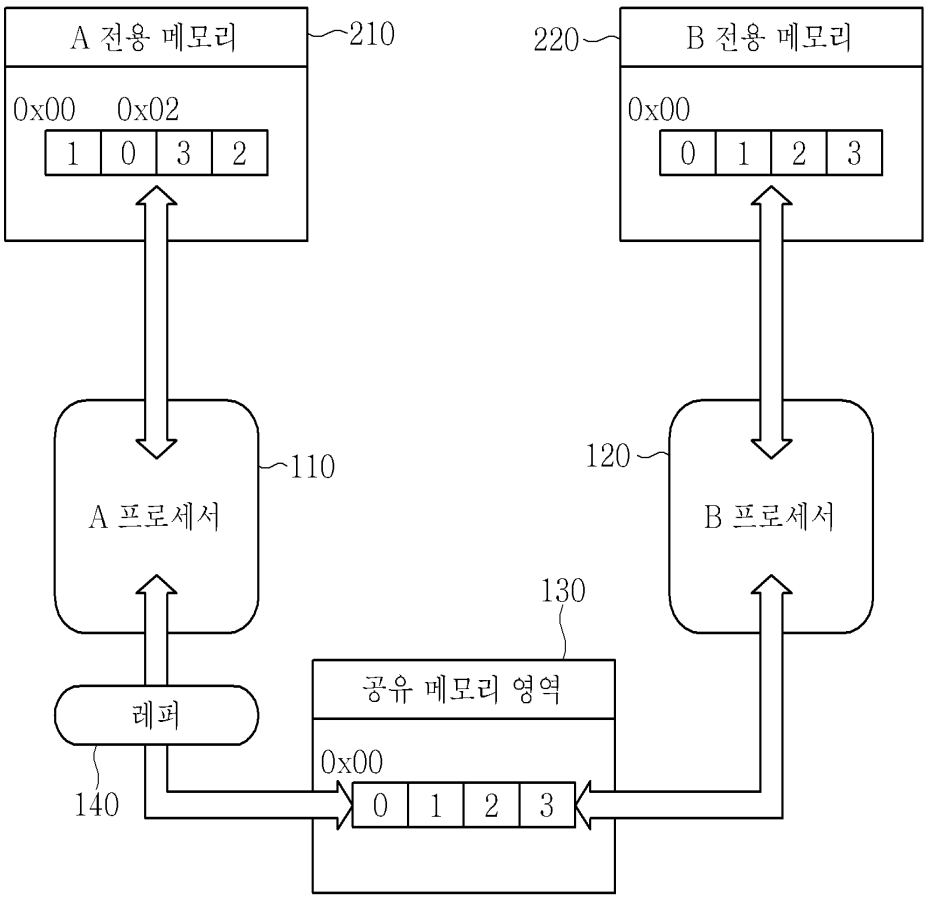
도면1


1000
전용 메모리1
전용 메모리2
전용 메모리3
전용 메모리N
200
동종 프로세서들로의 변경 용이
이종 프로세서0
이종 프로세서1
이종 프로세서2
이종 프로세서N
100
버 스
공유 메모리
300
직접 메모리 엑세스
400
기타 주변 장치

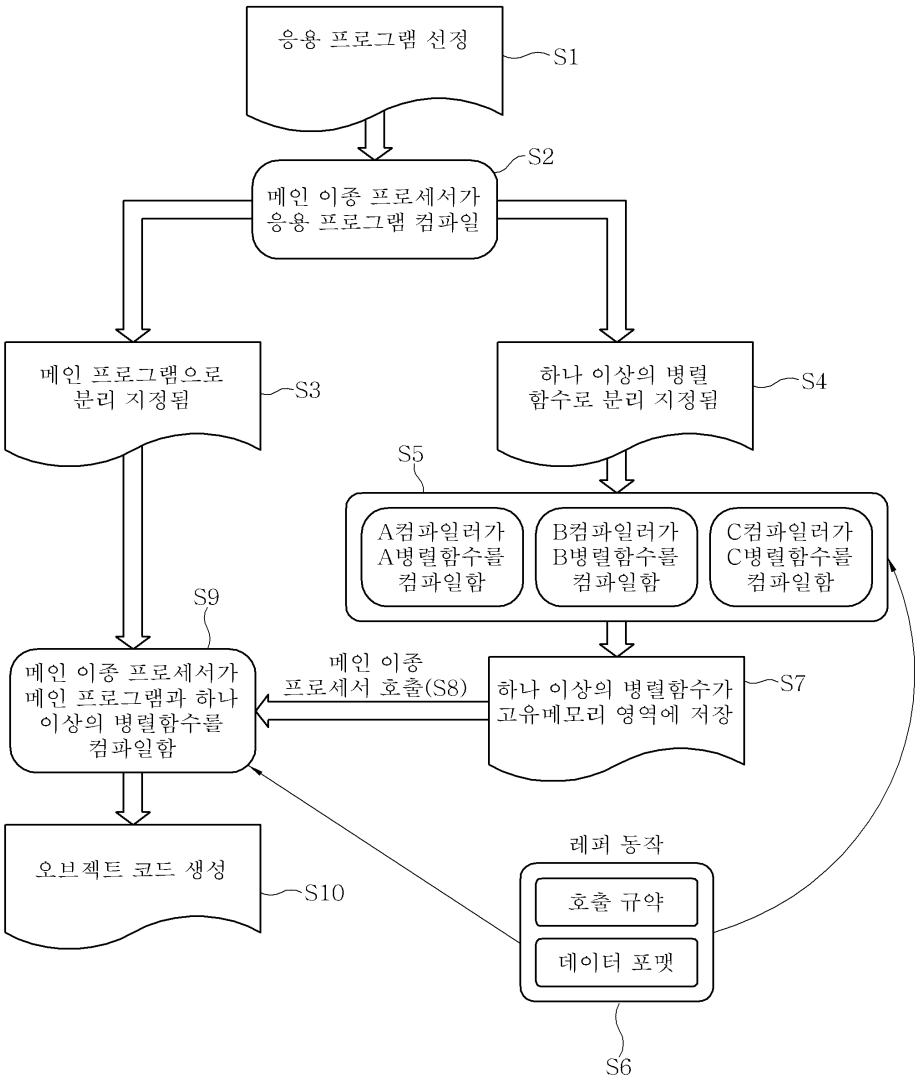
도면2


Source Program
. . .
result = foo(arg0, arg1, arg2)
. . .
X86 프로세서
R0
R1
Null
Null
R2
Null
. . .
111
110
파워 PC 프로세서
121
R0
R1
'arg0'
'arg1'
R2
'arg2'
. . .
120
①
②
③
④
130
112
122
X86 스택
'arg2'
'arg1'
'arg0'
공유 스택
'arg2'
'arg1'
'arg0'
Address of
parallel task function
파워 PC 스택
Null
Null
Null
X86
레퍼(140)
파워
레퍼(150)

도면3


A 전용 메모리
210
0x00
0x02
1
0
3
2
220
B 전용 메모리
0x00
0
1
2
3
110
A 프로세서
120
B 프로세서
130
공유 메모리 영역
레퍼
140
0x00
0
1
2
3

도면4


응용 프로그램 선정
S1
S2
메인 이종 프로세서가
응용 프로그램 컴파일
메인 프로그램으로
분리 지정됨
S3
하나 이상의 병렬
함수로 분리 지정됨
S4
S5
A컴파일러가
A병렬함수를
컴파일함
B컴파일러가
B병렬함수를
컴파일함
C컴파일러가
C병렬함수를
컴파일함
S9
메인 이종 프로세서가
메인 프로그램과 하나
이상의 병렬함수를
컴파일함
메인 이종
프로세서 호출(S8)
하나 이상의 병렬함수가
고유메모리 영역에 저장
S7
오브젝트 코드 생성
S10
레퍼 동작
호출 규약
데이터 포맷
S6