(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2025년06월13일
(11) 등록번호 10-2821210
(24) 등록일자 2025년06월11일

(51) 국제특허분류(Int. Cl.)
G06F 11/16 (2006.01) *G06F 15/163* (2006.01)
(52) CPC특허분류
G06F 11/1641 (2013.01)
G06F 15/163 (2013.01)
(21) 출원번호 10-2018-0123896
(22) 출원일자 2018년10월17일
심사청구일자 2021년10월06일
(65) 공개번호 10-2019-0058288
(43) 공개일자 2019년05월29일
(30) 우선권주장
15/819,402 2017년11월21일 미국(US)
(56) 선행기술조사문헌
EP2175371 A1*
(뒷면에 계속)
전체 청구항 수 : 총 10 항

(73) 특허권자
더 보잉 컴파니
미국, 일리노이스 60606, 시카고, 100 노스 리버사이드 플라자
(72) 발명자
리, 윙 씨.
미국 60606 일리노이스 시카고 노스 리버사이드 플라자 100
래미, 션 엠.
미국 60606 일리노이스 시카고 노스 리버사이드 플라자 100
(뒷면에 계속)
(74) 대리인
특허법인(유)남아이피그룹, 특허법인 남앤남

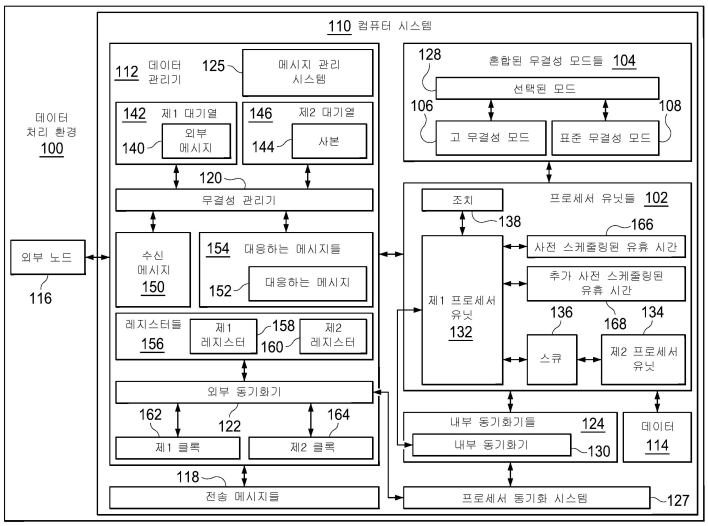
심사관 : 김계준

(54) 발명의 명칭 **명령 처리 정렬 시스템**

(57) 요 약

프로세서 유닛들을 동기화하기 위한 방법이 제공된다. 동기화 시스템과 통신하는 제1 프로세서 유닛(132)과 제2 프로세서 유닛(134) 사이에 원하지 않는 양의 스큐(136)가 존재하는지 여부를 결정하기 위해 외부 동기화기(122)와 통신이 이루어진다. 제1 프로세서 유닛(132)은 제1 프로세서 유닛(132)에 제1 프로세서 유닛(132)과 제2 프로세서 유닛(134) 사이의 원하지 않는 양의 스큐(136)가 존재할 때 원하지 않는 양의 스큐(136)가 감소되게, 필요한 결과를 생성하지 않고 조치(138)를 수행하도록 선택적으로 지시를 받는다. 제1 프로세서 유닛(132) 및 제2 프로세서 유닛(134)에 의해 생성된 대응하는 메시지들에 대해 무결성 검사들이 수행되는 고 무결성 모드(106)를 위해 제1 프로세서 유닛(132)과 제2 프로세서 유닛(134)이 서로 연관된다.

대 표 도 - 도1



(72) 발명자

쿤츠, 로날드 제임스

미국 60606 일리노이스 시카고 노스 리버사이드 플라자 100

웡, 딕 피.

미국 60606 일리노이스 시카고 노스 리버사이드 플라자 100

치아, 잭슨

미국 60606 일리노이스 시카고 노스 리버사이드 플라자 100

포나바이오, 안토니 에스.

미국 60606 일리노이스 시카고 노스 리버사이드 플라자 100

란가라잔, 무라리

미국 60606 일리노이스 시카고 노스 리버사이드 플라자 100

무어, 클라크 에드가

미국 60606 일리노이스 시카고 노스 리버사이드 플라자 100

샤프, 데이비드 클라이드

미국 60606 일리노이스 시카고 노스 리버사이드 플라자 100

노르드직, 아놀드 더블유.

미국 60606 일리노이스 시카고 노스 리버사이드 플라자 100

덴젤, 폴 유진

미국 60606 일리노이스 시카고 노스 리버사이드 플라자 100

(56) 선행기술조사문헌

WO2007006013 A2*

US20100262811 A1

US8924780 B2

KR1020080068710 A

KR1020070083772 A

EP02175371 A1*

*는 심사관에 의하여 인용된 문헌

명 세 서

청구범위

청구항 1

프로세서 동기화 시스템(127)으로서,

제1 프로세서 유닛(132) 상에서 실행되는 내부 동기화기(130)를 포함하며,

상기 내부 동기화기(130)는 외부 동기화기(122)와 통신하는 제1 프로세서 유닛(132)과 제2 프로세서 유닛(134) 사이에 원하지 않는 양의 스큐(skew)(136)가 존재하는지 여부를 결정하기 위해 상기 외부 동기화기(122)와 통신하고, 그리고 상기 제1 프로세서 유닛(132)과 상기 제2 프로세서 유닛(134) 사이의 원하지 않는 양의 스큐(136)가 존재할 때 상기 원하지 않는 양의 스큐(136)가 감소되게, 필요한 결과를 생성하지 않고 조치(138)를 수행할 것을 상기 제1 프로세서 유닛(132)에 선택적으로 지시하도록 구성되고,

상기 조치(138)는 상기 제1 프로세서 유닛(132)으로부터 사전 스케줄링된 유휴 시간을 제거하고,

상기 제1 프로세서 유닛(132) 및 상기 제2 프로세서 유닛(134)에 의해 생성된 대응하는 메시지들에 대해 무결성 검사들이 수행되는 고 무결성 모드(106)를 위해 상기 제1 프로세서 유닛(132)과 상기 제2 프로세서 유닛(134)이 서로 연관되는,

프로세서 동기화 시스템(127).

청구항 2

제1 항에 있어서,

상기 외부 동기화기(122)는 상기 제1 프로세서 유닛(132)과 상기 제2 프로세서 유닛(134) 사이의 상기 원하지 않는 양의 스큐(136)에 관한 정보를 저장하는,

프로세서 동기화 시스템(127).

청구항 3

제1 항 또는 제2 항에 있어서,

상기 외부 동기화기(122)는 상기 원하지 않는 양의 스큐(136)에 관한 정보를 레지스터들(156)에 저장하고,

상기 프로세서 동기화 시스템(127)과 통신하는 상기 제1 프로세서 유닛(132)과 상기 제2 프로세서 유닛(134) 사이에 상기 원하지 않는 양의 스큐(136)가 존재하는지 여부를 결정하기 위해 상기 외부 동기화기(122)와 통신할 때, 상기 내부 동기화기(130)는 상기 외부 동기화기(122)의 상기 제1 프로세서 유닛(132)에 대해 제1 레지스터(158)에 기록하고 상기 외부 동기화기(122)의 상기 제2 프로세서 유닛(134)에 대해 제2 레지스터(160)를 판독하도록 구성되며, 상기 제1 프로세서 유닛(132)과 상기 제2 프로세서 유닛(134)이 명령들을 동시에 처리하도록 정렬되는 시점을 결정하기 위해 상기 제1 레지스터(158)와 상기 제2 레지스터(160)의 값들을 비교하는,

프로세서 동기화 시스템(127).

청구항 4

제1 항 또는 제2 항에 있어서,

상기 외부 동기화기(122)는 상기 제1 프로세서 유닛(132)에 대한 제1 클록(162) 및 상기 제2 프로세서 유닛(134)에 대한 제2 클록(164)을 포함하며,

상기 제1 클록(162) 및 상기 제2 클록(164)은 명령들을 동시에 처리하도록 상기 제1 프로세서 유닛(132)과 상기 제2 프로세서 유닛(134)을 정렬하는 데 사용되는,

프로세서 동기화 시스템(127).

청구항 5

제1 항 또는 제2 항에 있어서,

상기 제1 프로세서 유닛(132)은 선두(leading) 프로세서 유닛이고;

상기 제2 프로세서 유닛(134)은 뒤처진(lagging) 프로세서 유닛이며;

차이 레지스터가 상기 선두 프로세서 유닛으로 0 값을 반환하고; 그리고

상기 차이 레지스터는 상기 뒤처진 프로세서 유닛으로 상기 원하지 않는 양의 스큐(136)의 값을 반환하는,

프로세서 동기화 시스템(127).

청구항 6

제1 항 또는 제2 항에 있어서,

상기 사전 스케줄링된 유휴 시간은, 상기 제1 프로세서 유닛(132) 내의 상기 내부 동기화기(130)가 상기 제1 프로세서 유닛(132)이 상기 제2 프로세서 유닛(134)보다 더 느린 것을 검출하는 것에 응답하여 제거되며, 상기 제1 프로세서 유닛(132)과 상기 제2 프로세서 유닛(134) 사이의 상기 원하지 않는 양의 스큐(136)를 감소시키는,

프로세서 동기화 시스템(127).

청구항 7

제6 항에 있어서,

상기 내부 동기화기(130)는 상기 사전 스케줄링된 유휴 시간을 제거한 후 상기 제1 프로세서 유닛(132)과 상기 제2 프로세서 유닛(134) 사이에 상기 원하지 않는 양의 스큐(136)가 여전히 존재한다면, 추가 사전 스케줄링된 유휴 시간을 제거하도록 구성되는,

프로세서 동기화 시스템(127).

청구항 8

프로세서 유닛들을 동기화하기 위한 방법으로서,

동기화 시스템과 통신하는 제1 프로세서 유닛(132)과 제2 프로세서 유닛(134) 사이에 원하지 않는 양의 스큐(136)가 존재하는지 여부를 결정하기 위해 외부 동기화기(122)와 통신하는 단계; 및

상기 제1 프로세서 유닛(132)에 상기 제1 프로세서 유닛(132)과 상기 제2 프로세서 유닛(134) 사이의 상기 원하지 않는 양의 스큐(136)가 존재할 때 상기 원하지 않는 양의 스큐(136)가 감소되도록, 필요한 결과를 생성하지 않고 조치(138)를 수행할 것을 상기 제1 프로세서 유닛(132)에 선택적으로 지시하는 단계를 포함하고,

상기 조치(138)는 상기 제1 프로세서 유닛(132)으로부터 사전 스케줄링된 유휴 시간을 제거하고,

상기 제1 프로세서 유닛(132) 및 상기 제2 프로세서 유닛(134)에 의해 생성된 대응하는 메시지들에 대해 무결성 검사들이 수행되는 고 무결성 모드(106)를 위해 상기 제1 프로세서 유닛(132)과 상기 제2 프로세서 유닛(134)이 서로 연관되는,

프로세서 유닛들을 동기화하기 위한 방법.

청구항 9

제8 항에 있어서,

상기 제1 프로세서 유닛과 상기 제2 프로세서 유닛 사이의 상기 원하지 않는 양의 스큐에 관한 정보를 상기 외부 동기화기에 저장하는 단계를 더 포함하는,

프로세서 유닛들을 동기화하기 위한 방법.

청구항 10

제8 항 또는 제9 항에 있어서,

상기 외부 동기화기는 상기 원하지 않는 양의 스큐에 관한 정보를 레지스터들에 저장하고,

상기 동기화 시스템과 통신하는 상기 제1 프로세서 유닛과 상기 제2 프로세서 유닛 사이에 상기 원하지 않는 양의 스큐가 존재하는지 여부를 결정하기 위해 상기 외부 동기화기와 통신하는 단계는:

상기 동기화 시스템의 상기 제1 프로세서 유닛에 대해 제1 레지스터에 기록하는 단계;

상기 동기화 시스템의 상기 제2 프로세서 유닛에 대해 제2 레지스터를 판독하는 단계; 및

상기 제1 프로세서 유닛과 상기 제2 프로세서 유닛이 명령들을 동시에 처리하도록 정렬되는 시점을 결정하기 위해 상기 제1 레지스터와 상기 제2 레지스터의 값들을 비교하는 단계를 포함하는,

프로세서 유닛들을 동기화하기 위한 방법.

발명의 설명

기 술 분 야

[0001] 본 개시내용은 일반적으로 개선된 컴퓨터 시스템에 관한 것으로, 특히 컴퓨터 시스템 내의 프로세서 유닛들에 대한 데이터 흐름을 관리하기 위한 방법 및 장치에 관한 것이다.

배 경 기 술

[0002] 항공기는 서로 다른 항공기 시스템들의 신뢰성이나 무결성, 또는 신뢰성과 무결성을 높이기 위해 많은 중복 컴포넌트들을 갖는다. 예를 들어, 시스템의 신뢰성을 높이거나, 시스템의 무결성을 높이거나, 시스템의 신뢰성과 무결성을 높이기 위해 컴포넌트들이 복제될 수 있다. 이러한 중복성은 시스템 성능을 향상시키기 위해 복제 시스템의 형태일 수 있다. 예를 들어, 항공기 내의 컴퓨터 시스템들은 하드웨어 중복뿐만 아니라 정보 중복을 포함할 수 있다. 소프트웨어 문제들, 방사선, 또는 다른 원인들의 소스들과 같은 다양한 소스들을 통해 정보의 부정확성이 발생할 수 있다.

[0003] 항공기 내의 컴퓨터들 상에서 실행되는 에러 검출 및 정정 프로세스들을 사용하여 정보 무결성이 얻어질 수 있다. 무결성은 또한 중복성을 포함할 수 있다. 중복성에 의해, 동일한 동작을 여러 번 수행하거나 서로 다른 컴퓨터들 또는 프로세서 유닛들 상의 동일한 애플리케이션을 실행하여 데이터의 다수의 사본들을 얻는 것이 사용될 수 있다. 데이터의 이러한 사본들이 검사되어, 사본들이 서로 일치하는지 여부를 결정할 수 있다.

[0004] 일부 해결책들은 수신기들에 의존하여 데이터의 중복 사본들을 비교하는 한편, 다른 해결책들은 데이터의 소스에서 무결성을 생성한다. 후자의 경우, 현재 해결책들은 단일 코어 프로세서 유닛들을 2개 또는 그보다 많은 병렬 레인(lane)들에 배치한다. 이러한 프로세서 유닛들에 의해 생성된 메시지들에 대해 데이터 버스 락스텝(lockstep) 검사가 수행된다. 이러한 타입들의 프로세스들은 외부 칩들, 프로세서 아키텍처의 수정들, 또는 이들의 어떤 결합을 사용하여 수행된다. 이러한 타입들의 해결책들은 특정 타입들의 프로세서들에 맞춰진다. 그 결과, 다른 타입들의 프로세서들이 이용될 때, 이러한 해결책들은 메모리 버스 아키텍처, 이용 가능한 비교 포인트들, 또는 다른 특성들과 같은 그러한 프로세서들의 특성들과 잘 작동하지 않을 수 있다.

[0005] 따라서 앞서 논의한 문제들뿐만 아니라, 다른 가능한 문제들 중 적어도 일부를 고려하는 방법 및 장치를 갖는 것이 바람직할 것이다. 예를 들어, 프로세서 유닛들을 사용하여 데이터를 처리하기 위한 원하는 효율을 얻으면서, 단일 코어를 갖는 프로세서 유닛들을 사용하거나 다수의 코어들을 사용할 때 데이터 무결성을 달성하는 것의 기술적 문제점을 극복하는 방법 및 장치를 갖는 것이 바람직할 것이다.

발명의 내용

[0006] 본 개시내용의 일 실시예는 제1 프로세서 유닛 상에서 실행되는 내부 동기화기를 포함하는 프로세서 동기화 시스템을 제공한다. 내부 동기화기는 외부 동기화기와 통신하는 제1 프로세서 유닛과 제2 프로세서 유닛 사이에 원하지 않는 양의 스큐(skew)가 존재하는지 여부를 결정하기 위해 외부 동기화기와 통신하도록 구성된다. 내부 동기화기는 제1 프로세서 유닛과 제2 프로세서 유닛 사이의 원하지 않는 양의 스큐가 존재할 때 원하지 않는 양의 스큐가 감소되도록, 필요한 결과를 생성하지 않고 조치를 수행할 것을 제1 프로세서 유닛에 선택적으로 지시한다. 제1 프로세서 유닛 및 제2 프로세서 유닛에 의해 생성된 대응하는 메시지들에 대해 무결성 검사들이 수

행되는 고 무결성 모드를 위해 제1 프로세서 유닛과 제2 프로세서 유닛이 서로 연관된다.

[0007] 본 개시내용의 다른 실시예는 프로세서 유닛들을 동기화하기 위한 방법을 제공한다. 동기화 시스템과 통신하는 제1 프로세서 유닛과 제2 프로세서 유닛 사이에 원하지 않는 양의 스큐가 존재하는지 여부를 결정하기 위해 외부 동기화기와 통신이 이루어진다. 제1 프로세서 유닛은 제1 프로세서 유닛에 제1 프로세서 유닛과 제2 프로세서 유닛 사이의 원하지 않는 양의 스큐가 존재할 때 원하지 않는 양의 스큐가 감소되게, 필요한 결과를 생성하지 않고 조치를 수행하도록 선택적으로 지시를 받는다. 제1 프로세서 유닛 및 제2 프로세서 유닛에 의해 생성된 대응하는 메시지들에 대해 무결성 검사들이 수행되는 고 무결성 모드를 위해 제1 프로세서 유닛과 제2 프로세서 유닛이 서로 연관된다.

[0008] 특징들 및 기능들은 본 개시내용의 다양한 실시예들에서는 독립적으로 달성될 수 있고 또는 또 다른 실시예들에서는 결합될 수 있는데, 여기서 추가 세부사항들은 다음 설명 및 도면들을 참조로 확인될 수 있다.

도면의 간단한 설명

[0009] 예시적인 실시예들의 특성으로 여겨지는 신규한 특징들은 첨부된 청구항들에서 제시된다. 그러나 예시적인 실시예들뿐만 아니라 이들의 선호되는 사용 모드, 추가 목적들 및 특징들 또한, 첨부 도면들과 함께 일독시 본 개시내용의 예시적인 실시예의 아래의 상세한 설명에 대한 참조에 의해 가장 잘 이해될 것이다.

도 1은 예시적인 실시예에 따른 데이터 처리 환경의 블록도의 예시이다.

도 2는 예시적인 실시예에 따른 혼합된 무결성 데이터 처리 시스템의 예시이다.

도 3은 예시적인 실시예에 따라 복수의 프로세서들에 대한 데이터 전송을 관리하기 위한 프로세스의 흐름도의 예시이다.

도 4는 예시적인 실시예에 따라 프로세서 유닛들을 동기화하기 위한 프로세스의 흐름도의 예시이다.

도 5는 예시적인 실시예에 따라 프로세서 유닛들을 동기화하기 위한 프로세스의 흐름도의 예시이다.

도 6은 예시적인 실시예에 따라 프로세서 유닛들을 동기화하기 위한 프로세스의 흐름도의 예시이다.

도 7은 예시적인 실시예에 따라 프로세서 유닛들로부터의 메시지들의 전송을 동기화하기 위한 프로세스의 흐름도의 예시이다.

도 8은 예시적인 실시예에 따라 메시지들을 전송하기 위한 프로세스의 흐름도의 예시이다.

도 9는 예시적인 실시예에 따라 메시지들을 전송하기 위한 프로세스의 흐름도의 예시이다.

도 10은 예시적인 실시예에 따라 메시지들을 수신하기 위한 프로세스의 흐름도의 예시이다.

도 11은 예시적인 실시예에 따라 메시지들을 수신하기 위한 프로세스의 흐름도의 예시이다.

도 12는 예시적인 실시예에 따라 메시지들을 수신하기 위한 프로세스의 흐름도의 예시이다.

도 13은 예시적인 실시예에 따라 메시지들을 수신하기 위한 프로세스의 흐름도의 예시이다.

도 14는 예시적인 실시예에 따른 데이터 처리 시스템의 블록도의 예시이다.

도 15는 예시적인 실시예에 따른 항공기 제조 및 서비스 방법의 블록도의 예시이다.

도 16은 예시적인 실시예가 구현될 수 있는 항공기의 블록도의 예시이다.

발명을 실시하기 위한 구체적인 내용

[0010] 예시적인 실시예들은 하나 또는 그보다 많은 서로 다른 고려사항들을 인식하여 고려한다. 예를 들어, 예시적인 실시예들은 하나 또는 그보다 많은 중복 애플리케이션들을 실행하는 임의의 쌍 또는 그룹의 프로세서 유닛들을 사용할 때 데이터 출력의 무결성을 높이는 것이 바람직함을 인식하여 고려한다. 예시적인 실시예들은 표준 무결성 동작 모드로 지칭되는 통상의 데이터 처리와 비교하여 데이터의 무결성을 높이기 위한 동작 모드는 고 무결성 동작 모드임을 인식하여 고려한다.

[0011] 따라서 예시적인 실시예들은 데이터를 관리하기 위한 방법, 장치 및 시스템을 제공한다. 한 예시적인 예에서, 메시지 관리 시스템은 무결성 관리기를 포함한다. 무결성 관리기는 혼합된 무결성 모드들을 가지며, 혼합된 무

결성 모드들 중 선택된 모드를 기초로 프로세서 유닛들과 외부 노드 간의 메시지들의 교환을 관리하도록 구성된다. 무결성 관리기는 프로세서들 및 외부 노드와 통신하는 하드웨어에 위치되며, 고 무결성 모드에서 프로세서 유닛들로부터의 중복 계산된 출력들이 일치함을 검사한다.

[0012] 다른 예시적인 예에서, 프로세서 동기화 시스템은 제1 프로세서 유닛 상에서 실행되는 내부 동기화기를 포함한다. 내부 동기화기는 외부 동기화기와 통신하는 제1 프로세서 유닛과 제2 프로세서 유닛 사이에 원하지 않는 양의 스큐(skew)가 존재하는지 여부를 결정하기 위해 외부 동기화기와 통신하도록 구성된다. 내부 동기화기는 원하지 않는 스큐가 존재할 때 제1 프로세서 유닛과 제2 프로세서 유닛 사이의 스큐의 양이 감소되게, 필요한 결과를 생성하지 않고 조치를 수행할 것을 제1 프로세서 유닛에 선택적으로 지시하도록 구성된다.

[0013] 이제 도면들을 참조하면, 그리고 특히 도 1을 참조하면, 예시적인 실시예에 따라 데이터 처리 환경의 블록도의 예시가 도시된다. 데이터 처리 환경(100)은 혼합된 무결성 모드들(104)에서 프로세서 유닛들(102)이 관리될 수 있는 환경이다. 이 예에서, 프로세서 유닛들(102)은 멀티 코어 프로세서 유닛, 단일 코어 프로세서 유닛, 동종 멀티 코어 프로세서 유닛, 이종 멀티 코어 프로세서 유닛, 그래픽 프로세서 유닛, 범용 프로세서 유닛, 또는 다른 어떤 적당한 타입의 프로세서 유닛 중 적어도 하나로부터 선택된다.

[0014] 예시적인 예에서, 프로세서 유닛들(102)은 동일한 타입이거나 서로 다른 타입일 수 있다. 프로세서 유닛들(102)이 동일한 타입일 때, 그러한 프로세서 유닛들은 동일한 명령 세트, 디자인, 모델, 부품 번호, 또는 프로세서 유닛들(102)에 대한 타입을 정의하기 위한 다른 파라미터들 중 적어도 하나를 가질 수 있다.

[0015] 이 예시적인 예에서, 혼합된 무결성 모드들(104)은 고 무결성 모드(106) 및 표준 무결성 모드(108)를 포함한다. 프로세서 유닛들(102)에 대한 레인들 간의 메시지 비교 또는 메시지 동기화 중 적어도 하나가 고 무결성 모드(106)에서 수행된다. 예시적인 예에서, 레인은 처리 유닛이다.

[0016] 고 무결성 모드(106)에 있을 때, 무결성 관리기(120)는 프로세서 유닛들(102) 중 다수의 프로세서 유닛들로부터의 중복 계산된 출력들이 일치함을 검사한다. 예시적인 예에서, 일치는 다수의 프로세서 유닛들로부터 전송된 데이터가 동일함을 의미한다. 무결성 관리기(120)는 다수의 프로세서 유닛들로부터 확실히 동일한 데이터가 전송되고 있는지를 검사한다.

[0017] 예를 들어, 고 무결성 모드(106)로 동작하는 2개의 프로세서 유닛들이 동일한 애플리케이션을 실행할 수 있고, 동일한 기능 또는 데이터의 처리를 수행하도록 명령을 받는다. 고 무결성 모드(106)에서, 무결성 관리기(120)는 프로세서 유닛들(102) 중 2개의 프로세서 유닛들에 의해 생성된 출력들이 확실히 동일한지를 검사한다.

[0018] 도시된 바와 같이, 프로세서 유닛들(102)은 컴퓨터 시스템(110)에 위치된다. 도시된 바와 같이, 컴퓨터 시스템(110)은 물리적 하드웨어 시스템이고 하나 또는 그보다 많은 데이터 처리 시스템들을 포함한다. 하나보다 많은 데이터 처리 시스템이 존재할 때, 프로세서 유닛들은 단일 데이터 처리 시스템에 또는 데이터 처리 시스템들 중 하나보다 많은 데이터 처리 시스템에 위치될 수 있다. 또한, 하나보다 많은 데이터 처리 시스템이 존재할 때, 그러한 데이터 처리 시스템들은 통신 매체를 사용하여 서로 통신한다. 통신 매체는 네트워크일 수 있다. 데이터 처리 시스템들은 컴퓨터, 서버 컴퓨터, 태블릿, 또는 다른 어떤 적절한 데이터 처리 시스템 중 적어도 하나로부터 선택될 수 있다.

[0019] 이 예시적인 예에서, 데이터 관리기(112)는 프로세서 유닛들(102)에 대한 데이터(114)의 처리 또는 데이터(114)의 교환 중 적어도 하나를 관리한다. 본 명세서에서 사용되는 바와 같이, 항목들의 리스트에 사용되는 경우에 "~ 중 적어도 하나"라는 문구는, 열거된 항목들 중 하나 또는 그보다 많은 항목의 서로 다른 결합들이 사용될 수 있으며 리스트 내의 각각의 항목 중 단 하나만이 필요할 수 있음을 의미한다. 즉, "~ 중 적어도 하나"는 리스트로부터 항목들의 임의의 결합 및 임의의 수의 항목들이 사용될 수 있지만, 리스트 내의 항목들 전부가 요구되는 것은 아님을 의미한다. 항목은 특정 객체, 물건 또는 카테고리일 수 있다.

[0020] 예를 들어, "항목 A, 항목 B 또는 항목 C 중 적어도 하나"는 제한 없이, 항목 A, 항목 A와 항목 B, 또는 항목 B를 포함할 수 있다. 이 예는 또한 항목 A, 항목 B 및 항목 C 또는 항목 B와 항목 C를 포함할 수 있다. 물론, 이러한 항목들의 임의의 결합들이 존재할 수 있다. 일부 예시적인 예들에서, "~ 중 적어도 하나"는 예를 들어, 제한 없이, 2개의 항목 A; 1개의 항목 B; 그리고 10개의 항목 C; 4개의 항목 B와 7개의 항목 C; 또는 다른 적당한 결합들일 수 있다.

[0021] 도시된 바와 같이, 데이터 관리기(112)는 컴퓨터 시스템(110)에 위치된다. 데이터 관리기(112)는 컴퓨터 시스템(110) 내의 프로세서 유닛들(102)과 동일한 또는 다른 데이터 처리 시스템 상에 위치될 수 있다.

[0022] 도시된 바와 같이, 데이터 관리기(112)는 프로세서 유닛들(102)에 의한 데이터(114)의 처리를 관리한다. 예를 들어, 프로세서 유닛들(102)이 고 무결성 모드(106)로 동작할 때, 데이터 관리기(112)는 프로세서 유닛들(102)의 타이밍을 정렬하여 데이터(114)를 처리할 수 있다. 이러한 타이밍 정렬은 프로세서 유닛들(102)에 의한 데이터(114)의 처리가 동기화된 방식으로 데이터(114)를 동시에 처리하게 할 수 있다. 즉, 프로세서 유닛들(102)은 데이터(114)를 처리하기 위해 동일한 애플리케이션을 실행하여 실질적으로 동시에 결과들을 생성한다. 이러한 결과들은 프로세서 유닛들(102) 중 2개 이상에 의해 생성될 수 있는데, 데이터(114)의 처리로부터 생성된 결과들의 무결성을 결정하기 위해 이러한 결과들이 비교될 수 있다.

[0023] 도시된 바와 같이, 프로세서 유닛들(102) 중 2개 이상은 고 무결성 모드(106)로 동작할 수 있는 한편, 프로세서 유닛들(102) 중 다른 프로세서 유닛들은 표준 무결성 모드(108)로 동작할 수 있다. 즉, 데이터 관리기(112)는 혼합된 무결성 모드들(104)의 서로 다른 모드들로 동시에 처리 및 데이터를 관리할 수 있다. 혼합된 무결성 모드들(104)은, 프로세서 유닛들(102) 중 일부는 고 무결성 모드(106)로 동작할 수 있는 한편, 프로세서 유닛들(102) 중 다른 프로세서 유닛들은 표준 무결성 모드(108)로 동작할 수 있음을 의미한다. 혼합된 무결성 모드들(104)은 또한, 모든 프로세서 유닛들(102)이 고 무결성 모드(106)로 또는 표준 무결성 모드(108)로 동작할 수 있음을 의미한다.

[0024] 다른 예시적인 예들에서, 데이터 관리기(112)는 외부 노드(116)와의 데이터(114)의 교환을 관리할 수 있다. 고 무결성 모드(106)에서 동작할 때, 데이터 관리기(112)는 데이터(114)를 포함하는 전송 메시지들(118)의 수신 또는 송신 중 적어도 하나의 타이밍을 정렬할 수 있다. 예를 들어, 데이터 관리기(112)는 프로세서 유닛들(102) 중 2개 이상이 전송 메시지들(118)과 동일한 메시지들을 실질적으로 동시에, 실질적으로 동일한 순서로, 또는 이들의 어떤 결합으로 수신하도록 프로세서 유닛들(102) 중 이러한 프로세서 유닛들에 의한 메시지들의 수신을 관리할 수 있다.

[0025] 이러한 그리고 다른 기능들은 프로세서 유닛들(102)에 대한 데이터 관리기(112) 중 적어도 하나 내에서 다수의 서로 다른 기능들을 사용하여 수행될 수 있다. 예를 들어, 기능들은 데이터 관리기(112) 내의 무결성 관리기(120), 데이터 관리기(112) 내의 외부 동기화기(122), 또는 프로세서 유닛들(102)의 내부 동기화기들(124) 중 하나 또는 그보다 많은 내부 동기화기 중 적어도 하나를 사용하여 수행될 수 있다.

[0026] 한 예시적인 예에서, 메시지 관리 시스템(125)은 무결성 관리기(120)를 사용하여 제공될 수 있다. 이 예시적인 예에서, 무결성 관리기(120)는 혼합된 무결성 모드들(104)을 갖는다. 도시된 바와 같이, 무결성 관리기(120)는 혼합된 무결성 모드들(104) 중 선택된 모드(128)를 기초로 프로세서 유닛들(102)과 외부 노드(116) 간의 전송 메시지들(118)의 교환을 관리하도록 구성된다. 이 예시적인 예에서, 무결성 관리기(120)는 프로세서 유닛들(102) 및 외부 노드(116)와 통신하는 하드웨어에 위치된다.

[0027] 도시된 바와 같이, 외부 노드(116)는 컴퓨터 시스템(110) 밖에 위치된다. 다른 예들에서는, 외부 노드(116)가 컴퓨터 시스템(110) 내에 위치될 수 있다. 외부 노드(116)는 다수의 서로 다른 형태들을 취할 수 있다. 예를 들어, 외부 노드(116)는 프로세서 유닛들(102)과 통신하는 외부 프로세서, 내장형 컴퓨터, 데스크톱 컴퓨터, 네트워크 디바이스, 네트워크 스위치, 및 다른 어떤 적당한 타입의 외부 디바이스를 포함하는 그룹으로부터 선택될 수 있다.

[0028] 한 예시적인 예에서, 프로세서 유닛들(102)의 제1 부분은 고 무결성으로 동작하고, 프로세서 유닛들(102)의 제2 부분은 표준 무결성으로 동작한다. 무결성 관리기(120)는 선택된 모드(128)로서 고 무결성 모드(106)에서는 프로세서 유닛들(102)의 제1 부분에 대한 전송 메시지들(118)을 관리하고, 선택된 모드(128)로서 표준 무결성 모드(108)에서는 프로세서 유닛들(102)의 제2 부분에 대한 전송 메시지들(118)을 관리하도록 구성된다. 전송 메시지들(118)은 프로세서 유닛들(102)에 전송되거나 프로세서 유닛들(102)로부터 전송될 수 있는 메시지이다.

[0029] 도시된 바와 같이, 무결성 관리기(120)는 고 무결성 모드(106)와 표준 무결성 모드(108)에서 동시에 동작하는 프로세서 유닛들(102)을 관리하도록 구성된다. 이러한 타입의 동작은 프로세서 유닛들(102)이 고 무결성 모드(106) 또는 표준 무결성 모드(108) 중 적어도 하나에서 동작하는 혼합된 무결성 모드들(104)의 지원을 가능하게 한다. 즉, 프로세서 유닛들(102)은 이러한 모드들 중 어느 하나 또는 둘 다에서 동시에 동작할 수 있다.

[0030] 이 예시적인 예에서, 무결성 관리기(120)는 혼합된 무결성 모드들(104) 중 선택된 모드(128)가 고 무결성 모드(106)일 때, 프로세서 유닛들(102)에 대한 전송 메시지들(118)의 수신 또는 프로세서 유닛들(102)에 대한 전송 메시지들(118)의 송신 중 적어도 하나를 동기화하도록 구성된다.

[0031] 예를 들어, 무결성 관리기(120)는 제1 프로세서 유닛(132)에 대한 제1 대기열(142)에 외부 노드(116)로부터의

전송 메시지들(118)로 수신된 외부 메시지(140)를 배치하고, 제1 프로세서 유닛(132) 및 제2 프로세서 유닛(134)이 고 무결성을 사용하여 동작할 때, 제2 프로세서 유닛(134)에 대한 제2 대기열(146)에 외부 메시지(140)의 사본(144)을 배치하도록 구성된다. 무결성 관리기(120)는 병렬 레인 일관성이 존재하게 동일한 깊이로 제1 프로세서 유닛(132)이 제1 대기열(142)로부터 판독할 뿐만 아니라 제2 프로세서 유닛(134)이 제2 대기열(146)로부터 판독하도록 제1 대기열(142) 및 제2 대기열(146)을 제어한다. 예시적인 예에서들, 깊이는 대기열 내의 메시지들의 양이다.

[0032] 예시적인 예에서는, 대기열들의 관리뿐만 아니라, 무결성 관리기(120)는 제1 프로세서 유닛(132)으로부터 수신된 수신 메시지(150)를 제2 프로세서 유닛(134)으로부터 수신된 대응하는 메시지들(154) 내의 대응하는 메시지(152)와 비교한다. 수신 메시지(150)와 대응하는 메시지(152)가 일치하면, 무결성 관리기(120)는 수신 메시지(150)를 외부 노드(116)에 전송한다.

[0033] 무결성 관리기(120)에 의해 이루어진 비교는 다수의 상이한 방식들로 수행될 수 있다. 예를 들어, 비교는 비트별 비교 또는 순환 중복 검사 중 적어도 하나를 사용하여 이루어질 수 있다. 이 예에서는, 수신 메시지(150)와 대응하는 메시지(152) 사이에 일치가 존재하지 않는다면 또는 원하지 않는 양의 스큐가 존재함을 나타내는 선택된 기간의 시간 내에 대응하는 메시지(152)가 수신되지 않는다면, 무결성 관리기(120)는 수신 메시지(150)를 폐기한다. 이 예에서, 일치가 존재하면, 대응하는 메시지(152)가 폐기된다.

[0034] 또한, 프로세서 유닛들(102) 각각은 무결성 관리기(120)에 메시지들을 송신하기 전에 데이터 값들을 절단하여, 무결성 관리기(120)가 정확한 검사인 비트별 비교를 사용하여 메시지들을 비교할 때 값들의 정확도를 낮출 수 있다. 예를 들어, 메시지들 내의 값들이 64 비트라면, 절단이 사용되지 않는 경우에는 64 비트 전부가 검사된다. 어떤 경우들에는, 비트별 비교를 사용하여 데이터를 처리할 때 서로 다른 레인들에서 동작하는 프로세서 유닛들(102) 사이에서 원하지 않는 결과들이 발생할 수 있다. 그 결과, 프로세서 유닛들(102)은 하나 또는 그보다 많은 비트들을 절단하여 정확도 수준을 낮출 수 있다. 따라서 프로세서 유닛들(102)이 고 무결성 모드(106)일 때 프로세서 유닛들(102)이 비트들을 절단함으로써 가변적인 정확도 수준이 달성될 수 있다.

[0035] 다른 예시적인 예에서, 외부 동기화기(122)와 통신하는 내부 동기화기들(124)을 이용하는 프로세서 동기화 시스템(127)이 제공될 수 있다. 도시된 바와 같이, 내부 동기화기들(124) 중 내부 동기화기(130)가 프로세서 유닛들(102) 중 제1 프로세서 유닛(132) 상에서 실행된다. 내부 동기화기(130)는 외부 동기화기(122)와 통신하는 제1 프로세서 유닛(132)과 제2 프로세서 유닛(134) 사이에 원하지 않는 양의 스큐(136)가 존재하는지 여부를 결정하기 위해 데이터 관리기(112) 내의 외부 동기화기(122)와 통신하도록 구성된다. 예시적인 예에서, 스큐는 2개 또는 그보다 많은 프로세서 유닛들로부터 신호들이 수신되는 시간의 차이이다. 스큐는 클록 스큐, 및 프로세서 유닛들 간의 다른 요인들로 인해 유도될 수 있다.

[0036] 내부 동기화기(130)는, 원하지 않는 양의 스큐(136)가 존재할 때 제1 프로세서 유닛(132)과 제2 프로세서 유닛(134) 사이의 스큐(136)의 양이 감소되게, 필요한 결과를 생성하지 않고 조치(138)를 수행할 것을 제1 프로세서 유닛(132)에 선택적으로 지시하도록 구성된다.

[0037] 프로세서 유닛들(102)에 의한 처리시 타이밍을 정렬하기 위해 스큐(136)를 감소시키는 한 예시적인 예에서, 조치(138)는 제1 프로세서 유닛(132)으로부터 사전 스케줄링된 유휴 시간(166)을 제거하는데, 이는 제1 프로세서 유닛(132)과 제2 프로세서 유닛(134) 사이의 스큐(136)의 양을 감소시킨다. 뒤처진(lagging) 프로세서 유닛으로부터 사전 스케줄링된 유휴 시간을 제거하는 것은 해당 프로세서 유닛으로 하여금 명령들의 처리에 관련하여 시간을 건너뛰게 한다. 내부 동기화기(130)는 사전 스케줄링된 유휴 시간(166)을 제거한 후 제1 프로세서 유닛(132)과 제2 프로세서 유닛(134) 사이에 원하지 않는 양의 스큐(136)가 여전히 존재한다면, 추가 사전 스케줄링된 유휴 시간(168)을 제거하도록 구성된다.

[0038] 또 다른 예시적인 예에서, 제1 프로세서 유닛(132)은 선두(leading) 프로세서 유닛이고, 제2 프로세서 유닛(134)은 뒤처진 프로세서 유닛이며, 조치는 제1 프로세서 유닛(132)으로 하여금 유휴 시간을 추가하게 하여, 뒤처진 프로세서 유닛인 제2 프로세서 유닛(134)이 정렬된 타이밍을 가질 수 있게 한다. 이런 식으로, 동기 처리를 위해 2개의 프로세서 유닛들에 대한 타이밍이 정렬될 수 있다.

[0039] 도시된 바와 같이, 제1 프로세서 유닛(132)과 제2 프로세서 유닛(134)은, 제1 프로세서 유닛(132) 및 제2 프로세서 유닛(134)에 의해 생성된 전송 메시지들(118) 내의 대응하는 메시지들(154)에 대해 무결성 검사들이 수행되는 고 무결성 모드(106)를 위해 서로 연관된다.

[0040] 외부 동기화기(122)는 제1 프로세서 유닛(132)과 제2 프로세서 유닛(134) 사이의 스큐(136)의 양에 관한 정보를

저장할 수 있다. 이 정보는 외부 동기화기(122)의 레지스터들(156) 또는 다른 저장 메커니즘들에 저장될 수 있다. 이 정보는 예를 들어, 레지스터들(156)에 설정된 비트들의 그룹 또는 플래그일 수 있다. 본 명세서에서 사용되는 바와 같이, 항목들과 관련하여 사용될 때 "~의 그룹"은 하나 또는 그보다 많은 항목들을 의미한다. 예를 들어, "비트들의 그룹"은 하나 또는 그보다 많은 비트들이다.

[0041] 외부 동기화기(122)는 스큐(136)의 양에 관한 정보를 레지스터들(156)에 저장한다. 내부 동기화기(130)는 외부 동기화기(122)의 제1 프로세서 유닛(132)에 대해 제1 레지스터(158)에 기록하고 외부 동기화기(122)의 제2 프로세서 유닛(134)에 대해 제2 레지스터(160)를 판독하도록 구성되며, 제1 레지스터(158)와 제2 레지스터(160)의 값들을 비교하여, 제1 프로세서 유닛(132)과 제2 프로세서 유닛(134)이 명령들을 동시에 처리하도록 타이밍 정렬되는 시점을 결정한다.

[0042] 다른 예시적인 예에서, 외부 동기화기(122)는 제1 프로세서 유닛(132)에 대한 제1 클록(162) 및 제2 프로세서 유닛(134)에 대한 제2 클록(164)을 포함한다. 제1 클록(162) 및 제2 클록(164)은 명령들을 동시에 처리하도록 제1 프로세서 유닛(132) 및 제2 프로세서 유닛(134)에 대한 타이밍을 정렬하는 데 사용된다.

[0043] 무결성 관리기(120), 외부 동기화기(122) 또는 내부 동기화기들(124) 중 적어도 하나는 소프트웨어, 하드웨어, 펌웨어, 또는 이들의 결합으로 구현될 수 있다. 소프트웨어가 사용될 때, 무결성 관리기(120), 외부 동기화기(122) 또는 내부 동기화기들(124) 중 적어도 하나에 의해 수행되는 동작들은 프로세서 유닛과 같은 하드웨어 상에서 실행되도록 구성된 프로그램 코드로 구현될 수 있다. 펌웨어가 사용될 때, 무결성 관리기(120), 외부 동기화기(122) 또는 내부 동기화기들(124) 중 적어도 하나에 의해 수행되는 동작들은 프로그램 코드 및 데이터로 구현되며 영구 메모리에 저장되어 프로세서 유닛 상에서 실행될 수 있다. 하드웨어가 이용될 때, 하드웨어는 무결성 관리기(120), 외부 동기화기(122) 또는 내부 동기화기들(124) 중 적어도 하나에서 동작들을 수행하도록 작동하는 회로들을 포함할 수 있다.

[0044] 예시적인 예들에서, 하드웨어는 회로 시스템, 집적 회로, 주문형 집적 회로(ASIC: application-specific integrated circuit), 프로그래밍 가능 로직 디바이스, 또는 다수의 동작들을 수행하도록 구성된 다른 어떤 적당한 타입의 하드웨어 중 적어도 하나로부터 선택된 형태를 취할 수 있다. 프로그래밍 가능 로직 디바이스에 의해, 디바이스는 다수의 동작들을 수행하도록 구성될 수 있다. 디바이스는 추후에 재구성될 수 있고 또는 다수의 동작들을 수행하도록 영구적으로 구성될 수 있다. 프로그래밍 가능 로직 디바이스들은 예를 들어, 프로그래밍 가능 로직 어레이, 프로그래밍 가능 어레이 로직, 필드 프로그래밍 가능 로직 어레이, 필드 프로그래밍 가능 게이트 어레이, 그리고 다른 적당한 하드웨어 디바이스들을 포함한다. 추가로, 프로세스들은 무기 컴포넌트들과 통합되는 유기 컴포넌트들로 구현될 수 있고, 인간을 배제한 유기 컴포넌트들로 전부 구성될 수 있다. 예를 들어, 프로세스들은 유기 반도체들의 회로들로서 구현될 수 있다.

[0045] 예를 들어, 무결성 관리기(120)는 하드웨어에서 단일 로직 디바이스로서 또는 복수의 서로 다른 타입들의 로직 디바이스들로서 구현될 수 있다. 서로 다른 타입들의 로직 디바이스들은 공통 모드 장애들과 같은 잠재적 문제들을 감소시키도록 선택될 수 있다. 예를 들어, 제1 로직 디바이스는 필드 프로그래밍 가능 게이트 어레이일 수 있는 한편, 제2 로직 디바이스는 주문형 집적 회로(ASIC)일 수 있다. 또한, 하나보다 많은 로직 디바이스가 사용될 때, 각각의 로직 디바이스는 특정 프로세서 유닛에 대해 메시지들이 처리되게 하는 레인을 갖는다. 로직 디바이스들이 시간 정렬될 수 있게 로직 디바이스들은 접속들을 가질 수 있다.

[0046] 예시적인 일례에서는, 프로세서 유닛들을 사용하여 데이터를 처리하기 위한 원하는 중복 효율을 얻는 것의 기술적 문제점을 극복하는 하나 또는 그보다 많은 기술적 해결책들이 존재한다. 그 결과, 하나 또는 그보다 많은 기술적 해결책들은 프로세서 유닛들에 의해 데이터를 처리하는 효율을 높이는 기술적 효과를 제공할 수 있다. 예를 들어, 하나 또는 그보다 많은 기술적 해결책들은 고 무결성 모드로 동작하는 프로세서 유닛들에 의한 메시지들의 처리 또는 데이터의 처리의 타이밍 정렬 중 적어도 하나를 가능하게 할 수 있다.

[0047] 그 결과, 컴퓨터 시스템(110)은 컴퓨터 시스템(110) 내의 데이터 관리기(112) 또는 내부 동기화기들(124) 중 적어도 하나가 데이터(114)의 처리 중에 혼합된 무결성 모드들(104)이 존재할 수 있게 하는 방식으로 데이터(114)의 처리를 관리할 수 있게 하는 특수 목적 컴퓨터 시스템으로서 동작한다. 특히, 데이터 관리기(112) 또는 내부 동기화기들(124) 중 적어도 하나는 컴퓨터 시스템(110)을 데이터 관리기(112)나 내부 동기화기들(124), 또는 이들 모두를 갖지 않는 현재 이용 가능한 일반 컴퓨터 시스템들과 비교되는 특수 목적 컴퓨터 시스템으로 변환한다.

[0048] 도 1의 데이터 처리 환경의 예시는 예시적인 실시예가 구현될 수 있는 방식에 대한 물리적 또는 구성적 제한들

을 의미하려는 것은 아니다. 예시된 것들에 추가로 또는 그 대신 다른 컴포넌트들이 사용될 수 있다. 일부 컴포넌트들은 불필요할 수 있다. 또한, 블록들은 일부 기능 컴포넌트들을 예시하기 위해 제시된다. 이러한 블록들 중 하나 또는 그보다 많은 블록은 예시적인 실시예로 구현될 때, 결합되거나, 분할되거나, 또는 결합되어 서로 다른 블록들로 분할될 수 있다.

[0049] 예를 들어, 데이터 처리 환경(100)은 단지 외부 노드(116)와 함께 도시된다. 다른 예시적인 예에서는, 외부 노드(116)에 추가하여 또는 그 대신에 하나 또는 그보다 많은 외부 노드들이 존재한다.

[0050] 예시적인 예는 항공기에 대해 설명되지만, 다른 예시적인 예들은 다른 용도들에 적용될 수 있다. 예를 들어, 예시적인 예는 의료 영상, 회계, 일기 예보, 또는 다른 적절한 용도들로의 사용을 위해 구현될 수 있다.

[0051] 또 다른 예로서, 프로세서 유닛들(102)에서 2개의 프로세서 유닛들이 설명되었다. 프로세서 유닛들(102) 중 하나 또는 그보다 많은 프로세서 유닛이 제1 프로세서 유닛(132) 및 제2 프로세서 유닛(134)에 추가하여 또는 이들 대신 존재할 수 있다.

[0052] 예를 들어, 제3 프로세서 유닛이 프로세서 유닛들(102)에 존재할 수 있다. 내부 동기화기(130)는, 외부 동기화기와 통신하는 제1 프로세서 유닛(132), 제2 프로세서 유닛(134) 그리고 제3 프로세서 유닛 사이에 원하지 않는 양의 스큐(136)가 존재하는지 여부를 결정하기 위해 외부 동기화기(122)와 통신하도록, 그리고 원하지 않는 양의 스큐(136)가 존재할 때 제1 프로세서 유닛(132)과 제2 프로세서 유닛(134) 그리고 제3 프로세서 유닛 사이의 스큐(136)의 양이 감소되게, 필요한 결과를 생성하지 않고 조치(138)를 수행할 것을 제1 프로세서 유닛(132)에 선택적으로 지시하도록 구성된다.

[0053] 예를 들어, 혼합된 무결성 모드들(104)에 대해 2개의 무결성 모드들이 도시된다. 다른 예시적인 예들에서는, 다른 수들의 무결성 모드들이 존재할 수 있다. 예를 들어, 3개의 무결성 모드들, 5개의 무결성 모드들, 또는 다른 어떤 수의 무결성 모드들이 이용될 수 있다. 이러한 추가 무결성 모드들은 예를 들어, 서로 다른 수들의 비교들을 사용한다. 예를 들어, 3개의 무결성 모드들로는, 애플리케이션이 프로세서 유닛들(102) 중 3개의 서로 다른 프로세서 유닛들 상에서 실행되어 동일한 데이터를 처리할 수 있다. 3개의 프로세서 유닛들 상에서 실행되는 애플리케이션으로부터의 출력들이 데이터 관리기(112) 내의 무결성 관리기(120)에 의해 비교되어 선출될 수 있다. 선출시, 3개의 프로세서 유닛들로부터의 출력들이 비교된다. 출력 값에 대해 대다수가 식별된다. 대다수는 3개 중 2개 또는 3개 중 3개의 출력 값들이 동일한 것일 수 있다. 이 출력 값은 사용되는 값이다. 소수의 출력 값은 폐기된다.

[0054] 다음에 도 2를 참조하면, 예시적인 실시예에 따라 혼합된 무결성 데이터 처리 시스템의 예시가 도시된다. 이 예시적인 예에서, 혼합된 무결성 데이터 처리 시스템(200)은 도 1의 데이터 관리기(112)의 제어에 따라 도 1의 프로세서 유닛들(102)을 사용하여 구현될 수 있다.

[0055] 이 예시적인 예에서, 프로세서 유닛(202) 및 프로세서 유닛(204)은 멀티 코어 프로세서들이다. 도시된 바와 같이, 프로세서 유닛(202)은 코어(206) 및 코어(208)를 포함한다. 프로세서 유닛(204)은 코어(210) 및 코어(212)를 포함한다. 도시된 바와 같이, 이러한 서로 다른 코어들은 타임 슬라이스(214), 타임 슬라이스(216) 및 타임 슬라이스(218)와 같은 타임 슬라이스들 동안 애플리케이션들을 실행한다.

[0056] 이 예시적인 예에서, 프로세서 유닛(202) 내의 코어(206)는 타임 슬라이스(214) 동안 애플리케이션 1(220)을, 타임 슬라이스(216) 동안 애플리케이션 4(222)를, 그리고 타임 슬라이스(218) 동안 애플리케이션 1(220)을 실행한다. 프로세서 유닛(202) 내의 코어(208)는 타임 슬라이스(214) 동안 애플리케이션 2(224)를, 타임 슬라이스(216) 동안 애플리케이션 5(226)를, 그리고 타임 슬라이스(218) 동안 애플리케이션 6(228)을 실행한다.

[0057] 도시된 바와 같이, 프로세서 유닛(204) 내의 코어(210)는 타임 슬라이스(214) 동안 애플리케이션 1(220)을, 타임 슬라이스(216) 동안 애플리케이션 4(222)를, 그리고 타임 슬라이스(218) 동안 애플리케이션 1(220)을 실행한다. 프로세서 유닛(204) 내의 코어(212)는 타임 슬라이스(214) 동안 애플리케이션 3(230)을, 타임 슬라이스(216) 동안 애플리케이션 5(226)를, 그리고 타임 슬라이스(218) 동안 애플리케이션 3(230)을 실행한다.

[0058] 애플리케이션 1(220)은 고 무결성 모드를 사용하여 프로세서 유닛(202) 내의 코어(206) 및 프로세서 유닛(204) 내의 코어(210)에 의해 실행된다. 비슷한 방식으로, 애플리케이션 4(222)는 또한 코어(206) 및 코어(210)에 의해 고 무결성으로 실행된다. 프로세서 유닛(202) 내의 코어(208) 및 프로세서 유닛(204) 내의 코어(212)는 둘 다 고 무결성 모드로 애플리케이션 5(226)를 실행한다. 애플리케이션 2(224), 애플리케이션 3(230), 애플리케이션 4(222) 및 애플리케이션 6(228)은 프로세서 유닛(204) 내의 코어(210) 및 프로세서 유닛(202) 내의 코어(208)에 의해 표준 무결성으로 실행된다. 데이터의 처리 또는 메시지들의 교환 중 적어도 하나는 이 예에서 혼

합된 무결성 모드들을 제공하도록 도 1의 데이터 관리기(112)에 의해 관리된다. 일부 예들에서는, 이 프로세스가 코어들에 대한 모드들에 적용되어 보다 높은 수준의 입도를 달성할 수 있다.

[0059] 도 1의 프로세서 유닛들(102)이 도 1의 데이터(114)의 혼합된 무결성 처리를 위해 어떻게 구현될 수 있는지의 일 구현의 일례로서, 혼합된 무결성 데이터 처리 시스템(200)의 예시가 제공된다. 이 예시는 다른 예시적인 예들이 구현될 수 있는 방식을 제한하려는 것은 아니다. 예를 들어, 다른 예시적인 예들에서는 다른 수들의 프로세서 유닛들이 구현될 수 있다.

[0060] 다음에 도 3을 참조하면, 예시적인 실시예에 따라 복수의 프로세서들에 대한 데이터 전송을 관리하기 위한 프로세스의 흐름도의 예시가 도시된다. 이 도면에 예시된 프로세스는 도 1의 무결성 관리기(120)에서 구현될 수 있다.

[0061] 이 프로세스는 프로세서 유닛들 및 외부 노드와 통신하는 하드웨어에 위치된 무결성 관리기에서 프로세서 유닛들과 외부 노드 간에 교환되는 메시지들을 수신함으로써 시작되며, 고 무결성 모드가 선택된 경우에 프로세서 유닛들로부터의 중복 계산된 출력들이 일치함을 검사한다(동작(300)). 이 프로세스는 혼합된 무결성 모드들 중 선택된 모드를 기초로 프로세서 유닛들에 의한 외부 노드와의 메시지들의 교환을 관리한다(동작(302)). 이후 프로세스가 종료된다.

[0062] 다음에 도 4를 참조하면, 예시적인 실시예에 따라 프로세서 유닛들을 동기화하기 위한 프로세스의 흐름도의 예시가 도시된다. 도 4에 예시된 프로세스는 도 1의 내부 동기화기(130)에서 구현될 수 있다.

[0063] 이 프로세스는 동기화 시스템과 통신하는 제1 프로세서 유닛과 제2 프로세서 유닛 사이에 원하지 않는 양의 스큐가 존재하는지 여부를 결정하기 위해 외부 동기화기와 통신함으로써 시작된다(동작(400)). 이 프로세스는 원하지 않는 스큐가 존재할 때 제1 프로세서 유닛과 제2 프로세서 유닛 사이의 스큐의 양이 감소되도록, 필요한 결과를 생성하지 않고 조치를 수행할 것을 제1 프로세서 유닛에 선택적으로 지시한다(동작(402)). 이후 프로세스가 종료된다. 한 예시적인 예에서, 이 프로세스는 제1 프로세서 유닛 및 제2 프로세서 유닛에 의해 생성된 대응하는 메시지들에 대해 무결성 검사들이 수행되는 고 무결성 모드를 위해 제1 프로세서 유닛과 제2 프로세서 유닛이 서로 연관될 때 사용될 수 있다.

[0064] 다음에 도 5를 참조하면, 예시적인 실시예에 따라 프로세서 유닛들을 동기화하기 위한 프로세스의 흐름도의 예시가 도시된다. 도 5에 도시된 프로세스는 도 1의 데이터 처리 환경(100)에서 구현될 수 있다. 이 프로세스는 도 1의 프로세서 유닛들(102) 내의 프로세서 유닛에서 구현될 수 있다. 특히, 프로세스는 도 1의 제1 프로세서 유닛(132) 상에서 실행되는 내부 동기화기(130)와 같은 내부 동기화기에서 구현될 수 있다.

[0065] 이 프로세스는 내부 동기화기 프로세서 유닛이 외부 동기화기 내의 공통 레지스터에 로직 1을 기록하는 것으로 시작된다(동작(500)). 공통 레지스터는 프로세서 유닛들이 데이터를 처리하기 위해 프로세서 유닛들의 타이밍의 서로에 대한 정렬을 가능하게 함으로써 액세스될 수 있다. 프로세서 유닛들은 특정 클록 속도로 실행되도록 설계될 수 있다. 그러나 프로세서 유닛들의 실제 물리적 구현들은 규격들로부터 약간 달라질 수 있다. 이 레지스터는 프로세서 유닛들이 동기화 방식으로 가능한 한 동일한 시점에 가깝게 명령들을 실행하도록, 실행 중에 프로세서 유닛들의 타이밍을 정렬하는 데 사용될 수 있다.

[0066] 이 프로세스는 외부 동기화기 내의 공통 상태 레지스터를 판독한다(동작(502)). 공통 상태 레지스터로부터 판독된 값이 로직 1인지 여부에 관한 결정이 이루어진다(동작(504)). 공통 상태 레지스터는 프로세서 유닛들 사이에 공유된다. 단순한 상태 레지스터는 하나보다 많은 프로세서 유닛에 의해 확인될 수 없다.

[0067] 그 값이 로직 1이라면, 프로세서 유닛이 처리를 시작한다(동작(506)). 이후 프로세스가 종료된다. 예시적인 예에서, 처리는 초기화, 애플리케이션 시작, 애플리케이션의 실행 계속, 또는 다른 어떤 적당한 조치를 포함할 수 있다. 다시 동작(504)을 참조하면, 값이 로직 1이 아니라면, 프로세스는 동작(502)으로 돌아간다.

[0068] 이제 도 6을 참조하면, 예시적인 실시예에 따라 프로세서 유닛들을 동기화하기 위한 프로세스의 흐름도의 예시가 도시된다. 흐름도는 제1 프로세서 유닛 상에서 실행되는 내부 동기화기, 이를테면 도 1의 제1 프로세서 유닛(132) 상에서 실행되는 내부 동기화기(130)에 의해 수행되는 동작들을 보여준다. 이 프로세스는 2개 또는 그보다 많은 프로세서 유닛들에 의해 실행되어 프로세서 유닛들에 의한 명령들의 처리를 동기화할 수 있다.

[0069] 이 예에서, 프로세서 유닛들 각각은 경과한 시간을 측정하는 클록을 갖는다. 클록은 프로세스가 작업에 대해 활발히 작동하고 있는 시간을 측정하는 중앙 처리 유닛 시간과는 달리 실시간을 측정하는 디지털 클록일 수 있다.

[0070] 도시된 바와 같이, 클록은 프로세서 유닛들에 의해 판독되는 외부 클록일 수 있다. 예를 들어, 클록은 외부 동기화기에 위치될 수 있다. 클록은 프로세서 사이클들이 아닌 날짜, 시간, 분 그리고 초 단위로 연대순의 시간을 측정한다.

[0071] 이 프로세스는 제1 프로세서 유닛 내의 내부 동기화기가 외부 동기화기 내의 클록을 판독하는 것으로 시작된다(동작(600)). 클록은 제1 프로세서 유닛에 대한 것이며, 동일한 애플리케이션 또는 세트의 명령들을 실행하고 있는 임의의 다른 프로세서 유닛들에 대한 클록들과 동기화된다.

[0072] 차이 레지스터로부터 판독된 값이 임계치보다 더 큰지 여부에 관한 결정이 이루어진다(동작(602)). 차이 레지스터는 클록에 대한 현재 시간과 프로세서 유닛 내의 내부 동기화기에 의해 차이 레지스터가 판독된 시간의 스냅샷 간의 차이를 나타내는 값을 저장한다. 이 예시적인 예에서, 차이 레지스터는 외부 동기화기에 위치된다.

[0073] 외부 동기화기는 2개의 프로세서 유닛들이 차이 레지스터를 판독하는 시간들의 차이를 나타내는 값으로 차이 레지스터를 설정한다. 도시된 바와 같이, 선두 프로세서 유닛은 0의 값을 얻을 것이다. 뒤처진 프로세서 유닛은 실제 스큐 시간을 확인한다. 예를 들어, 뒤처진 프로세서 유닛은 선두 프로세서 유닛이 차이 레지스터를 판독하는 시간과 뒤처진 프로세서 유닛이 차이 레지스터를 판독하는 시간 사이의 차이인 수를 얻을 것이다.

[0074] 도시된 바와 같이, 임계치는 다수의 서로 다른 요소들을 기초로 선택될 수 있다. 예를 들어, 2개의 처리 유닛들 사이에 허용되는 스큐의 양이 임계치를 설정하는 데 사용될 수 있다. 차이 레지스터의 값이 임계치보다 더 크다면, 동기화기는 어떠한 결과들도 얻지 않고 조치를 수행하도록 제2 프로세서 유닛에 지시한다(동작(604)).

[0075] 이 예시적인 예에서는, 뒤처진 프로세서 유닛을 가속하기 위해 이 조치가 수행된다. 예를 들어, 뒤처진 프로세서 유닛의 사전 스케줄링된 유휴 시간이 제거될 수 있다. 대안으로, 선두 프로세서 유닛은 유휴 상태가 되어, 뒤처진 프로세서 유닛이 따라잡을 수 있게 할 수 있다. 예를 들어, 2개의 프로세서 유닛들이 동일한 데이터를 사용하여 동일한 프로그램을 실행하고 있을 때, 결과를 얻기 위한 데이터의 처리를 포함하지 않는 조치를 수행하는 것은 필요한 결과를 생성하지 않는 조치로 간주된다.

[0076] 다른 예에서, 동작(604)에서의 조치는 여분의 파티션 또는 사전 스케줄링된 유휴 시간을 제거하는 것일 수 있다. 즉, 여분의 파티션을 제거하는 것은, 프로세서 유닛으로 하여금 이 프로세서 유닛과 동기화되고 있는 다른 프로세서 유닛 상의 대응하는 애플리케이션에 의해 수행되지 않는 조치를 수행하게 한다. 그 다음, 프로세스는 동작(602)으로 돌아간다.

[0077] 동작(602)에서, 차이 레지스터의 값이 임계치보다 더 크지 않다면, 프로세스는 또한 동작(602)으로 돌아간다. 이 경우, 제1 프로세서 유닛이 제2 프로세서 유닛과 동기화되거나 제1 프로세서 유닛이 제2 프로세서 유닛에 뒤처지고 있다.

[0078] 이제 도 7을 참조하면, 예시적인 실시예에 따라 프로세서 유닛들로부터의 메시지들의 전송을 동기화하기 위한 프로세스의 흐름도의 예시가 도시된다. 도 7에 예시된 프로세스는 도 1의 제1 프로세서 유닛(132) 상에서 실행되는 내부 동기화기(130)와 같은 내부 동기화기에 의해, 무결성 관리기(120)의 대기열에 있는 메시지들을 기초로 메시지들의 전송을 관리하는 데 사용된다. 즉, 이 프로세스는 대기열의 비어 있는 부분이 미리 결정된 임계치 미만으로 감소될 때까지 메시지들이 대기열 내에 모일 수 있게 한다.

[0079] 이 프로세스는 외부 동기화기 내의 대기열 깊이 레지스터를 판독함으로써 시작된다(동작(700)). 대기열 깊이 레지스터들은 대기열에 얼마나 많은 메시지들이 있는지를 나타내는 대기열 깊이를 포함한다. 대기열은 2개의 프로세서 유닛들에 의해 전송되는 메시지들을 일치시키는 데 사용될 수 있다.

[0080] 외부 동기화기 내의 대기열이 n개의 메시지들에 대한 공간을 갖는지 여부에 관한 결정이 이루어진다(동작(702)). 동작(702)에서, n개의 메시지들은 미리 정해진 수의 메시지들이다. 일례로, 대기열은 10개의 메시지들을 보유할 수 있고, 프로세서 유닛은 매번 3개의 메시지들을 기록할 수 있다. 대기열 깊이가 9라면, 3개 또는 그보다 많은 메시지들을 기록하기에는 대기열에 충분한 공간이 없다.

[0081] 대기열이 n개의 메시지들에 대한 공간을 갖지 않는다면, 프로세스는 유휴 시간을 추가한 다음, 선두 프로세서 유닛으로 돌아간다(동작(704)). 그 다음, 프로세스는 동작(700)으로 돌아간다. 이 프로세스는 대기열에 3개의 메시지들을 기록하기 위해 공간이 이용 가능할 때까지 반복된다. 그 결과, 대기열이 n개의 메시지들에 대한 공간을 갖지 않는다면, 프로세서 유닛은 대기열에 메시지들을 배치하지 않는다.

[0082] 그렇지 않은 경우, 대기열이 n개의 메시지들에 대한 공간을 갖는다면, 프로세서 유닛 내의 내부 동기화기는 외부 동기화기 내의 대기열에 메시지들을 기록한다(동작(706)). 그 다음, 프로세스가 종료된다. 전송할 추가 메

시지들이 존재하면, 프로세스가 재시작될 수 있다.

[0083] 그 결과, 무결성 관리기는 제1 프로세서 유닛이 제1 대기열로부터 판독하고 제2 프로세서 유닛이 제2 대기열로부터 판독하며, 이러한 판독들이 둘 다 동일한 대기열 깊이로 수행되어 병렬 레인 일관성을 유지하도록 제1 대기열 및 제2 대기열을 제어할 수 있다. 메시지들의 기록은 내부 동기화기에 의해 수행되는 것으로 도시되지만, 이 동작은 내부 동기화기가 n개의 메시지들이 기록될 수 있는 시점을 표시하여 프로세서 유닛 내의 다른 컴포넌트에 의해 수행될 수 있다.

[0084] 도 8을 참조하면, 예시적인 실시예에 따라 메시지들을 전송하기 위한 프로세스의 흐름도의 예시가 도시된다. 이 프로세스는 도 1의 프로세서 유닛들(102)로부터 중복 생성된 아웃바운드 메시지들의 타이밍을 정렬하고 이러한 메시지들의 무결성을 보장하도록 도 1의 무결성 관리기(120)에서 구현될 수 있다. 이 예에서, 각각의 메시지는 식별자를 기초로 대기열 내의 특정 위치에 배치된다. 이 식별자는 메시지의 헤더에 있을 수 있다.

[0085] 이 프로세스는 제1 프로세서 유닛으로부터 메시지를 수신함으로써 시작된다(동작(800)). 이 프로세스는 메시지가 고 무결성 메시지인지 여부를 결정한다(동작(802)). 이 결정은 다수의 상이한 방식들로 이루어질 수 있다. 예를 들어, 헤더는 메시지가 고 무결성 메시지인지 여부에 관한 표시를 포함할 수 있다. 다른 예시적인 예에서는, 메시지가 고 무결성 메시지인지 여부를 결정하기 위해 구성 파일, 레지스터, 또는 다른 어떤 종류의 표시자가 검사될 수 있다.

[0086] 메시지가 고 무결성 메시지라면, 프로세스는 메시지를 저장한다(동작(804)). 동작(804)에서, 메시지는 다양한 타입들의 저장 디바이스들에 저장될 수 있다. 예를 들어, 메시지는 대기열에 저장되는데, 대기열은 다수의 서로 다른 타입들의 디바이스들을 사용하여 구현될 수 있다. 예를 들어, 대기열은 버퍼, 포트 어레이를 사용하여, 또는 다른 어떤 적당한 타입의 디바이스에서 구현될 수 있다. 도시된 바와 같이, 포트 어레이는 메시지들이 특정 위치가 아닌 서로 다른 위치들에 위치될 수 있는 동적 세트의 버퍼들일 수 있다. 포트 어레이의 경우, 무결성 관리기는 서로 다른 프로세서 유닛들에 대한 버퍼들을 탐색하여, 처리할 메시지들의 쌍을 만든다. 즉, 비교되어야 하는 메시지들은 동적 세트의 버퍼들에서 동일한 순서가 아닐 수 있다.

[0087] 이 프로세스는 제2 프로세서 유닛으로부터 대응하는 메시지를 수신하길 기다린다(동작(806)). 제2 메시지가 수신되었는지 아니면 타임아웃이 발생했는지에 관한 결정이 이루어진다(동작(808)). 타임아웃이 발생했다면, 메시지가 폐기되고(동작(810)), 에러가 표시된다(동작(812)). 이후 프로세스가 종료된다.

[0088] 다시 동작(808)을 참조하면, 제2 메시지가 수신되었다면, 프로세스는 두 메시지들 간의 비교를 수행한다(동작(814)). 이 비교는 다수의 상이한 방식들로 수행될 수 있다. 예를 들어, 비트별 비교, 순환 중복 검사 서명들의 비교, 그리고 다른 기술들이 사용될 수 있다.

[0089] 두 메시지들이 일치하는지 여부에 관한 결정이 이루어진다(동작(816)). 두 메시지들이 일치한다면, 프로세스는 메시지를 전송한다(동작(818)). 이 예시적인 예에서, 메시지는 네트워크로 전송된다. 이후 프로세스가 종료된다.

[0090] 그렇지 않으면, 프로세스는 두 메시지들 모두를 폐기한다(동작(820)). 동작(820)에서는, 일치가 존재하지 않기 때문에 메시지들이 폐기된다. 다음에 프로세스는 앞서 설명한 동작(812)으로 진행한다. 다시 동작(802)을 참조하면, 메시지가 고 무결성을 갖지 않는다면, 프로세스는 앞서 설명한 동작(818)으로 진행한다.

[0091] 다음에 도 9를 참조하면, 예시적인 실시예에 따라 메시지들을 전송하기 위한 프로세스의 흐름도의 예시가 도시된다. 이 프로세스는 중복 생성된 아웃바운드 메시지들의 타이밍을 정렬하고 이러한 메시지들의 무결성을 보장하도록 도 1의 무결성 관리기(120)에서 구현될 수 있다.

[0092] 이 프로세스는 프로세서 유닛으로부터 메시지를 수신함으로써 시작된다(동작(900)). 메시지가 고 무결성 메시지인지 여부에 관한 결정이 이루어진다(동작(902)). 메시지가 고 무결성 메시지라면, 프로세스는 메시지를 프로세서 유닛에 대한 레인 내의 대기열에 배치한다(동작(904)). 도시된 바와 같이, 대기열은 선입선출(FIFO: first in first out) 대기열이고 버퍼로서 구현될 수 있다.

[0093] 다른 프로세서에 대한 레인 내의 대기열이 하나 또는 그보다 많은 메시지들을 갖는지 여부에 관한 결정이 이루어진다(동작(906)). 다른 프로세서의 레인 내의 대기열이 하나 또는 그보다 많은 메시지들을 갖지 않는다면, 프로세스는 타임아웃 검사를 수행한다(동작(908)). 타임아웃이 발생하지 않았다면, 프로세스는 동작(906)으로 돌아간다.

[0094] 다른 대기열이 하나 또는 그보다 많은 메시지들을 갖는다면, 프로세스는 대기열들 내의 2개의 헤드 메시지들 간

의 비트별 비교를 수행한다(동작(910)). 이 예시적인 예에서, 헤드 메시지는 대기열에서 가져올 다음 메시지이다.

[0095] 두 메시지들이 일치하는지 여부에 관한 결정이 이루어진다(동작(912)). 두 메시지들이 일치한다면, 프로세스는 메시지를 전송한다(동작(914)). 이 예시적인 예에서, 메시지는 네트워크로 전송된다. 이후 프로세스가 종료된다. 그렇지 않으면, 프로세스는 두 메시지들 모두를 폐기하고 에러가 표시된다(동작(916)).

[0096] 다시 동작(902)을 참조하면, 메시지가 고 무결성 메시지가 아니라면, 프로세스는 동작(914)으로 진행한다. 다시 동작(908)으로 돌아가서, 타임아웃이 발생했다면, 프로세스는 동작(916)으로 진행한다.

[0097] 도 10에서는, 예시적인 실시예에 따라 메시지들을 수신하기 위한 프로세스의 흐름도의 예시가 도시된다. 이 프로세스는 무결성 관리기에서 구현될 수 있다. 이 프로세스는 메시지들을 가져오는 것을 예시한다.

[0098] 이 프로세스는 메시지의 도착을 검출함으로써 시작된다(동작(1000)). 프로세스는 메시지를 대기열의 뒤에 배치한다(동작(1002)). 그 다음, 프로세스는 대기열 깊이 카운터를 증분하고(동작(1004)), 이후 프로세스가 종료된다. 이 예시적인 예에서, 대기열 깊이 카운터는 메시지들이 어떻게 판독될 수 있는지를 나타내는 데 사용될 수 있다. 무결성 관리기는 이 대기열 깊이 카운터를 사용하여, 메시지들이 반드시 동일한 순서로 판독되게 할 수 있다.

[0099] 다음에 도 11을 참조하면, 예시적인 실시예에 따라 메시지들을 수신하기 위한 프로세스의 흐름도의 예시가 도시된다. 이 예시적인 예에서, 프로세스는 도 1의 무결성 관리기(120)에서 구현될 수 있다. 이 프로세스는 대기열에서 메시지들을 가져오는 것을 예시한다.

[0100] 이 프로세스는 제1 프로세서 유닛에 대한 제1 대기열에 대해 판독 요청을 수신함으로써 시작된다(동작(1100)). 제2 프로세서 유닛에 대한 제2 대기열이 제2 프로세서 유닛에 의해 이미 판독되었는지 여부에 관한 결정이 이루어진다(동작(1102)). 제2 대기열이 이미 판독되었다면, 프로세스는 제2 프로세서 유닛에 의해 판독된 제2 대기열에서 제2 대기열 깊이 값을 선택한다(동작(1104)).

[0101] 무결성 관리기 없이, 각각의 프로세서 유닛은 일반적으로, 메시지들의 판독시 사용하기 위해 그 자체의 대기열 깊이 카운터를 사용한다. 예시적인 예에서, 무결성 관리기는 특정 대기열 깊이 카운터로부터 대기열 깊이 값을 선택하여 두 프로세서 유닛들 모두가 동일한 수의 메시지들을 판독하게 한다.

[0102] 프로세스는 선택된 대기열 깊이 값을 제1 프로세서 유닛에 전송한다(동작(1106)). 동작(1106)은 제1 프로세서 유닛이 대기열 깊이 값을 기초로 표시된 메시지들의 수를 판독할 수 있게 한다. 프로세스는 대기열 깊이 카운터의 대기열 깊이 값을 판독된 메시지들의 수만큼 감소시킨다(동작(1108)). 동작(1108)에서, 대기열 깊이 카운터는 제1 프로세서 유닛에 대한 제1 대기열 깊이 카운터 또는 제2 프로세서 유닛에 대한 제2 대기열 깊이 값일 수 있다. 감소된 대기열 깊이 카운터는 제1 프로세서 유닛에 어떤 대기열 깊이 값이 제공되었는지를 기초로 한다. 이후 프로세스가 종료된다. 이 예시적인 예에서, 선택된 대기열 깊이 카운터로부터 특정 대기열 깊이 값의 선택은, 두 프로세서 유닛들이 메시지들을 판독하기 위해 동일한 대기열 깊이 값을 사용하도록 이루어진다.

[0103] 다시 동작(1102)을 참조하면, 제2 프로세서에 대한 제2 대기열이 아직 판독되지 않았다면, 프로세스는 제2 대기열 깊이 카운터의 제2 대기열 깊이 값이 제1 대기열 깊이 카운터의 제1 대기열 깊이 값보다 더 큰지 여부를 결정한다(동작(1110)). 제2 대기열 깊이 값이 제1 대기열 깊이 값보다 더 크다면, 프로세스는 메시지들을 판독하기 위해 제1 대기열 깊이 값을 사용한다(동작(1112)). 다음에 프로세스는 동작(1106)으로 진행한다. 그렇지 않으면, 프로세스는 메시지들을 판독하기 위해 제2 대기열 깊이 값을 사용하고(동작(1114)) 동작(1106)으로 진행한다.

[0104] 다음에 도 12를 참조하면, 예시적인 실시예에 따라 메시지들을 수신하기 위한 프로세스의 흐름도의 예시가 도시된다. 이 예시적인 예에서, 프로세스는 도 1의 무결성 관리기(120)에서 구현될 수 있다. 이 프로세스는 메시지들을 푸시하는 것을 예시한다.

[0105] 이 프로세스는 메시지의 도착을 검출함으로써 시작된다(동작(1200)). 프로세스는 메시지를 프로세서 유닛에 대한 메모리에 푸시한다(동작(1202)). 그 다음, 프로세스는 대기열 깊이를 증분하고(동작(1204)), 이후 프로세스가 종료된다.

[0106] 다음에 도 13을 참조하면, 예시적인 실시예에 따라 메시지들을 수신하기 위한 프로세스의 흐름도의 예시가 도시된다. 이 예시적인 예에서, 프로세스는 도 1의 무결성 관리기(120)에서 구현될 수 있다. 이 프로세스는 메시지들을 푸시하는 것을 예시한다. 이 프로세스는 프로세서 유닛 내의 메모리로부터 판독할 메시지들을 프로세서

유닛에 알리는 데 사용될 수 있다.

[0107] 이 프로세스는 제1 프로세서 유닛으로부터 메시지들을 판독하라는 요청을 수신함으로써 시작된다(동작(1300)). 제2 프로세서 유닛에 대한 제2 대기열이 이미 판독되었는지 여부에 관한 결정이 이루어진다(동작(1302)). 대기열이 이미 판독되었다면, 프로세스는 제2 프로세서 유닛에 대한 대기열에 대해 제2 판독 포인터를 제공한다(동작(1304)). 이 예시적인 예에서, 판독 포인터는 대기열로부터 어떻게 메시지들이 판독될 수 있는지를 가리킨다. 예를 들어, 판독 포인터는 대기열에서 프로세서 유닛에 의해 판독될 수 있는 마지막 메시지를 가리킬 수 있다. 즉, 판독 포인터는 대기열 깊이 값과 비슷한 대기열 깊이를 나타낼 수 있다.

[0108] 프로세서 유닛은 그 내부 메모리로부터 판독 포인터까지 다수의 메시지들을 판독한다(동작(1306)). 메시지들은 이 예에서 무결성 관리기에 의해 푸시된다. 무결성 관리기는 푸시 카운터를 판독된 메시지들의 수만큼 감소시킨다(동작(1308)). 이후 프로세스가 종료된다. 무결성 관리기가 새로운 메시지를 수신하면, 무결성 관리기는 새로운 메시지를 프로세서 유닛 내의 메모리에 푸시하고 푸시 카운트를 업데이트한다. 판독 카운트는 그러한 푸시된 메시지들 중 얼마나 많은 메시지들을 프로세서 유닛이 판독하도록 허용되는지이다. 따라서 X개의 메시지들이 푸시되고 프로세서 유닛이 Y개의 메시지들을 판독하도록 허용된다면, X-Y개의 메시지들이 프로세서 메모리에 남는다. 푸시 카운트는 Y만큼 감소되어 메모리에 판독되지 않은 메시지들이 얼마나 많이 남아 있는지를 나타낸다.

[0109] 다시 동작(1302)을 참조하면, 제2 프로세서 유닛에 대한 제2 대기열이 아직 판독되지 않았다면, 프로세스는 제2 프로세서 유닛에 대한 푸시 카운터가 제1 레인에 대한 푸시 카운터보다 더 큰지 여부를 결정한다(동작(1310)). 제2 대기열 깊이에 대한 푸시 카운터가 제1 레인에 대한 제1 푸시 카운터보다 더 크다면, 프로세스는 제1 푸시 카운터의 값을 메시지들을 판독하기 위한 판독 포인트로서 사용한다(동작(1312)). 다음에 프로세스는 동작(1306)으로 진행한다. 그렇지 않으면, 프로세스는 제2 레인에 대한 푸시 카운터를 판독 포인터로서 사용한다(동작(1314)). 다음에 프로세스는 동작(1306)으로 진행한다.

[0110] 도시된 서로 다른 실시예들의 흐름도들 및 블록도들은 예시적인 실시예의 장치들 및 방법들의 일부 가능한 구현들의 아키텍처, 기능 및 동작을 예시한다. 이와 관련하여, 흐름도들 또는 블록도들 내의 각각의 블록은 모듈, 세그먼트, 기능, 또는 동작이나 단계의 일부 중 적어도 하나를 나타낼 수 있다. 예를 들어, 블록들 중 하나 또는 그보다 많은 블록은 프로그램 코드, 하드웨어, 또는 프로그램 코드와 하드웨어의 결합으로 구현될 수 있다. 하드웨어로 구현될 때, 하드웨어는 예를 들어, 흐름도들 또는 블록도들의 하나 또는 그보다 많은 동작들을 수행하도록 제조 또는 구성되는 집적 회로들의 형태를 취할 수 있다. 프로그램 코드와 하드웨어의 결합으로서 구현될 때, 구현은 펌웨어의 형태를 취할 수 있다. 흐름도들 또는 블록도들의 각각의 블록은 특수 목적 하드웨어 및 특수 목적 하드웨어에 의해 실행되는 프로그램 코드의 상이한 동작들 또는 결합들을 수행하는 특수 목적 하드웨어 시스템들을 사용하여 구현될 수 있다.

[0111] 예시적인 실시예의 일부 대안적인 구현들에서는, 블록들에서 언급된 기능 또는 기능들이 도면들에서 언급된 순서와 다르게 발생할 수 있다. 예를 들어, 어떤 경우들에는, 연속하여 도시된 2개의 블록들이 실질적으로 동시에 수행될 수 있고, 또는 블록들이 수반되는 기능에 따라 간혹 역순으로 수행될 수 있다. 또한, 흐름도 또는 블록도에서 예시된 블록들 외에도 다른 블록들이 추가될 수 있다. 예를 들어, 도 12의 프로세스는 메시지들의 판독에 관해 설명된다. 이 프로세스는 또한, 메시지들을 기록하거나 전송하는 데 사용될 수 있다.

[0112] 이제 도 14를 참조하면, 예시적인 실시예에 따라 데이터 처리 시스템의 블록도의 예시가 도시된다. 데이터 처리 시스템(1400)은 도 1의 컴퓨터 시스템(110) 및 외부 노드(116)를 구현하는 데 사용될 수 있다. 이 예시적인 예에서, 데이터 처리 시스템(1400)은 프로세서 유닛(1404), 메모리(1406), 영구 저장소(1408), 통신 유닛(1410), 입력/출력(I/O: input/output) 유닛(1412) 그리고 디스플레이(1414) 간의 통신들을 제공하는 통신 프레임워크(1402)를 포함한다. 이 예에서, 통신 프레임워크(1402)는 버스 시스템의 형태를 취할 수 있다.

[0113] 프로세서 유닛(1404)은 메모리(1406)에 로딩될 수 있는 소프트웨어에 대한 명령들을 실행하는 역할을 한다. 프로세서 유닛(1404)은 특정 구현에 따라, 다수의 프로세서들, 멀티-프로세서 코어, 또는 다른 어떤 타입의 프로세서일 수 있다.

[0114] 메모리(1406) 및 영구 저장소(1408)는 저장 디바이스들(1416)의 예들이다. 저장 디바이스는 예를 들어, 제한 없이, 데이터, 함수 형태의 프로그램 코드, 또는 다른 적당한 정보 중 적어도 하나와 같은 정보를 임시로나, 영구적으로나, 아니면 임시 및 영구적 둘 다로 저장할 수 있는 하드웨어의 임의의 부분(piece)이다. 저장 디바이스들(1416)은 또한 이러한 예시적인 예들에서 컴퓨터 판독 가능 저장 디바이스들로 지칭될 수 있다. 이러한 예

들에서 메모리(1406)는 예를 들어, 랜덤 액세스 메모리 또는 임의의 다른 적당한 휘발성 또는 비휘발성 저장 디바이스일 수 있다. 영구 저장소(1408)는 특정 구현에 따라 다양한 형태들을 취할 수 있다.

[0115] 예를 들어, 영구 저장소(1408)는 하나 또는 그보다 많은 컴포넌트들 또는 디바이스들을 포함할 수 있다. 예를 들어, 영구 저장소(1408)는 하드 드라이브, 솔리드 스테이트 하드 드라이브, 플래시 메모리, 재기록 가능한 광 디스크, 재기록 가능한 자기 테이프, 또는 상기의 어떤 결합일 수 있다. 영구 저장소(1408)에 의해 사용되는 매체는 또한 착탈식일 수 있다. 예를 들어, 영구 저장소(1408)에 착탈식 하드 드라이브가 사용될 수 있다.

[0116] 통신 유닛(1410)은 이러한 예시적인 예들에서, 다른 데이터 처리 시스템들 또는 디바이스들과의 통신들을 제공한다. 이러한 예시적인 예들에서, 통신 유닛(1410)은 네트워크 인터페이스 카드이다.

[0117] 입력/출력 유닛(1412)은 데이터 처리 시스템(1400)에 접속될 수 있는 다른 디바이스들과의 데이터의 입력 및 출력을 가능하게 한다. 예를 들어, 입력/출력 유닛(1412)은 키보드, 마우스, 또는 다른 어떤 적당한 입력 디바이스 중 적어도 하나를 통해 사용자 입력에 대한 접속을 제공할 수 있다. 또한, 입력/출력 유닛(1412)은 프린터에 출력을 전송할 수 있다. 디스플레이(1414)는 사용자에게 정보를 디스플레이하기 위한 메커니즘을 제공한다.

[0118] 운영 시스템, 애플리케이션들 또는 프로그램들 중 적어도 하나에 대한 명령들은 통신 프레임워크(1402)를 통해 프로세서 유닛(1404)과 통신하는 저장 디바이스들(1416)에 위치될 수 있다. 서로 다른 실시예들의 프로세스들은 메모리(1406)와 같은 메모리에 위치될 수 있는 컴퓨터 구현 명령들을 사용하여 프로세서 유닛(1404)에 의해 수행될 수 있다.

[0119] 이러한 명령들은 프로세서 유닛(1404) 내의 프로세서에 의해 판독 및 실행될 수 있는 프로그램 코드, 컴퓨터 사용 가능 프로그램 코드 또는 컴퓨터 판독 가능 프로그램 코드로 지칭된다. 서로 다른 실시예들에서의 프로그램 코드는 서로 다른 물리적 또는 컴퓨터 판독 가능 저장 매체, 이를테면 메모리(1406) 또는 영구 저장소(1408) 상에 구현될 수 있다.

[0120] 프로그램 코드(1418)는 선택적으로 착탈식인 컴퓨터 판독 가능 매체(1420) 상에 함수 형태로 위치되며, 프로세서 유닛(1404)에 의한 실행을 위해 데이터 처리 시스템(1400)으로 로딩되거나 전송될 수 있다. 이러한 예시적인 예들에서 프로그램 코드(1418)와 컴퓨터 판독 가능 매체(1420)는 컴퓨터 프로그램 제품(1422)을 형성한다. 예시적인 예에서, 컴퓨터 판독 가능 매체(1420)는 컴퓨터 판독 가능 저장 매체(1424)일 수 있다. 이러한 예시적인 예들에서, 컴퓨터 판독 가능 저장 매체(1424)는 프로그램 코드(1418)를 전파하거나 송신하는 매체라기보다는 프로그램 코드(1418)를 저장하는 데 사용되는 물리적 또는 유형의 저장 디바이스이다.

[0121] 대안으로, 프로그램 코드(1418)는 컴퓨터 판독 가능 신호 매체를 사용하여 데이터 처리 시스템(1400)으로 전송될 수 있다. 컴퓨터 판독 가능 신호 매체는 예를 들어, 프로그램 코드(1418)를 포함하는 전파 데이터 신호일 수 있다. 예를 들어, 컴퓨터 판독 가능 신호 매체는 전자기 신호, 광 신호 또는 임의의 다른 적절한 타입의 신호 중 적어도 하나일 수 있다. 이러한 신호들은 무선 통신 링크들, 광섬유 케이블, 동축 케이블, 전선 또는 임의의 다른 적절한 타입의 통신 링크와 같은 통신 링크들 중 적어도 하나를 통해 송신될 수 있다.

[0122] 데이터 처리 시스템(1400)에 대해 예시된 서로 다른 컴포넌트들은 서로 다른 실시예들이 구현될 수 있는 방식에 대한 구조적 제한들을 제공하려는 것이 아니다. 서로 다른 예시적인 실시예들이 데이터 처리 시스템(1400)에 대해 예시된 것들에 추가로 또는 그 대신에 컴포넌트들을 포함하는 데이터 처리 시스템으로 구현될 수 있다. 도 14에 도시된 다른 컴포넌트들은 도시된 예시적인 예들과 다를 수 있다. 서로 다른 실시예들은 프로그램 코드(1418)를 실행할 수 있는 임의의 하드웨어 디바이스 또는 시스템을 사용하여 구현될 수 있다.

[0123] 본 개시내용의 예시적인 실시예들은 도 15에 도시된 것과 같은 항공기 제조 및 서비스 방법(1500) 그리고 도 16에 도시된 것과 같은 항공기(1600)와 관련하여 설명될 수 있다. 먼저 도 15를 참조하면, 예시적인 실시예에 따라 항공기 제조 및 서비스 방법의 블록도의 예시가 도시된다. 예비 생산 동안, 항공기 제조 및 서비스 방법(1500)은 도 16의 항공기(1600)의 규격 및 설계(1502) 그리고 자재 조달(1504)을 포함할 수 있다.

[0124] 생산 동안에는, 항공기(1600)의 컴포넌트 및 하위 부품 제조(1506) 그리고 시스템 통합(1508)이 이루어진다. 이후, 항공기(1600)는 운항(1512)되기 위해 인증 및 납품(1510)을 거칠 수 있다. 고객에 의한 운항(1512) 동안, 항공기(1600)는 정기 유지보수 및 서비스(1514)를 위해 스케줄링되는데, 이는 수정, 재구성, 개조 및 다른 유지보수 또는 서비스를 포함할 수 있다.

[0125] 항공기 제조 및 서비스 방법(1500)의 프로세스들 각각은 시스템 통합자, 제3자, 오퍼레이터, 또는 이들의 어떤 결합에 의해 수행 또는 실행될 수 있다. 이러한 예들에서, 오퍼레이터는 고객일 수 있다. 이러한 설명을 목적

으로, 시스템 통합자는 임의의 수의 항공기 제작사들 및 메이저 시스템 하도급 업체들을 제한 없이 포함할 수 있고; 제3자는 임의의 수의 판매사들, 하도급 업체들, 공급사들을 제한 없이 포함할 수 있으며; 오퍼레이터는 항공사, 리스(leasing) 회사, 군사업체, 서비스 기관 등일 수 있다.

[0126] 이제 도 16을 참조하면, 예시적인 실시예가 구현될 수 있는 항공기의 블록도의 예시가 도시된다. 이 예에서, 항공기(1600)는 도 15의 항공기 제조 및 서비스 방법(1500)에 의해 생산되며, 복수의 시스템들(1604) 및 내부(1606)와 함께 기체(1602)를 포함할 수 있다. 시스템들(1604)의 예들은 추진 시스템(1608), 전기 시스템(1610), 유압 시스템(1612), 환경 시스템(1614) 및 컴퓨터 시스템(1616) 중 하나 이상을 포함한다. 임의의 수의 다른 시스템들이 포함될 수 있다. 항공 우주 산업의 예가 도시되지만, 서로 다른 예시적인 실시예들은 자동차 산업과 같은 다른 산업들에 적용될 수 있다. 본 명세서에서 구현되는 장치들 및 방법들은 도 15의 항공기 제조 및 서비스 방법(1500)의 단계들 중 적어도 하나의 단계 동안 이용될 수 있다.

[0127] 한 예시적인 예에서, 도 15의 컴포넌트 및 하위 부품 제조(1506)에서 생산된 컴포넌트들 또는 하위 부품들은 도 15에서 항공기(1600)가 운항중(1512)인 동안 생산된 컴포넌트들 또는 하위 부품들과 비슷한 방식으로 제작 또는 제조될 수 있다. 또 다른 예로서, 도 15의 컴포넌트 및 하위 부품 제조(1506) 그리고 시스템 통합(1508)과 같은 생산 단계들 동안 하나 또는 그보다 많은 장치 실시예들, 방법 실시예들, 또는 이들의 결합이 이용될 수 있다. 항공기(1600)가 운항중(1512)인 동안, 도 15에서 유지보수 및 서비스(1514) 동안, 또는 두 경우 모두, 하나 또는 그보다 많은 장치 실시예들, 방법 실시예들, 또는 이들의 결합이 이용될 수 있다.

[0128] 예를 들어, 도 1의 데이터 관리기(112)와 같은 데이터 관리기 또는 내부 동기화기들(124)과 같은 하나 또는 그보다 많은 내부 동기화기들 중 적어도 하나는 항공기(1600) 내의 컴퓨터 시스템(1616)에서 구현될 수 있다. 이러한 컴포넌트들의 사용은 혼합된 무결성 모드들에서 데이터의 처리를 가능하게 할 수 있다. 서로 다른 컴포넌트들은 데이터 처리의 동기화, 메시지들의 교환, 또는 이들의 어떤 결합을 현재 사용되는 컴퓨터 시스템들에 비해 보다 효율적인 방식으로 가능하게 할 수 있다.

[0129] 다수의 다른 예시적인 실시예들의 사용의 구현은 항공기(1600)의 조립을 실질적으로 더 신속히 처리하거나, 항공기(1600)의 비용을 줄일 수 있고, 또는 항공기(1600)의 조립을 더 신속히 처리하게 하는 것과 항공기(1600)의 비용을 줄이는 것 모두를 할 수 있다. 데이터 처리의 향상된 효율성은 항공기(1600)에 필요한 컴퓨터들의 수 또는 크기 중 적어도 하나를 감소시킬 수 있다. 이러한 향상된 효율성은 항공기 내의 현재 사용되는 데이터 처리 시스템들의 처리 능력들을 증가시켜, 유지보수 및 개조의 양을 줄일 수 있다.

[0130] 따라서 하나 또는 그보다 많은 예시적인 예들은 항공기와 같은 플랫폼들 또는 다른 적당한 플랫폼의 프로세서 시스템들에 의해 데이터의 전반적인 처리 비용을 줄이는 데 사용될 수 있다. 다른 타입의 플랫폼은 예를 들어, 모바일 플랫폼, 고정 플랫폼, 육상 기반 구조, 수계 기반 구조 및 공간 기반 구조를 포함한다. 보다 구체적으로, 플랫폼은 수상함, 탱크, 병력 수송차, 기차, 우주선, 우주 정거장, 위성, 잠수함, 자동차, 발전소, 다리, 댐, 집, 제조 설비, 빌딩 및 다른 적당한 플랫폼들일 수 있다.

[0131] 프로세서 유닛들을 사용하여 데이터를 처리하기 위한 원하는 중복 효율을 얻는 것의 기술적 문제점을 극복하는 하나 또는 그보다 많은 기술적 해결책들이 존재한다. 그 결과, 하나 또는 그보다 많은 기술적 해결책들은 프로세서 유닛들에 의한 데이터 처리시 효율을 높일 수 있다. 예를 들어, 하나 또는 그보다 많은 기술적 해결책들은 고 무결성 모드로 동작하는 프로세서 유닛들에 의한 메시지들의 처리 또는 데이터의 처리의 타이밍 정렬 중 적어도 하나를 가능하게 할 수 있다. 그 결과, 프로세서 유닛들은 고 무결성 모드, 표준 무결성 모드, 또는 이들의 어떤 결합을 사용하여 개별적으로 또는 실질적으로 동시에 데이터를 처리할 수 있다.

[0132] 서로 다른 예시적인 실시예들의 설명은 예시 및 설명을 목적으로 제시되었으며, 개시된 형태로 실시예들을 총망라하거나 이에 한정되도록 의도되는 것은 아니다. 서로 다른 예시적인 예들은 조치들 또는 동작들을 수행하는 컴포넌트들을 설명한다. 예시적인 실시예에서, 컴포넌트는 설명된 조치 또는 동작을 수행하도록 구성될 수 있다. 예를 들어, 컴포넌트는, 컴포넌트에 의해 수행되는 것으로 예시적인 예들에서 설명되는 조치 또는 동작을 수행하는 능력을 컴포넌트에 제공하는 구조에 대한 구성 또는 설계를 가질 수 있다.

[0133] 추가로, 본 개시내용은 다음 조항들에 따른 실시예들을 포함한다:

[0134] 조항 1. 프로세서 동기화 시스템은:

[0135] 제1 프로세서 유닛 상에서 실행되는 내부 동기화기를 포함하며, 여기서 내부 동기화기는 외부 동기화기와 통신하는 제1 프로세서 유닛과 제2 프로세서 유닛 사이에 원하지 않는 양의 스큐가 존재하는지 여부를 결정하기 위해 외부 동기화기와 통신하고, 그리고 제1 프로세서 유닛과 제2 프로세서 유닛 사이의 원하지 않는 양의 스큐가

존재할 때 원하지 않는 양의 스큐가 감소되게, 필요한 결과를 생성하지 않고 조치를 수행할 것을 제1 프로세서 유닛에 선택적으로 지시하도록 구성되고, 제1 프로세서 유닛 및 제2 프로세서 유닛에 의해 생성된 대응하는 메시지들에 대해 무결성 검사들이 수행되는 고 무결성 모드를 위해 제1 프로세서 유닛과 제2 프로세서 유닛이 서로 연관된다.

[0136] 조항 2. 조항 1의 프로세서 동기화 시스템에서, 제1 프로세서 유닛과 제2 프로세서 유닛은 동일한 타입이다.

[0137] 조항 3. 조항 1의 프로세서 동기화 시스템에서, 제1 프로세서 유닛과 제2 프로세서 유닛은 서로 다른 타입이다.

[0138] 조항 4. 조항 1의 프로세서 동기화 시스템에서, 외부 동기화기는 제1 프로세서 유닛과 제2 프로세서 유닛 사이의 원하지 않는 양의 스큐에 관한 정보를 저장한다.

[0139] 조항 5. 조항 4의 프로세서 동기화 시스템에서, 외부 동기화기는 원하지 않는 양의 스큐에 관한 정보를 레지스터들에 저장하고, 프로세서 동기화 시스템과 통신하는 제1 프로세서 유닛과 제2 프로세서 유닛 사이에 원하지 않는 양의 스큐가 존재하는지 여부를 결정하기 위해 외부 동기화기와 통신할 때, 내부 동기화기는 외부 동기화기의 제1 프로세서 유닛에 대해 제1 레지스터에 기록하고 외부 동기화기의 제2 프로세서 유닛에 대해 제2 레지스터를 판독하도록 구성되며, 제1 프로세서 유닛과 제2 프로세서 유닛이 명령들을 동시에 처리하도록 정렬되는 시점을 결정하기 위해 제1 레지스터와 제2 레지스터의 값들을 비교한다.

[0140] 조항 6. 조항 1의 프로세서 동기화 시스템에서, 외부 동기화기는 제1 프로세서 유닛에 대한 제1 클록 및 제2 프로세서 유닛에 대한 제2 클록을 포함하며, 제1 클록 및 제2 클록은 명령들을 동시에 처리하도록 제1 프로세서 유닛과 제2 프로세서 유닛을 정렬하는 데 사용된다.

[0141] 조항 7. 조항 6의 프로세서 동기화 시스템에서, 제1 프로세서 유닛은 선두 프로세서 유닛이고; 제2 프로세서 유닛은 뒤처진 프로세서 유닛이며; 차이 레지스터가 선두 프로세서 유닛으로 0 값을 반환하고; 그리고 차이 레지스터는 뒤처진 프로세서 유닛으로 원하지 않는 양의 스큐의 값을 반환한다.

[0142] 조항 8. 조항 1의 프로세서 동기화 시스템에서, 조치는 제1 프로세서 유닛 내의 내부 동기화기가 제1 프로세서 유닛이 제2 프로세서 유닛보다 더 느린 것을 검출하는 경우, 제1 프로세서 유닛으로부터 사전 스케줄링된 유휴 시간을 제거하며, 이는 제1 프로세서 유닛과 제2 프로세서 유닛 사이의 원하지 않는 양의 스큐를 감소시킨다.

[0143] 조항 9. 조항 8의 프로세서 동기화 시스템에서, 내부 동기화기는 사전 스케줄링된 유휴 시간을 제거한 후 제1 프로세서 유닛과 제2 프로세서 유닛 사이에 원하지 않는 양의 스큐가 여전히 존재한다면, 추가 사전 스케줄링된 유휴 시간을 제거하도록 구성된다.

[0144] 조항 10. 조항 1의 프로세서 동기화 시스템에서, 제1 프로세서 유닛은 선두 프로세서 유닛이고, 제2 프로세서 유닛은 뒤처진 프로세서 유닛이며, 조치는 제1 프로세서 유닛으로 하여금, 제1 프로세서 유닛을 더 느리게 동작하게 하는 사전 스케줄링된 유휴 시간을 추가하게 한다.

[0145] 조항 11. 조항 1의 프로세서 동기화 시스템은:

[0146] 제3 프로세서 유닛을 더 포함하며, 내부 동기화기는 외부 동기화기와 통신하는 제1 프로세서 유닛과 제2 프로세서 유닛 그리고 제3 프로세서 유닛 사이에 원하지 않는 양의 스큐가 존재하는지 여부를 결정하기 위해 외부 동기화기와 통신하고, 그리고 제1 프로세서 유닛과 제2 프로세서 유닛 그리고 제3 프로세서 유닛 사이의 원하지 않는 양의 스큐가 존재할 때 원하지 않는 양의 스큐가 감소되게, 필요한 결과를 생성하지 않고 조치를 수행할 것을 제1 프로세서 유닛에 선택적으로 지시하도록 구성된다.

[0147] 조항 12. 조항 1의 프로세서 동기화 시스템에서, 제1 프로세서 유닛 및 제2 프로세서 유닛은 멀티 코어 프로세서 유닛, 단일 코어 프로세서 유닛, 동종 멀티 코어 프로세서 유닛, 이종 멀티 코어 프로세서 유닛, 그래픽 프로세서 유닛 및 범용 프로세서 유닛 중 적어도 하나를 포함하는 그룹으로부터 각각 선택된다.

[0148] 조항 13. 조항 1의 프로세서 동기화 시스템에서, 내부 동기화기는 제1 프로세서 유닛 상에서 실행되는 소프트웨어 또는 하드웨어 중 적어도 하나를 포함한다.

[0149] 조항 14. 프로세서 유닛들을 동기화하기 위한 방법으로서, 이 방법은:

[0150] 동기화 시스템과 통신하는 제1 프로세서 유닛과 제2 프로세서 유닛 사이에 원하지 않는 양의 스큐가 존재하는지 여부를 결정하기 위해 외부 동기화기와 통신하는 단계; 및

[0151] 제1 프로세서 유닛에 제1 프로세서 유닛과 제2 프로세서 유닛 사이의 원하지 않는 양의 스큐가 존재할 때 원하

지 않는 양의 스큐가 감소되도록, 필요한 결과를 생성하지 않고 조치를 수행할 것을 제1 프로세서 유닛에 선택적으로 지시하는 단계를 포함하며, 제1 프로세서 유닛 및 제2 프로세서 유닛에 의해 생성된 대응하는 메시지들에 대해 무결성 검사들이 수행되는 고 무결성 모드를 위해 제1 프로세서 유닛과 제2 프로세서 유닛이 서로 연관된다.

[0152] 조항 15. 조항 14의 방법에서, 제1 프로세서 유닛과 제2 프로세서 유닛은 동일한 타입이다.

[0153] 조항 16. 조항 14의 방법에서, 제1 프로세서 유닛과 제2 프로세서 유닛은 서로 다른 타입이다.

[0154] 조항 17. 조항 14의 방법은:

[0155] 제1 프로세서 유닛과 제2 프로세서 유닛 사이의 원하지 않는 양의 스큐에 관한 정보를 외부 동기화기에 저장하는 단계를 더 포함한다.

[0156] 조항 18. 조항 17의 방법에서, 외부 동기화기는 원하지 않는 양의 스큐에 관한 정보를 레지스터들에 저장하고, 동기화 시스템과 통신하는 제1 프로세서 유닛과 제2 프로세서 유닛 사이에 원하지 않는 양의 스큐가 존재하는지 여부를 결정하기 위해 외부 동기화기와 통신하는 단계는:

[0157] 동기화 시스템의 제1 프로세서 유닛에 대해 제1 레지스터에 기록하는 단계;

[0158] 동기화 시스템의 제2 프로세서 유닛에 대해 제2 레지스터를 판독하는 단계; 및

[0159] 제1 프로세서 유닛과 제2 프로세서 유닛이 명령들을 동시에 처리하도록 정렬되는 시점을 결정하기 위해 제1 레지스터와 제2 레지스터의 값들을 비교하는 단계를 포함한다.

[0160] 조항 19. 조항 14의 방법에서, 외부 동기화기는 제1 프로세서 유닛에 대한 제1 클록 및 제2 프로세서 유닛에 대한 제2 클록을 포함하며, 외부 동기화기는 제2 프로세서 유닛에 대한 제2 클록 동안 레지스터를 판독한다.

[0161] 조항 20. 조항 14의 방법에서, 다수의 명령들은 제1 프로세서 유닛으로부터 사전 스케줄링된 유휴 시간을 제거하며, 이는 제1 프로세서 유닛과 제2 프로세서 유닛 사이의 원하지 않는 양의 스큐를 감소시킨다.

[0162] 조항 21. 조항 20의 방법은:

[0163] 사전 스케줄링된 유휴 시간을 제거한 후 제1 프로세서 유닛과 제2 프로세서 유닛 사이에 원하지 않는 양의 스큐가 여전히 존재한다면, 추가 사전 스케줄링된 유휴 시간을 제거하는 단계를 더 포함한다.

[0164] 조항 22. 조항 14의 방법에서, 제1 프로세서 유닛은 선두 프로세서 유닛이고, 제2 프로세서 유닛은 뒤처진 프로세서 유닛이며, 다수의 명령들은 제1 프로세서 유닛을 유휴 상태가 되게 한다.

[0165] 조항 23. 조항 14의 방법에서, 제3 프로세서 유닛이 존재하며, 이 방법은:

[0166] 동기화 시스템과 통신하는 제1 프로세서 유닛과 제2 프로세서 유닛 그리고 제3 프로세서 유닛 사이에 원하지 않는 양의 스큐가 존재하는지 여부를 결정하기 위해 외부 동기화기와 통신하는 단계; 및

[0167] 제1 프로세서 유닛과 제2 프로세서 유닛 그리고 제3 프로세서 유닛 사이의 원하지 않는 양의 스큐가 존재할 때 원하지 않는 양의 스큐가 감소되도록, 필요한 결과를 생성하지 않고 조치를 수행할 것을 제1 프로세서 유닛에 선택적으로 지시하는 단계를 더 포함하며, 제1 프로세서 유닛, 제2 프로세서 유닛 및 제3 프로세서 유닛에 의해 생성된 대응하는 메시지들에 대해 무결성 검사들이 수행되는 고 무결성 모드를 위해 제1 프로세서 유닛과 제2 프로세서 유닛 그리고 제3 프로세서 유닛이 서로 연관된다.

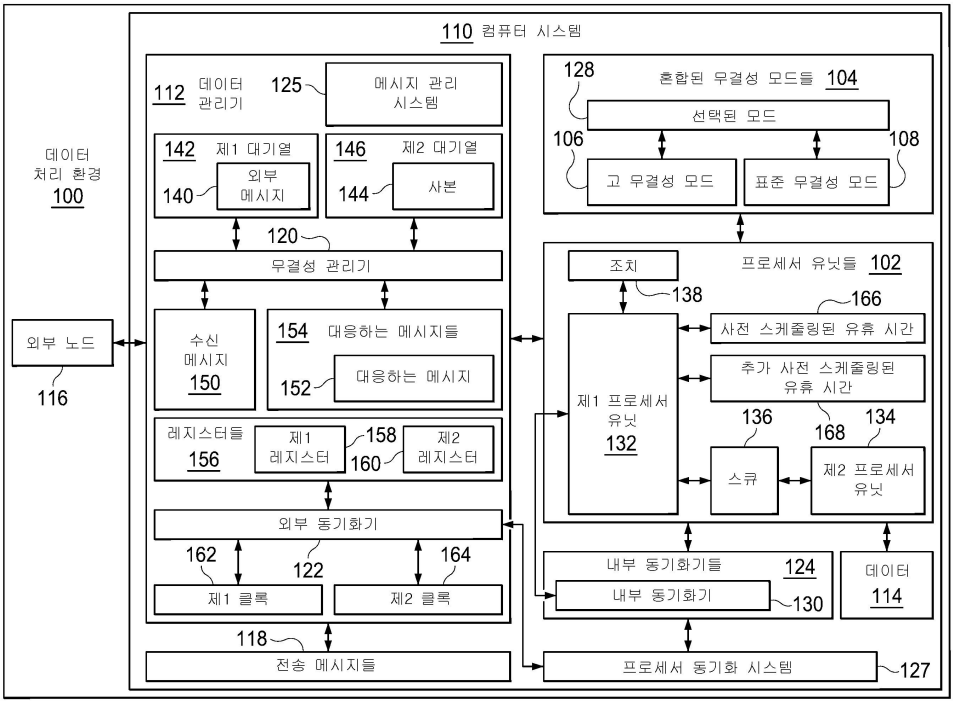
[0168] 조항 24. 조항 14의 방법에서, 제1 프로세서 유닛 및 제2 프로세서 유닛은 멀티 코어 프로세서 유닛, 단일 코어 프로세서 유닛, 동종 멀티 코어 프로세서 유닛, 이종 멀티 코어 프로세서 유닛, 그래픽 프로세서 유닛 및 범용 프로세서 유닛 중 적어도 하나를 포함하는 그룹으로부터 각각 선택된다.

[0169] 많은 수정들 및 변형들이 당해 기술분야에서 통상의 지식을 가진 자들에게 명백할 것이다. 추가로, 서로 다른 예시적인 실시예들은 다른 바람직한 실시예들과 비교할 때 다른 특징들을 제공할 수 있다. 예를 들어, 예시적인 실시예들은 혼합된 무결성 모드들에 관해 설명되었고, 예시적인 실시예는 락스텝을 실행하는 프로세서들, 또는 처리 동기화 또는 메시지 교환 중 적어도 하나가 설계되는 다른 타입들의 환경들에 적용될 수 있다. 선택된 실시예 또는 실시예들은 실시예들의 원리들, 실제 적용을 가장 잘 설명하기 위해, 그리고 당해 기술분야에서 통상의 지식을 가진 다른 자들이 고려되는 특정 용도에 맞는 다양한 수정들을 갖는 다양한 실시예들에 대한 개시

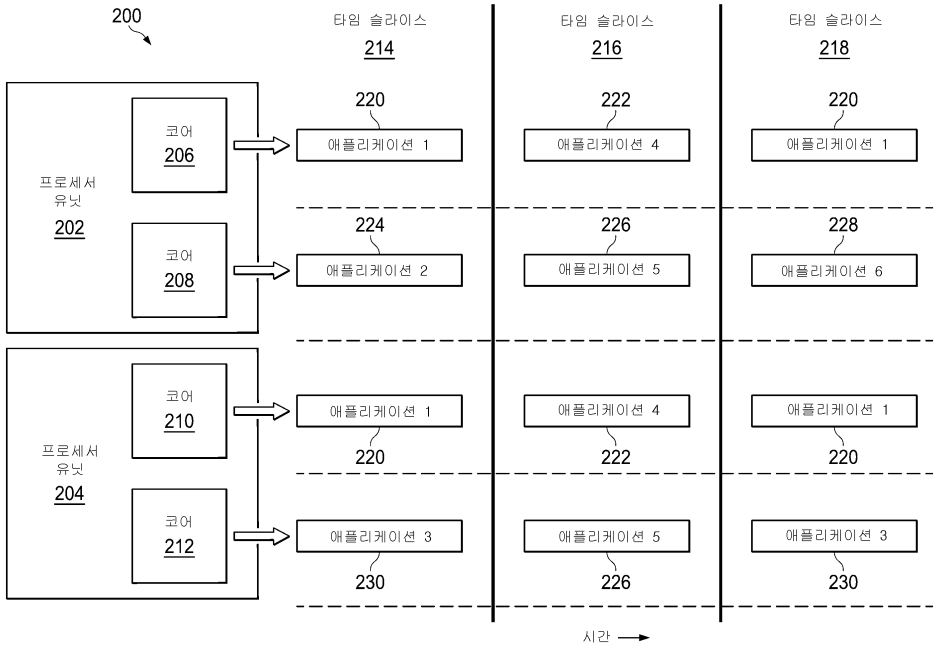
내용을 이해할 수 있게 하기 위해 선택되고 설명된다.

도면

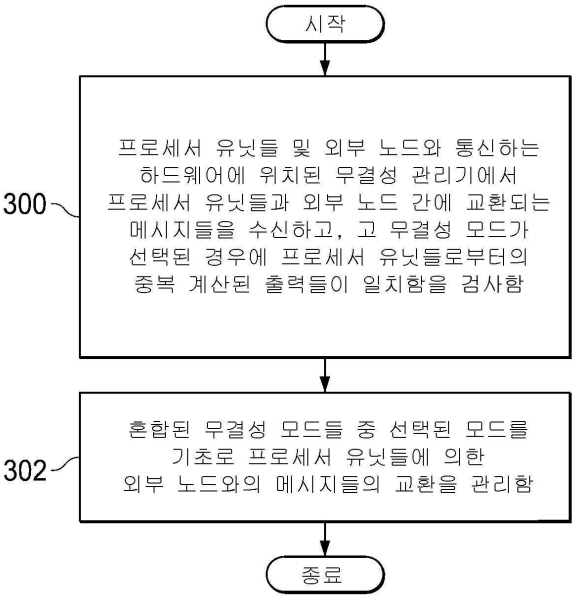
도면1



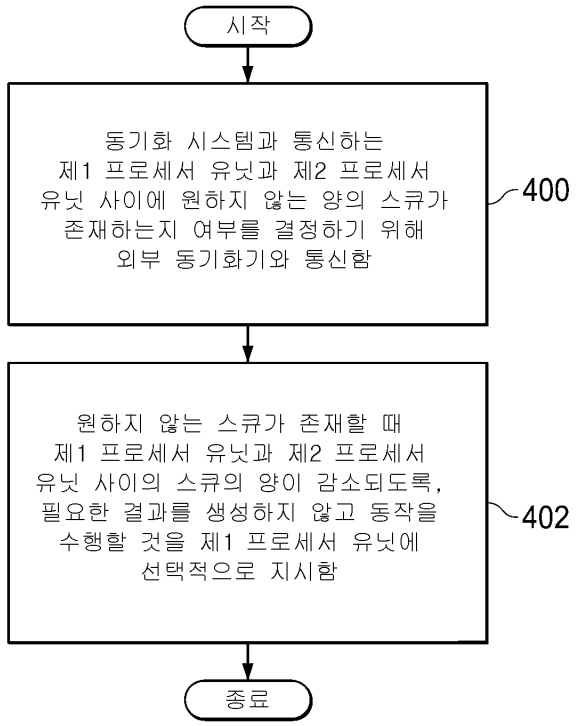
도면2



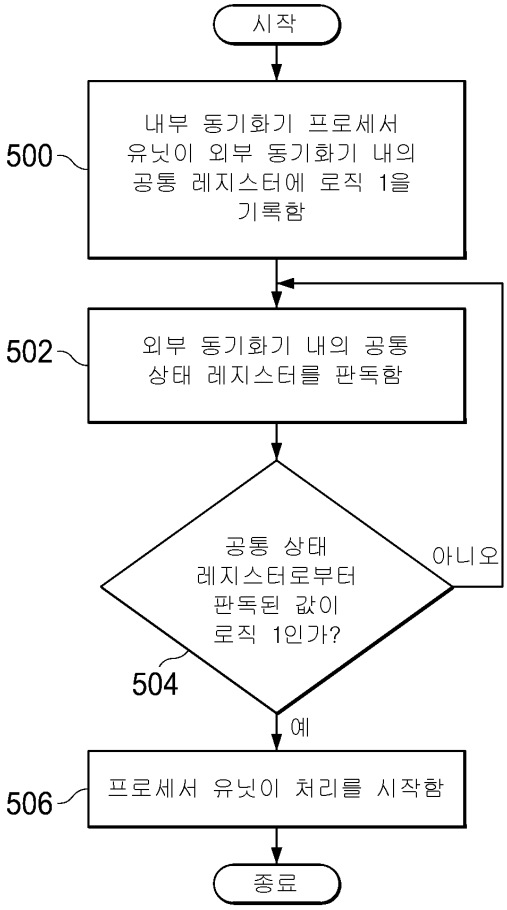
도면3


시작
프로세서 유닛들 및 외부 노드와 통신하는 하드웨어에 위치된 무결성 관리기에서 프로세서 유닛들과 외부 노드 간에 교환되는 메시지들을 수신하고, 고 무결성 모드가 선택된 경우에 프로세서 유닛들로부터의 중복 계산된 출력들이 일치함을 검사함
300
혼합된 무결성 모드들 중 선택된 모드를 기초로 프로세서 유닛들에 의한 외부 노드와의 메시지들의 교환을 관리함
302
종료

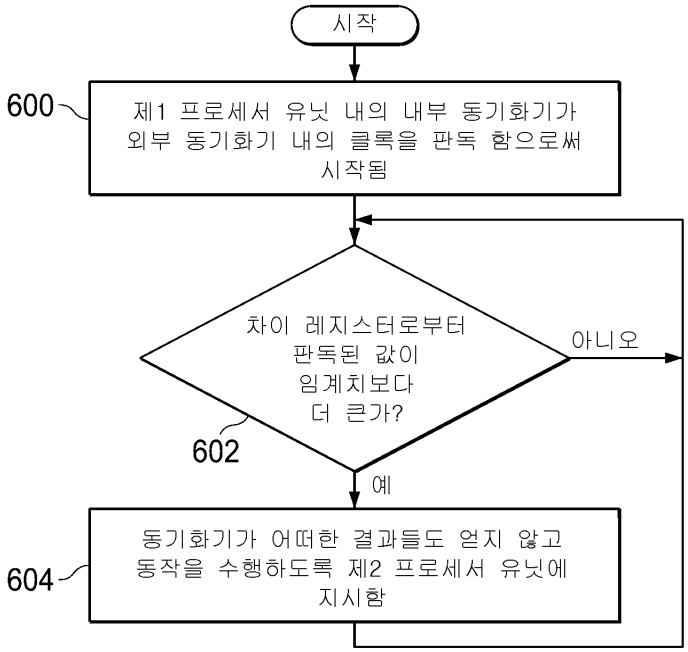
도면4


시작
동기화 시스템과 통신하는 제1 프로세서 유닛과 제2 프로세서 유닛 사이에 원하지 않는 양의 스큐가 존재하는지 여부를 결정하기 위해 외부 동기화기와 통신함
400
원하지 않는 스큐가 존재할 때 제1 프로세서 유닛과 제2 프로세서 유닛 사이의 스큐의 양이 감소되도록, 필요한 결과를 생성하지 않고 동작을 수행할 것을 제1 프로세서 유닛에 선택적으로 지시함
402
종료

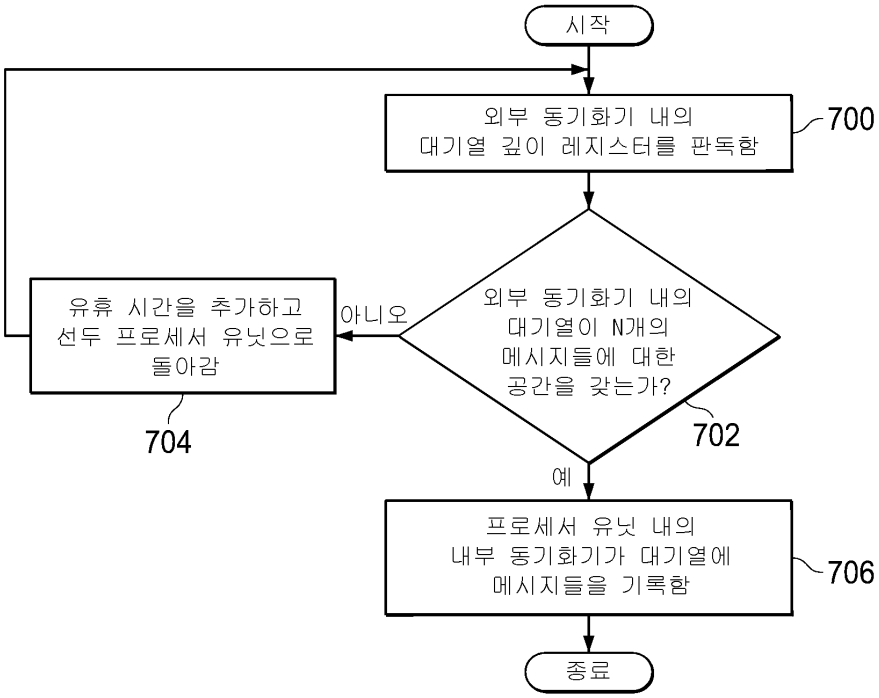
도면5


시작
내부 동기화기 프로세서 유닛이 외부 동기화기 내의 공통 레지스터에 로직 1을 기록함
500
외부 동기화기 내의 공통 상태 레지스터를 판독함
502
공통 상태 레지스터로부터 판독된 값이 로직 1인가?
504
아니오
예
프로세서 유닛이 처리를 시작함
506
종료

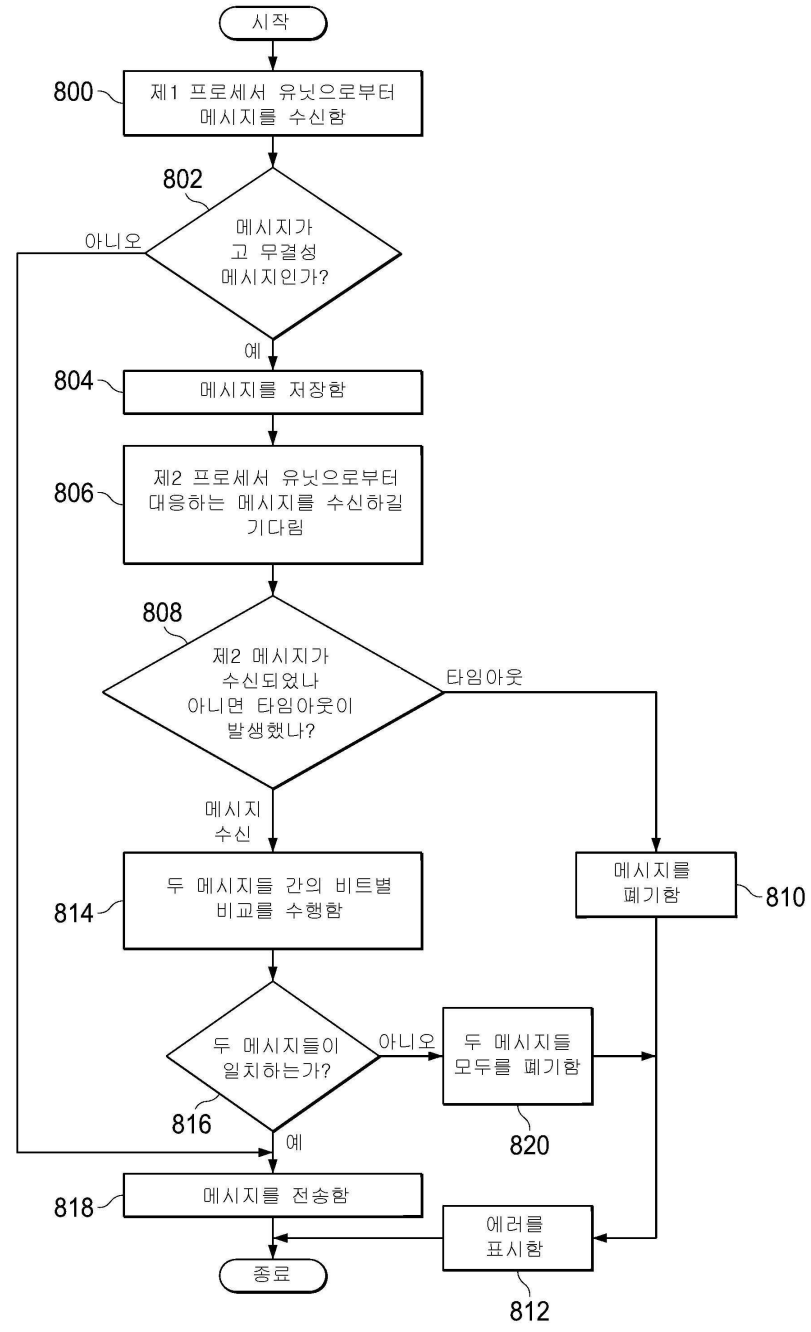
도면6


시작
제1 프로세서 유닛 내의 내부 동기화기가 외부 동기화기 내의 클록을 판독 함으로써 시작됨
600
차이 레지스터로부터 판독된 값이 임계치보다 더 큰가?
602
아니오
예
동기화기가 어떠한 결과들도 얻지 않고 동작을 수행하도록 제2 프로세서 유닛에 지시함
604

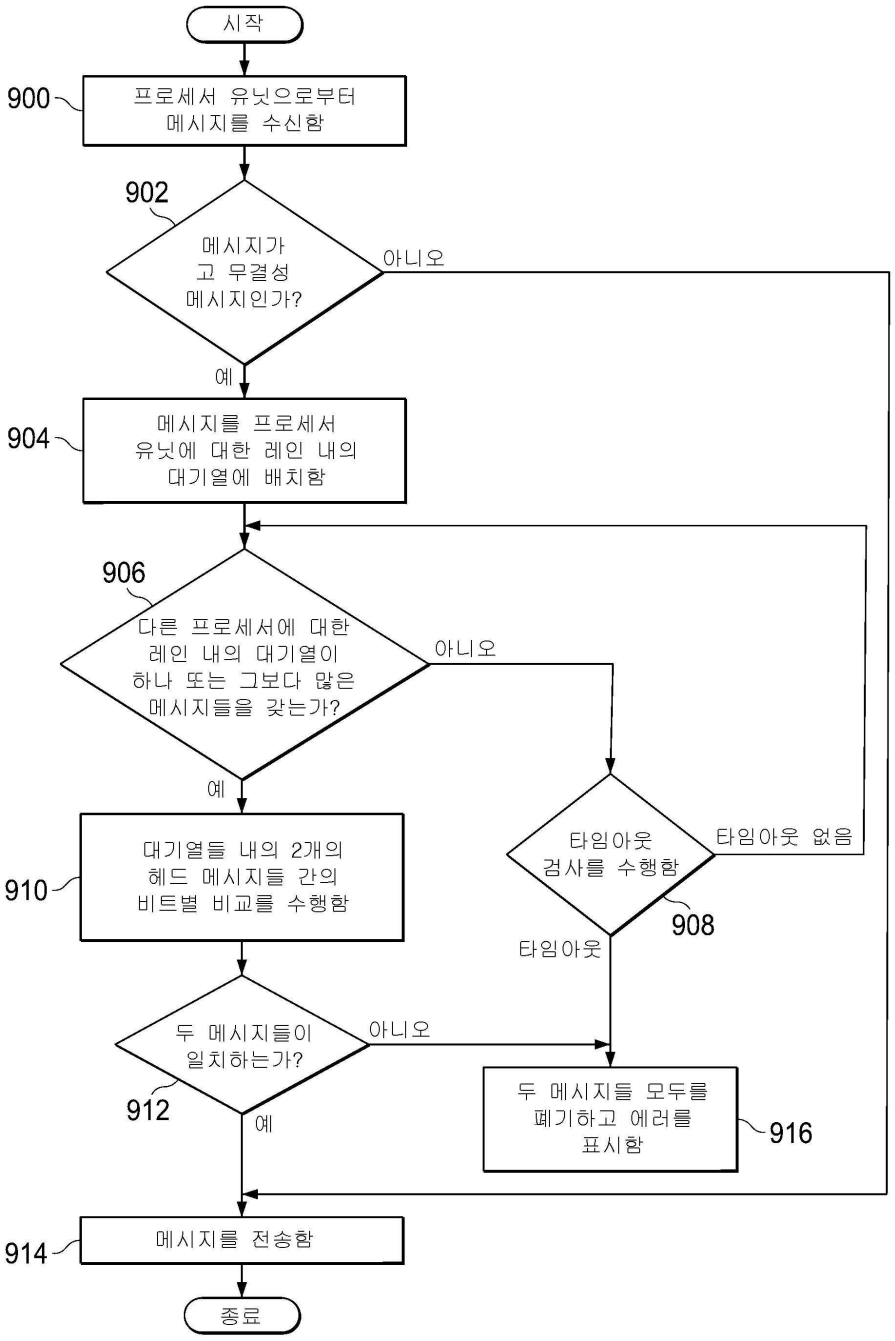
도면7


시작
외부 동기화기 내의
대기열 깊이 레지스터를 판독함
700
외부 동기화기 내의
대기열이 N개의
메시지들에 대한
공간을 갖는가?
702
아니오
유휴 시간을 추가하고
선두 프로세서 유닛으로
돌아감
704
예
프로세서 유닛 내의
내부 동기화기가 대기열에
메시지들을 기록함
706
종료

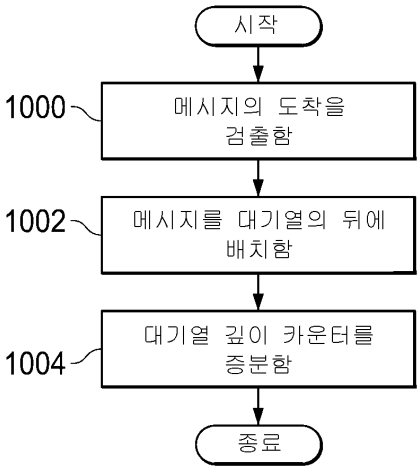
도면8


시작
800
제1 프로세서 유닛으로부터 메시지를 수신함
802
메시지가 고 무결성 메시지인가?
아니오
예
804
메시지를 저장함
806
제2 프로세서 유닛으로부터 대응하는 메시지를 수신하길 기다림
808
제2 메시지가 수신되었나 아니면 타임아웃이 발생했나?
타임아웃
메시지 수신
814
두 메시지들 간의 비트별 비교를 수행함
메시지를 폐기함
810
두 메시지들이 일치하는가?
816
아니오
두 메시지들 모두를 폐기함
820
예
818
메시지를 전송함
에러를 표시함
812
종료

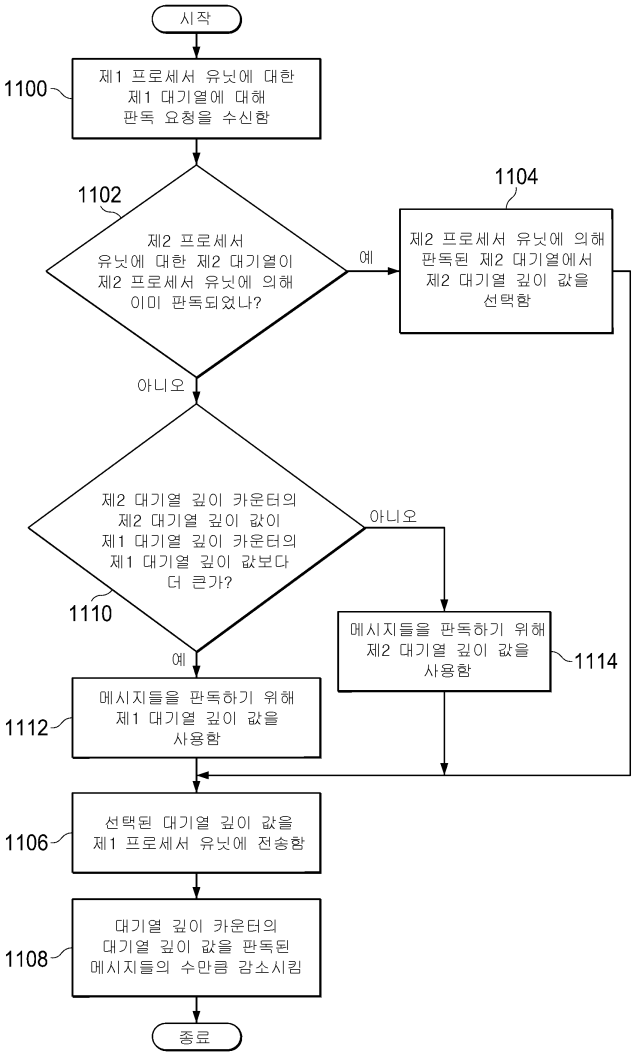
도면9


시작
900
프로세서 유닛으로부터 메시지를 수신함
902
메시지가 고 무결성 메시지인가?
아니오
예
904
메시지를 프로세서 유닛에 대한 레인 내의 대기열에 배치함
906
다른 프로세서에 대한 레인 내의 대기열이 하나 또는 그보다 많은 메시지들을 갖는가?
아니오
예
910
대기열들 내의 2개의 헤드 메시지들 간의 비트별 비교를 수행함
타임아웃 검사를 수행함
타임아웃 없음
908
타임아웃
두 메시지들이 일치하는가?
아니오
912
예
두 메시지들 모두를 폐기하고 에러를 표시함
916
914
메시지를 전송함
종료

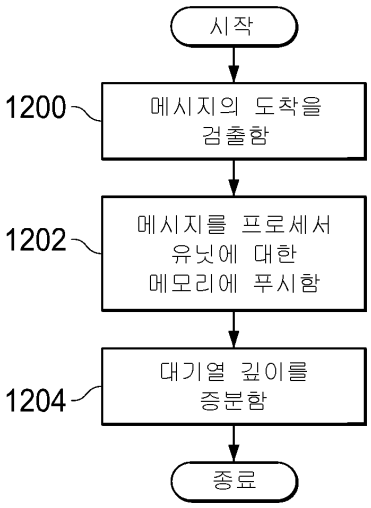
도면10


시작
1000
메시지의 도착을 검출함
1002
메시지를 대기열의 뒤에 배치함
1004
대기열 깊이 카운터를 증분함
종료

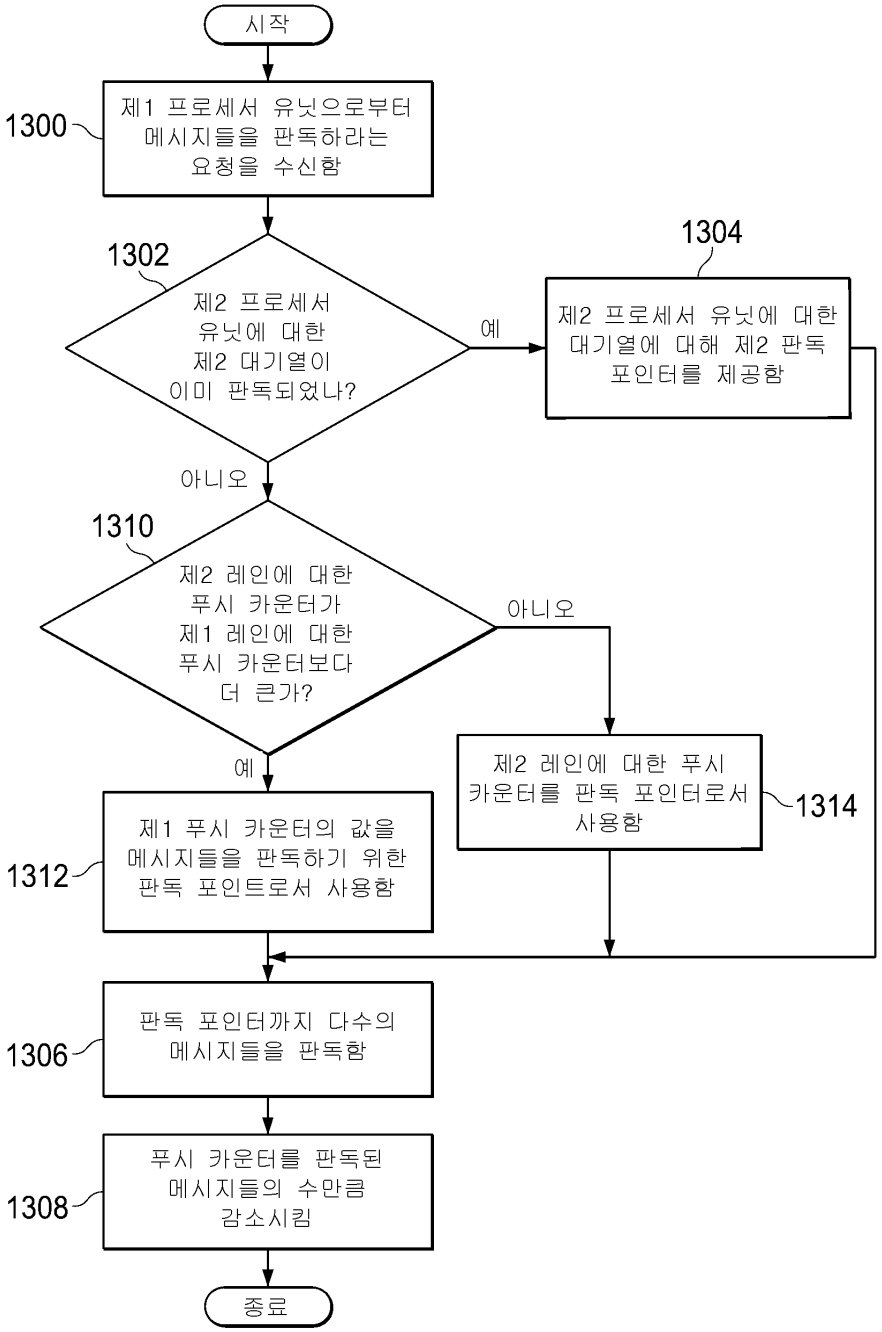
도면11


시작
1100
제1 프로세서 유닛에 대한 제1 대기열에 대해 판독 요청을 수신함
1102
제2 프로세서 유닛에 대한 제2 대기열이 제2 프로세서 유닛에 의해 이미 판독되었나?
예
1104
제2 프로세서 유닛에 의해 판독된 제2 대기열에서 제2 대기열 깊이 값을 선택함
아니오
제2 대기열 깊이 카운터의 제2 대기열 깊이 값이 제1 대기열 깊이 카운터의 제1 대기열 깊이 값보다 더 큰가?
1110
아니오
메시지들을 판독하기 위해 제2 대기열 깊이 값을 사용함
1114
예
1112
메시지들을 판독하기 위해 제1 대기열 깊이 값을 사용함
1106
선택된 대기열 깊이 값을 제1 프로세서 유닛에 전송함
1108
대기열 깊이 카운터의 대기열 깊이 값을 판독된 메시지들의 수만큼 감소시킴
종료

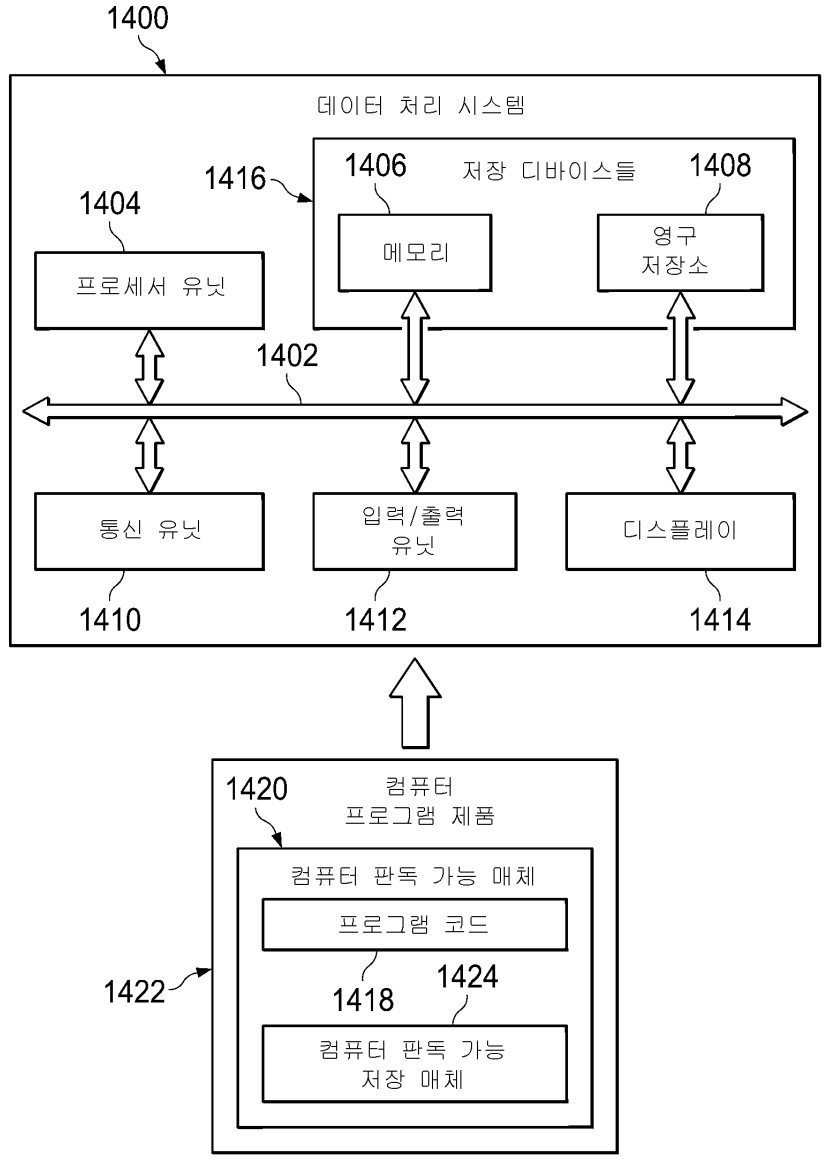
도면12


시작
1200
메시지의 도착을 검출함
1202
메시지를 프로세서 유닛에 대한 메모리에 푸시함
1204
대기열 깊이를 증분함
종료

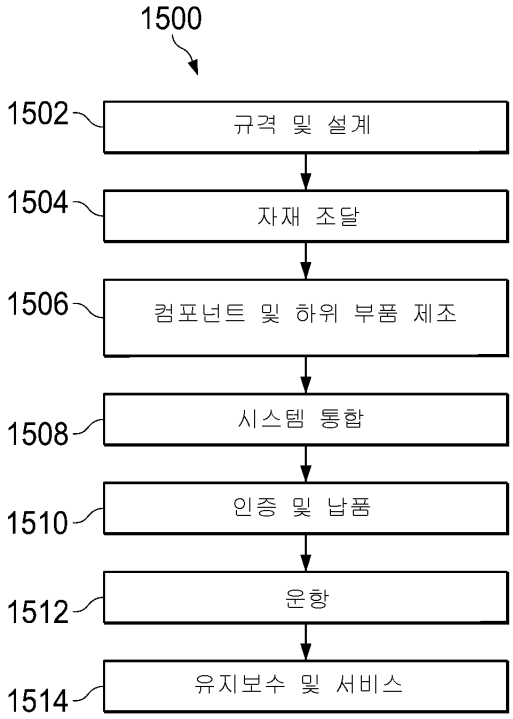
도면13


시작
1300
제1 프로세서 유닛으로부터 메시지들을 판독하라는 요청을 수신함
1302
제2 프로세서 유닛에 대한 제2 대기열이 이미 판독되었나?
예
1304
제2 프로세서 유닛에 대한 대기열에 대해 제2 판독 포인터를 제공함
아니오
1310
제2 레인에 대한 푸시 카운터가 제1 레인에 대한 푸시 카운터보다 더 큰가?
아니오
예
제2 레인에 대한 푸시 카운터를 판독 포인터로서 사용함
1314
1312
제1 푸시 카운터의 값을 메시지들을 판독하기 위한 판독 포인트로서 사용함
1306
판독 포인터까지 다수의 메시지들을 판독함
1308
푸시 카운터를 판독된 메시지들의 수만큼 감소시킴
종료

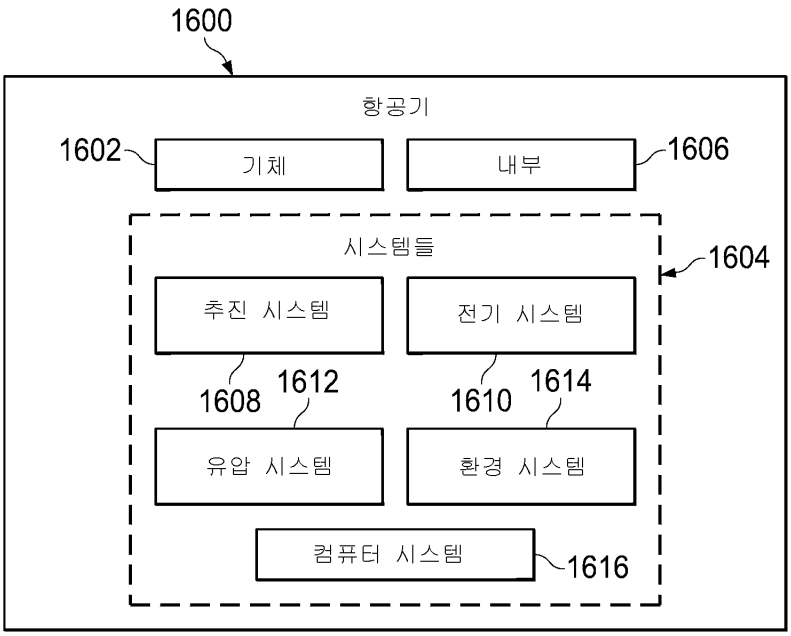
도면14


1400
데이터 처리 시스템
1406
저장 디바이스들
1408
1416
1404
메모리
영구
저장소
프로세서 유닛
1402
통신 유닛
입력/출력
유닛
디스플레이
1410
1412
1414
1420
컴퓨터
프로그램 제품
컴퓨터 판독 가능 매체
프로그램 코드
1422
1418
1424
컴퓨터 판독 가능
저장 매체

도면15


1500
1502
규격 및 설계
1504
자재 조달
1506
컴포넌트 및 하위 부품 제조
1508
시스템 통합
1510
인증 및 납품
1512
운항
1514
유지보수 및 서비스

도면16


1600
항공기
1602
기체
내부
1606
시스템들
1604
추진 시스템
전기 시스템
1608
1612
1610
1614
유압 시스템
환경 시스템
컴퓨터 시스템
1616