(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2019년07월17일
(11) 등록번호 10-2000957
(24) 등록일자 2019년07월11일

(51) 국제특허분류(Int. Cl.)
G06F 11/22 (2017.01)
(21) 출원번호 10-2014-7011055
(22) 출원일자(국제) 2012년09월20일
심사청구일자 2017년09월07일
(85) 번역문제출일자 2014년04월24일
(65) 공개번호 10-2014-0084068
(43) 공개일자 2014년07월04일
(86) 국제출원번호 PCT/US2012/056250
(87) 국제공개번호 WO 2013/062693
국제공개일자 2013년05월02일
(30) 우선권주장
13/284,491 2011년10월28일 미국(US)
(56) 선행기술조사문헌
KR1020070020247 A*
US07536679 B1
US20090217311 A1*
US20040181763 A1
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
테라다인 인코퍼레이티드
미국 01864 매사추세츠주 노스 리딩 엔알700-2-3 리버파크 드라이브 600
(72) 발명자
프릭 로이드 케이.
미국 매사추세츠 01463 페퍼렐 피.오.박스 111
린드 데이비드 존
미국 매사추세츠 01845 노스 앤도버 윈터 스트리트 575
(74) 대리인
특허법인와이에스장

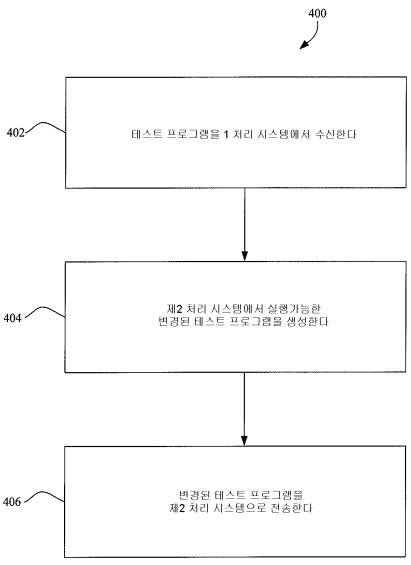
전체 청구항 수 : 총 17 항 심사관 : 김계준

(54) 발명의 명칭 **프로그램가능한 테스트 기기**

(57) 요 약

일반적으로, 테스트 기기는 테스트 기기에 인터페이스된 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램가능하고, 상기 테스트 기기의 동작을 제어하도록 프로그램가능한 제1 처리 시스템 및 장치 검사 전용인 제2 처리 시스템을 포함한다. 상기 제2 처리 장치는 상기 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램가능하고, 상기 제1 처리 시스템은 제1 애플리케이션 프로그래밍 인터페이스(API)를 갖고 있고 상기 제2 처리 시스템은 제2 API를 갖고 있고, 상기 제1 API 및 제2 API는 상이한 API이고, 상기 제1 API 및 제2 API는 적어도 일부 중복되는 함수를 갖고 있다.

대 표 도 - 도4



명 세 서

청구범위

청구항 1

테스트 기기에 인터페이스된 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램가능하고, 상기 테스트 기기의 동작을 제어하도록 프로그램가능한 제1 처리 시스템; 및

장치 검사 전용이고, 상기 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램가능한 제2 처리 시스템을 포함하고,

상기 제1 처리 시스템은 제1 애플리케이션 프로그래밍 인터페이스(API)를 갖고 있고 상기 제2 처리 시스템은 제2 API를 갖고 있고, 상기 제1 API 및 제2 API는 상이한 API이고, 상기 제1 API 및 제2 API는 적어도 일부 중복되는 함수를 갖고 있으며,

상기 제1 처리 시스템은 상기 제2 처리 시스템 상에서 동작 가능한 테스트 프로그램을 생성하기 위한 변환 엔진을 포함하고, 상기 변환 엔진은 상기 제2 API를 갖는 제2 처리 시스템 상에서 동작 가능한 변경된 테스트 프로그램을 생성하기 위하여 상기 제1 API를 갖는 상기 제1 처리 시스템 상에서 동작 가능한 테스트 프로그램을 변경함으로써 상기 테스트 프로그램을 생성하는 것을 특징으로 하는 테스트 기기.

청구항 2

제1항에 있어서, 상기 제1 API의 함수는 상기 제2 API의 대응 함수와 연관되어 있는 것을 특징으로 하는 테스트 기기.

청구항 3

제2항에 있어서, 상기 제1 API의 함수는 상기 제1 처리 시스템에 의해 수신된 정보에 응답하여 상기 제2 API의 함수를 호출하도록 구성된 것을 특징으로 하는 테스트 기기.

청구항 4

제3항에 있어서, 상기 정보는 상기 제1 API의 함수를 사용하여 기록된 프로그래밍 코드를 포함하고, 상기 제1 API의 함수는 상기 프로그래밍 코드를 상기 제2 처리 시스템에서 실행되도록 구성하기 위해 상기 제2 API의 함수를 호출하는 것을 특징으로 하는 테스트 기기.

청구항 5

제4항에 있어서, 상기 프로그래밍 코드를 상기 제2 처리 시스템에서 실행되도록 구성하는 단계는 상기 제1 API의 대응 함수의 사용에 응답하여 상기 제2 API와 연관된 함수를 사용하는 코드를 생성하는 단계를 포함하는 것을 특징으로 하는 테스트 기기.

청구항 6

제1항에 있어서, 상기 제1 처리 시스템은 상기 제1 처리 시스템과 인터페이스하고 상기 제1 API 및 제2 API를 통해 상기 제2 처리 시스템과 인터페이스하기 위한 사용자 인터페이스를 생성하도록 프로그램가능한 것을 특징으로 하는 테스트 기기.

청구항 7

제1항에 있어서, 상기 제1 처리 시스템은 상기 제2 API를 통해 상기 제2 처리 시스템과 인터페이스하기 위한 사용자 인터페이스를 생성하도록 프로그램가능한 것을 특징으로 하는 테스트 기기.

청구항 8

제1항에 있어서, 상기 제1 처리 시스템은 상기 테스트 기기에 인터페이스된 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램되어 있고,

상기 제2 처리 시스템은 상기 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램되어 있지 않은 것을 특징으로 하는 테스트 기기.

청구항 9

제1항에 있어서, 상기 제1 처리 시스템은 상기 테스트 기기에 인터페이스된 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램되어 있지 않고,

상기 제2 처리 시스템은 상기 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램된 것을 특징으로 하는 테스트 기기.

청구항 10

제1항에 있어서, 상기 장치에 결합하도록 구성된 하나 이상의 포트를 더 포함하고,

상기 제1 처리 시스템이 상기 하나 이상의 포트를 통해 상기 장치에 테스트 데이터를 전송하도록 구성되어 있거나,

상기 제2 처리 시스템이 상기 하나 이상의 포트를 통해 상기 장치에 테스트 데이터를 전송하도록 구성되어 있는 것을 특징으로 하는 테스트 기기.

청구항 11

제1항에 있어서, 상기 제1 처리 시스템은 범용 운영체제를 실행하도록 프로그램된 하나 이상의 마이크로프로세서를 포함하고,

상기 제2 처리 시스템은 상기 테스트 기기에 인터페이스된 상응하는 장치를 검사하도록 각각 프로그램된 하나 이상의 내장형 마이크로프로세서를 포함하는 것을 특징으로 하는 테스트 기기.

청구항 12

테스트 기기에서 실행되는 방법으로서,

테스트 기기에 인터페이스된 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램가능하고, 상기 테스트 기기의 동작을 제어하도록 프로그램된 제1 처리 시스템에서 테스트 프로그램을 수신하는 단계;

장치 검사 전용이고 상기 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램가능한 제2 처리 시스템에서 실행가능한 변경된 테스트 프로그램을 상기 테스트 프로그램에 기초하여 제1 처리 시스템 상에서 생성하는 단계; 및

상기 변경된 테스트 프로그램을 상기 제2 처리 시스템에 전송하는 단계를 포함하고,

상기 제1 처리 시스템은 제1 애플리케이션 프로그래밍 인터페이스(API)를 가지며, 상기 제2 처리 시스템은 제2 API를 가지며, 상기 제1 API 및 상기 제2 API는 서로 상이한 API이고, 상기 제1 API 및 상기 제2 API는 적어도 일부 중복되는 함수를 갖고 있으며; 및

상기 제1 처리 시스템은 상기 제1 처리 시스템상에서 실행 가능한 테스트 프로그램을 변경하여 제2 처리 시스템 상에서 실행 가능한 변경된 테스트 프로그램을 생성하는 변환 엔진을 포함하는 것을 특징으로 하는 테스트 기기에서 실행되는 방법.

청구항 13

제12항에 있어서, 상기 테스트 프로그램은 상기 제1 API의 함수에 대한 하나 이상의 호출을 포함하는 것을 특징으로 하는 테스트 기기에서 실행되는 방법.

청구항 14

제13항에 있어서, 상기 변경된 테스트 프로그램을 생성하는 단계는 상기 제1 API의 하나 이상의 함수에 상응하는 상기 제2 API의 하나 이상의 대응 함수를 식별하는 단계를 포함하는 것을 특징으로 하는 테스트 기기에서 실행되는 방법.

청구항 15

제14항에 있어서, 상기 변경된 테스트 프로그램을 생성하는 단계는 상기 제1 API의 함수에 대한 하나 이상의 호출을 상기 제2 API의 대응 함수에 대한 하나 이상의 호출로 대체하는 단계를 포함하는 것을 특징으로 하는 테스트 기기에서 실행되는 방법.

청구항 16

제12항에 있어서, 상기 장치를 검사하기 위해 상기 제2 처리 시스템에서 상기 변경된 테스트 프로그램을 실행하는 단계를 더 포함하는 것을 특징으로 하는 테스트 기기에서 실행되는 방법.

청구항 17

제12항에 있어서, 상기 변경된 테스트 프로그램이 상기 제2 처리 시스템에서 수정될 수 있도록 하는 상기 제2 처리 시스템과 연관된 개발 환경을 제공하는 단계를 더 포함하는 것을 특징으로 하는 테스트 기기에서 실행되는 방법.

청구항 18

삭제

발명의 설명

기 술 분 야

[0001] 본 발명은 일반적으로 프로그램가능한 테스트 기기에 관한 것이다.

배 경 기 술

[0002] 자동 테스트 기기(ATE)는 반도체 장치 및 회로 기판 어셈블리와 같은 전자장치를 제조하는데 역할을 담당한다. 제조자는 일반적으로 제조 공정 동안 장치의 동작을 검증하기 위해 자동 테스트 기기, 또는 "테스터 기기"를 사용한다. 이러한 장치는 "피검사 장치"(DUT) 또는 "피검사 유닛"(UUT)으로 불린다. 고장의 조기 검출은 결함 장치를 처리함으로써 발생될 수 있는 비용을 제거하여 전체 제조 비용을 절감할 수 있다. 제조자는 또한 다양한 사양에 등급을 부여하기 위해 ATE를 사용한다. 장치는 검사되고 속도와 같은 영역에서의 상이한 성능 레벨에 따라 분류되어질 수 있다. 장치는 실제 성능 레벨에 따라 라벨 부여되고 판매될 수 있다.

발명의 내용

해결하려는 과제

과제의 해결 수단

[0003] 일반적으로, 하나의 특징에서, 테스트 기기는 테스트 기기에 인터페이스된 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램가능하고, 상기 테스트 기기의 동작을 제어하도록 프로그램가능한 제1 처리 시스템 및 장치 검사 전용인 제2 처리 시스템을 포함한다. 상기 제2 처리 장치는 상기 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램가능하고, 상기 제1 처리 시스템은 제1 애플리케이션 프로그래밍 인터페이스(API)를 갖고 있고 상기 제2 처리 시스템은 제2 API를 갖고 있고, 상기 제1 API 및 제2 API는 상이한 API이고, 상기 제1 API 및 제2 API는 적어도 일부 중복되는 함수를 갖고 있다.

[0004] 일반적으로 다른 특징에서, 테스트 기기에서 실행되는 방법은 테스트 기기에 인터페이스된 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램가능하고, 상기 테스트 기기의 동작을 제어하도록 프로그램된 제1 처리 시스템에서 테스트 프로그램을 수신하는 단계; 장치 검사 전용이고 상기 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램가능한 제2 처리 시스템에서 실행가능한 변경된 테스트 프로그램을 상기 테스트 프로그램에 기초하여 생성하는 단계; 및 상기 변경된 테스트 프로그램을 상기 제2 처리 시스템에 전송하는 단계를 포함한다.

[0005] 이러한 특징은 하나 이상의 다음의 특징을 포함할 수 있다. 상기 제1 API의 함수는 상기 제2 API의 대응 함수와 연관되어 있다. 상기 제1 API의 함수는 상기 제1 처리 시스템에 의해 수신된 정보에 응답하여 상기 제2 API의 함수를 호출하도록 구성되어 있다. 상기 정보는 상기 제1 API의 함수를 사용하여 기록된 프로그래밍 코드를 포함하고, 상기 제1 API의 함수는 상기 프로그래밍 코드를 상기 제2 처리 시스템에서 실행되도록 구성하기 위해 상기 제2 API의 함수를 호출한다. 상기 제2 처리 시스템에서 실행되도록 상기 프로그래밍 코드를 구성하는 단계는 상기 제1 API의 대응 함수의 사용에 응답하여 상기 제2 API와 연관된 함수를 사용하는 코드를 생성하는 단계를 포함한다. 상기 제1 처리 시스템은 상기 제1 처리 시스템과 인터페이스하고 상기 제1 API 및 제2 API를 통해 상기 제2 처리 시스템과 인터페이스하기 위한 사용자 인터페이스를 생성하도록 프로그램가능하다.

[0006] 상기 제1 처리 시스템은 상기 제2 API를 통해 상기 제2 처리 시스템과 인터페이스하기 위한 사용자 인터페이스를 생성하도록 프로그램가능하다. 상기 제1 처리 시스템은 상기 테스트 기기에 인터페이스된 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램되어 있고, 상기 제2 처리 시스템은 상기 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램되어 있지 않다. 상기 제1 처리 시스템은 상기 테스트 기기에 인터페이스된 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램되어 있지 않고, 상기 제2 처리 시스템은 상기 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램되어 있다. 상기 테스트 기기는 상기 장치에 결합하도록 구성된 하나 이상의 포트를 더 포함하고, 상기 제1 처리 시스템이 상기 하나 이상의 포트를 통해 상기 장치에 테스트 데이터를 전송하도록 구성되어 있거나, 상기 제2 처리 시스템이 상기 하나 이상의 포트를 통해 상기 장치에 테스트 데이터를 전송하도록 구성되어 있다.

[0007] 상기 제1 처리 시스템은 제1 애플리케이션 프로그래밍 인터페이스(API)를 갖고 있고 상기 제2 처리 시스템은 제2 API를 갖고 있고, 상기 제1 API 및 제2 API는 상이한 API이고, 상기 제1 API 및 제2 API는 적어도 일부 중복되는 함수를 갖고 있다. 상기 테스트 프로그램은 상기 제1 API의 함수에 대한 하나 이상의 호출을 포함한다. 상기 변경된 테스트 프로그램을 생성하는 단계는 상기 제1 API의 하나 이상의 함수에 상응하는 상기 제2 API의 하나 이상의 대응 함수를 식별하는 단계를 포함한다. 상기 변경된 테스트 프로그램을 생성하는 단계는 상기 제1 API의 함수에 대한 하나 이상의 호출을 상기 제2 API의 대응 함수에 대한 하나 이상의 호출로 대체하는 단계를 포함한다. 상기 변경된 테스트 프로그램은 상기 장치를 검사하기 위해 상기 제2 처리 시스템에서 실행된다. 상기 변경된 테스트 프로그램이 상기 제2 처리 시스템에서 수정될 수 있도록 하는, 상기 제2 처리 시스템과 연관된 개발 환경이 제공되어 있다.

[0008] 이러한 요약부를 포함하는, 본원에 기술된 2개 이상의 특징은 여기에 구체적으로 기술되지 않은 실시예를 형성하도록 조합될 수 있다.

[0009] 여기에 기술된 시스템 및 기술 또는 그 일부는 하나 이상의 비일시적인 기계 판독가능 저장 매체에 저장되고 하나 이상의 처리 장치에서 실행가능한 명령어를 포함하는 컴퓨터 프로그램 제품으로서 구현될 수 있다. 여기에 기술된 시스템 및 기술, 또는 그 일부는 장치, 방법, 또는 하나 이상의 처리 장치 및 언급된 기능을 구현하기 위해 실행가능한 명령어를 저장하는 메모리를 포함할 수 있는 전자 시스템으로서 구현될 수 있다.

[0010] 하나 이상의 구현예는 첨부된 도면 및 아래의 설명에서 상세하게 설명된다. 다른 특징, 목적, 및 장점이 설명 및 도면, 그리고 청구범위로부터 명백해질 것이다.

도면의 간단한 설명

[0011] 도 1은 테스트 기기 예의 블록도이다.

도 2는 도 1의 테스트 기기 예로 통합될 수 있는 특징부의 예를 도시하는 블록도이다.

도 3은 테스트 시스템 예의 블록도이다.

도 4는 테스트 시스템에 포함된 테스터 예의 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0012] 다수의 처리층(때로 "계층"으로 부른다)을 포함할 수 있는 테스트 기기가 여기에 설명되어 있다. 이러한 다수의 처리층은 함수 라이브러리를 포함하는 각 애플리케이션 프로그래밍 인터페이스(API)를 포함할 수 있다. 이러한 함수 라이브러리는 적어도 하나의 레벨에서 테스트 기기에 제공된 테스트 프로그램 또는 테스트 로직과 같은 프로그램에 의해 호출될 수 있는 다양한 함수를 저장할 수 있다. 일부 예에서, 제1 층은 테스트 기기에 인

터페이스되는 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램될 수 있고, 테스트 기기의 동작을 제어하도록 프로그램될 수 있다. 동일한 테스트 기기의 제2 층은 장치 검사 전용일 수 있고, 이러한 장치를 검사하도록 하나 이상의 테스트 프로그램을 실행할 수 있다.

[0013] 제1 층은 제1 API를 포함할 수 있고 제2 층은 제2 API를 포함할 수 있고, 제1 API 및 제2 API는 어떤 면에서는 서로 상이할 수 있다. 테스트 프로그램이 제1 층에서 생성되고 제2 층에서 실행될 수 있도록 하기 위해, 제1 API 및 제2 API는 적어도 일부 중복되는 함수를 포함할 수 있다(예를 들어, 제1 API의 함수는 제2 API 함수와 상응할 수 있다). 따라서, 제1 층에서 생성된 테스트 프로그램(예를 들어, 제1 API의 함수를 포함하거나 호출하는 테스트 프로그램)이 제2 층의 처리 장치와 호환가능하도록 제조될 수 있고, 제1 API의 함수의 적어도 일부는 제2 API와 연관된 상대 함수를 갖고 있다.

[0014] 일부 예에서, 제1 층은 제1 API의 함수와 제2 API의 대응 함수 사이의 연관성(association) 또는 맵핑을 저장할 수 있다. 이러한 저장된 연관성 및 맵핑은 제1 층에서 생성된 테스트 프로그램을 (예를 들어, 테스트 프로그램에서의 제1 API의 함수를 제1 API의 함수와 상응하도록 대체하거나 수정함으로써) 제2 층에서 실행되기에 적합한 형태로 변경하거나 변환하는데 사용될 수 있다. 그 다음, 이러한 변경된 테스트 프로그램은 실행을 위해 제2 층에 전송될 수 있다.

[0015] 도 1은 상기 테스트 기기(100)의 구현예의 블록도이다. 도 1에서, 테스트 기기(100)는 3 계층 처리 시스템을 포함하고 있다. 그러나, 다른 구현예에서, 보다 많거나 적은 수의 계층이 존재할 수 있다. 테스트 기기(100)의 상이한 계층은 DUT로의 계층의 상대적인 관계를 반영한다. 이러한 예에서, 제1 계층(101)은 컴퓨터(102)를 포함하고 있다. 컴퓨터(102)는 외부 네트워크와의 통신과 같은, 테스트 기기(100)의 다양한 특징을 제어한다. 또한, 컴퓨터(102)는 하술된 바와 같이, 다양한 검사 동작을 수행하도록 프로그램가능하다. 제2 계층(104)은 테스트 전용의 하나 이상의 처리 장치(106 내지 108)를 포함하고 있다. 예를 들어, 처리 장치(106 내지 108)는 보통 테스트 기기 제어 및 네트워크 통신과 같은 논-테스트 기능을 수행하지 않는다. 제3 계층(110)은 DUT(115)로의 인터페이스와 같이 동작하고 DUT에 하나 이상의 검사 동작을 수행하도록 프로그램가능한 로직(111 내지 113)을 포함하고 있다.

[0016] 이러한 제1 계층(101)의 예에서, 컴퓨터(102)는 하나 이상의 마이크로프로세서 또는 단일 멀티코어 마이크로프로세서(도시되지 않음)와 같은 하나 이상의 처리 장치를 포함하고 있다. 컴퓨터(102)는 외부 환경과의 테스트 기기 통신을 제어하고 테스트 기기(100)의 동작을 제어하기 위해 다양한 "하우스키핑" 기능을 수행하도록 실행가능한 코드를 저장하는 메모리(도시되지 않음)를 포함하고 있다. 예를 들어, 컴퓨터(102)는 네트워크 인터페이스를 통해 테스트 기기와 하나 이상의 외부 엔티티 사이의 통신을 교환하고, 악성코드, 메모리 관리, 전력 제어 및, DUT를 검사하는 것과 구체적으로 관련되어 있지 않은 다른 기능을 위해 테스트 기기를 스캔하는 기능을 담당할 수 있다.

[0017] 컴퓨터(102)는 또한 테스트 기기(100)에 인터페이스된 DUT(예를 들어, 115)에 검사 동작을 수행하도록 프로그램가능하다. 이러한 검사 동작은 버스 속도, 반응 시간, 또는 DUT의 임의의 다른 적합한 동작 특성을 검사하는 것을 포함하지만 이제 제한되는 것은 아니다. 일반적으로, 수행되는 검사는 검사되는 장치의 타입 및, 검사 동안 구해진 정보에 의존한다.

[0018] 하나 이상의 테스트 프로그램이 검사를 수행하기 위해 컴퓨터(102)의 메모리에 로딩될 수 있고 컴퓨터(102)의 처리 장치에 의해 실행될 수 있다. 검사를 수행하는 동안, 컴퓨터(102)는 테스트 기기(100)가 지속 동작하도록, 상술된 바와 같은, 다른 기능을 계속 수행할 수 있다. 따라서, 테스트 반응시간(latency)(예를 들어, 테스트의 시작과 테스트 결과의 수신 사이의 시간량)은 수 밀리초 정도일 수 있다. 또한, 타이밍 가변성 역시 테스트의 다른 실행이 아닌 테스트의 하나의 실행 동안 시스템 기능(예를 들어, 바이러스 스캐닝)을 수행하는 컴퓨터(102)에 의해 영향을 받을 수 있다. 이로 인해 바이러스 스캔이 실행할 때와 비교하여 바이러스 스캔이 실행되지 않을 때 시간상 보다 가깝게 테스트 시퀀스가 연속으로 수행될 수도 있다. 이것은 단지 테스트 반응시간의 예일 뿐이다. 상이한 시스템에서, 컴퓨터(102)의 처리 장치의 속도, 테스트 프로그램을 수행하기 위해 사용가능한 컴퓨터(102)의 메모리 양, 처리 장치의 주의 분할등과 같은 수많은 요인이 테스트 반응시간에 영향을 줄 수 있다.

[0019] 컴퓨터(102)를 통해 테스트를 수행하는데 있어서 가능한 이점은 테스트 프로그램의 개발 비용과 관련되어 있다. 보다 구체적으로, 컴퓨터(102)는 윈도우잉, 또는 다른 비교적 사용자 친화적인 운영 시스템을 실행할 수 있다. 이러한 운영 시스템에서의 테스트 프로그램의 개발에 유용한 툴은 보통 널리 사용가능하고, 일반적으로 테스트 프로그램 개발자에게 친숙하다. 그래서, 컴퓨터(102)에서 실행되기 위한, 컴퓨터(102)에서의 테스트 프로그램

의 개발 비용은 다층 구조의 다른 계층에서 실행되는 테스트 프로그램의 개발 비용보다 적을 수 있다. 그러나, 이러한 일반화는 모든 경우에 적용되지 않을 수 있다.

[0020] 이러한 예에서, 제2 계층(104)은 다수의 내장형 처리 장치(106 내지 108)를 포함하고 있다. 여기에서, 3개의 내장형 처리 장치가 도시되어 있지만, 테스트 기기(100)는 예를 들어, 하나, 둘, 넷, 다섯 이상의 임의의 적합한 수의 내장형 처리 장치를 포함할 수 있다. 이러한 처리 장치는 이들이 테스트 기기(100)내에 통합되어 있고 테스트 기능의 수행(예를 들어, 테스트 기기(100)에 인터페이스된 DUT 테스트)에 전용된다는 점에서 내장되어 있다. 내장형 처리 장치(106 내지 108)는 보통 컴퓨터(102)에 의해 수행되는 상술된 "하우스키핑" 동작과 같은 테스트 기기 동작의 기능을 담당하지 않는다. 그러나, 일부 실시예에서, 내장형 처리 장치(106 내지 108)는 하나 이상의 이러한 동작, 또는 구체적으로 DUT 테스트가 아닌 다른 동작을 수행하도록 프로그램될 수 있다.

[0021] 각 내장형 처리 장치(106 내지 108)는 예를 들어, 단일 코어 또는 다수의 코어를 갖는 마이크로컨트롤러 또는 마이크로프로세서를 포함할 수 있다. 각 마이크로프로세서는 컴퓨터(102)를 통해 또는 직접적으로 프로그램가능하다. 예를 들어, 테스트 기기(100)의 사용자는 내장형 처리 장치(106)를 프로그램하기 위해 컴퓨터(102)의 운영 시스템과 상호작용할 수 있다. 대안으로, 그를 통해 각 내장형 처리 장치가 프로그램될 수 있는 다이렉트 인터페이스, 예를 들어, 하드웨어 또는 소프트웨어가 존재할 수 있다. 이러한 맥락에서, 프로그램한다는 것은 DUT를 검사하기 위해, 내장형 처리 장치에서 실행될 수 있는 하나 이상의 테스트 프로그램을 각 내장형 처리 장치에 저장하는 것을 가리킨다.

[0022] 도 1에 도시된 바와 같이, 각 내장형 처리 장치는 컴퓨터(102)에 그리고 각 프로그램가능한 로직(이러한 예에서, 필드 프로그램가능 게이트 어레이(FPGA))에 인터페이스되어 있다. 하술되는 바와 같이, 각 FPGA는 검사될 개별적인 DUT(도시되지 않음) 또는 단일 DUT의 일부(예를 들어, 도시된 바와 같은 DUT 상의 버스(122, 123, 124))로의 인터페이스로서 동작한다. 따라서, 이러한 예에서, 각 내장형 처리 장치는 검사되는 상응하는 DUT, 또는 그 일부를 위해 특정하게 설계된 테스트 프로그램에 의해 프로그램될 수 있다. 언급된 바와 같이, 적합한 테스트 프로그램이 이러한 내장형 처리 장치에 직접 로딩될 수 있거나 컴퓨터(102)를 통해 로딩될 수 있다. 각 내장형 처리 장치는 자체 테스트 프로그램을 별개로 실행할 수 있고, 다른 내장형 처리 장치와 동시에 실행할 수 있다. 일부 실시예에서, 내장형 처리 장치의 각 테스트 프로그램이 어떻게 실행될 지에 대해 내장형 처리 장치 사이에 조정(coordination)이 있을 수 있다. 이러한 조정은 내장형 처리 장치 자체에 의해 또는 컴퓨터(102)에 의해 구현될 수 있다. 일부 실시예에서, 이러한 조정은 이러한 구조의 상이한 계층에서의 장치와 관련될 수 있다. 일부 실시예에서, 상이한 내장형 처리 장치(106 내지 108)는 적합한 조정으로 또는 적합한 조정 없이 동일한 테스트 프로그램의 상이한 부분(예를 들어, 모듈)을 구현할 수 있다.

[0023] 내장형 처리 장치를 통한 테스트 수행하는데 있어 가능한 이점은 테스트 반응시간 및 타이밍 가변성과 관련되어 있다. 보다 구체적으로, 내장형 처리 장치가 테스트 전용이기 때문에, 그들 자원은 보통 다른 태스크에 의한 부담이 없다. 그래서, 테스트 반응시간은 컴퓨터(102)에 의해 달성되는 것보다 적을 수 있고, 타이밍 가변성은 감소될 수 있다. 예를 들어, 내장형 처리 장치를 위한 테스트 반응시간은 수 마이크로초 정도일 수 있다. 그러나, 이것은 단지 내장형 처리 장치 테스트 반응시간의 한 예일 뿐이다. 상이한 시스템에서, 처리 장치 속도, 테스트 프로그램을 실행하는데 사용가능한 메모리 양과 같은 다수의 요인이 테스트 반응시간에 영향을 줄 수 있다. 따라서, 상술된 일반화는 모든 경우에 적용될 수 없다.

[0024] 또한, 내장형 처리 장치에서의 테스트 프로그램의 개발을 위한 툴이 사용가능하다. 그래서, 내장형 처리 장치에서 수행되기 위한, 내장형 처리 장치용 테스트 프로그램의 개발 비용이 FPGA와 같은, 하드웨어에서 수행되기 위한 테스트 프로그램의 개발 비용 보다 적을 수 있다. 일부 예에서, 비용 문제는 다양한 계층에서 프로그램에 필요한 스킬 세트의 공통성 및 각 개발 환경의 성숙과 관련될 수 있다. 시스템 계층에 비해 내장형 계층을 위한 프로그래밍은 보다 향상된 SW 스킬을 필요로 할 수 있다. 이러한 스킬은 보통 덜 공통적일 수 있고 개발/아웃소싱하기에 보다 더 고가일 수 있다. 시스템 계층 개발 툴은 내장된 계층을 위한 상응하는 툴에 비해 매우 개선되고 강력하다. 이것은 시스템 계층을 위한 개발 비용을 감소시킬 수 있다 (모든 다른 것들은 동일하다). 또한, 내장형 처리 장치에서 테스트 프로그램을 디버깅(debugging)하는 비용은 유사한 이유로 시스템 처리 장치에서 이러한 테스트를 디버깅하는 비용보다 높을 수 있다. 시스템 처리 장치에서 유용한 디버깅 툴은 내장형 처리 장치에서 유용한 유사한 툴 보다 저렴하고, 보다 강력하고 보다 익숙할 수 있다.

[0025] 제3 계층(110)은 프로그램가능한 로직, 예를 들어, FPGA(111 내지 113)을 포함하지만 다른 타입의 프로그램가능한 로직이 FPGA 대신에 사용될 수 있다. 각 FPGA는 이러한 FPGA에 프로그램 이미지를 로딩함으로써 구성된다. 이러한 프로그램 이미지는 "FPGA 로드(FPGA load)"로 부른다. 이러한 예에서, 각 FPGA는 DUT 또는 그 일부(예

를 들어, DUT 버스)와 테스트 기기(100) 사이의 인터페이스로서 동작하도록 구성되어 있다. 예를 들어, FPGA는 포트 폭, 포트 속도(예를 들어, 10MHz 내지 40MHz), 입력 포트의 수, 출력 포트의 수등을 명시할 수 있다.

[0026] 제1 계층(101) 컴퓨팅 장치(예를 들어, 컴퓨터(102)) 및 제2 계층(104) 컴퓨팅 장치(예를 들어, 내장형 처리 장치(106 내지 108))는 제3 계층(110)을 통해 DUT(115)에 접근한다. 예를 들어, 도 1에 도시된 바와 같이, 각 내장형 처리 장치는 상응하는 FPGA를 통해 DUT(115)와 통신할 수 있다. 컴퓨터(102)는 어느 DUT, 또는 DUT의 일부가 현재 검사되고 있는지에 따라 하나 이상의 FPGA를 통해 DUT(115)와 통신할 수 있다. 일부 실시예에서, FPGA에 의해 구현되는 각 인터페이스는 프로그램가능하다. 다른 실시예에서, 각 FPGA에 의해 구현된 인터페이스는 정적이다(예를 들어, 프로그램가능하지 않다).

[0027] 각 FPGA 역시 FPGA가 인터페이스하는 상응하는 DUT 또는 그 일부에 하나 이상의 테스트를 수행하도록 구성가능할 수 있다. 예를 들어, 각 FPGA에 대한 FPGA 로드는 DUT의 다양한 특성을 검사하기 위해 FPGA에 의해 실행되는 하나 이상의 테스트 루틴을 포함할 수 있다. 상술된 바와 같이, 구현되는 루틴은 검사되는 장치 및 테스트 동안 구해지는 정보에 의존한다. 각 FPGA에 의해 실행되는 테스트 루틴은 다른 FPGA에 의해 실행되는 다른 테스트 루틴과는 독립적으로 실행될 수 있거나 다양한 FPGA 사이에 조정이 있을 수 있다. 각 FPGA는 별개로 그리고 다른 내장형 처리 장치와 동시에 자체 테스트 루틴을 실행할 수 있다. 일부 실시예에서, FPGA의 각각의 테스트 프로그램이 어떻게 실행될지에 대해 FPGA 사이에 조정이 있을 수 있다. 이러한 조정은 FPGA 자체에 의해, 이들의 상응하는 내장형 처리 장치에 의해 또는, 컴퓨터(102)에 의해 구현될 수 있다. 일부 실시예에서, 이러한 조정은 이러한 구조의 상이한 계층에서의 장치와 관련될 수 있다. 예를 들어, 컴퓨터(102)는 내장형 처리 장치(106 내지 108)과 협력하여 각 FPGA(111 내지 113)의 동작을 조정할 수 있다. 일부 실시예에서, 상이한 FPGA는 적합한 조정에 의해, 또는 적합한 조정 없이 동일한 테스트 루틴의 상이한 부분(예를 들어, 모듈)을 구현할 수 있다.

[0028] FPGA를 통해 테스트를 수행하는데 있어서 가능한 이점은 테스트 반응시간 및 타이밍 가변성의 감소와 관련되어 있다. 보다 구체적으로, FPGA가 하드웨어 장치이기 때문에, 이들은 내장형 처리 장치(106 내지 108) 또는 컴퓨터(102)에 프로그램된 테스트 루틴 보다는 더 높은 속도로 실행될 수 있다. 그래서, 테스팅 반응시간은 내장형 처리 장치(106 내지 108) 또는 컴퓨터(102)에 의해 달성되는 것 보다 적을 수 있다. 예를 들어, 내장형 처리 장치에 대한 테스트 반응시간은 수 나노초 정도일 수 있다. 그러나, 이것은 단지 FPGA 테스트 반응시간의 한 예일 뿐이다. 상이한 시스템에서, 수많은 요인이 테스트 반응시간에 영향을 줄 수 있다. 이에 따라, 상기 일반화는 모든 경우에 적용될 수 없다.

[0029] 일부 실시예에서, 테스팅은 이러한 구조의 하나의 계층 또는 다른 계층에 의해 배타적으로 수행될 수 있다. 예를 들어, 컴퓨터(102)는 DUT를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램될 수 있지만 이러한 구조의 다른 계층의 장치는 DUT 테스트를 수행하지 않는다. 내장형 처리 장치(106 내지 108)는 DUT를 검사하기 위해 하나 이상의 테스트 프로그램을 수행하도록 프로그램될 수 있지만, 이러한 구조의 다른 계층의 장치는 DUT 테스트를 수행하지 않는다. FPGA(111 내지 13)은 장치에 하나 이상의 테스트를 수행하도록 구성될 수 있지만 이러한 구조의 다른 계층의 장치는 DUT 테스트를 수행하지 않는다. 테스트를 수행하지 않는 장치는 반드시 이러한 시간 동안 휴지 상태일 필요는 없다. 예를 들어, 컴퓨터(102)는 상술된 하우스키핑 동작을 계속 수행할 수 있고, FPGA는 DUT로/로부터 데이터를 계속 전송할 수 있고(즉, DUT로의 인터페이스로서 동작할 수 있다), 내장형 처리 장치는 조정 또는 다른 통신(예를 들어, FPGA로부터 테스트 결과를 컴퓨터(102)에 전송하는 것)에 있어 계속 활성 상태를 나타낼 수 있다.

[0030] 다른 실시예에서, 테스팅은 이러한 구조의 상이한 계층에 의해 동시에 또는 합동하여 수행될 수 있다. 예를 들어, 2개 이상의 컴퓨터(102), 내장형 처리 장치(106 내지 108), 및 FPGA(111 내지 113)는 단일 DUT 또는 다수의 DUT에 하나 이상의 테스트 동작을 수행하기 위해, 합동하여, 동일한 시간에 또는 동일한 검사 시퀀스에서 동작할 수 있다. 이러한 합동을 위해, 적합한 프로그래밍이 컴퓨터(102) 및/또는 내장형 처리 장치(106 내지 108)에 로딩되고, 및/또는 적합한 이미지가 FPGA에 로딩된다. 예를 들어, 제1 테스트는 컴퓨터(102)에 의해 DUT에 수행될 수 있고, 제2 테스트는 내장형 처리 장치(106)에 의해 DUT에 수행될 수 있고, 제3 테스트는 FPGA(111)에 의해 DUT에 수행될 수 있다. 제1, 제2 및 제3 테스트는 별개의 테스트, 또는 동일한 검사 시퀀스의 일부일 수 있다. 제1, 제2 및 제3 테스트로부터의 데이터는 적합한 테스트 결과를 얻기 위해 예를 들어, 컴퓨터(102)에 조합되고 처리될 수 있다. 이러한 테스트 결과는 분석 및 보고를 위해 외부 컴퓨터(도시되지 않음)에 전송될 수 있다. 이러한 구조의 임의의 계층 또는 다른(예를 들어, 제3자의) 컴퓨터(도시되지 않음)가 조정을 수행할 수 있다.

[0031] 이러한 구조의 하나 이상의 계층이 프로그램되지 않은 실시예에서, 프로그램되지 않은 계층은 (적어도 이들의 테스트 기능에 관한 한) 우회될 수 있다. 이러한 프로그램되지 않은 계층은 프로그램화 및 계층 사이에 그리고 외부 네트워크와의 통신과 관련된 상술된 것과 같은 다양한 기능을 수행하도록 사전 프로그램되거나 사전 구성될 수 있다.

[0032] 다양한 계층에서의 장치는 실시간으로 프로그램되거나 구성될 수 있다. 여기에서, "실시간"은 테스트 시간에 또는 테스트 시간 직전에 프로그램하는 것을 포함한다. 즉, 테스트 기기는 DUT에 실행될 테스트 프로그램으로 사전프로그램될 필요는 없다. 이러한 테스트 프로그램은 적합한 시간에 기기에 통합될 수 있다. 테스트 기기에 존재하는 테스트 프로그램은 마찬가지로 적절한 대로 새로운 테스트 프로그램으로 대체될 수 있다.

[0033] 도 2는 다계층 구조를 포함하는 테스트 기기(200)를 도시하고 있다. 도 2의 예에서, 테스트 기기(200)는 시스템 층(202)(예를 들어, "제1 계층") 및 테스팅 층(204)(예를 들어, "제2 계층")을 포함하고 있다. 그러나, 다른 실시예에서, 보다 많거나 보다 적은 계층이 존재할 수 있다. 상술된 바와 같이, 시스템(200)의 상이한 계층은 DUT(도시되지 않음)로의 시스템 층(202) 및 테스팅 층(204)의 상대 관계를 반영한다. 이러한 예에서, 시스템 층(202)은 도 1에 도시된 제1 계층(101)에 유사한 기능을 제공할 수 있고 시스템 처리 장치(206)를 포함할 수 있다. 예를 들어, 시스템 처리 장치(206)는 외부 네트워크와의 통신 또는 (예를 들어, 이메일 프로그램 및 워드 프로세싱 프로그램과 같은, DUT를 검사하는 것과 관련되지 않은 프로그램을 제어하는) 다른 일반적인 컴퓨팅 태스크와 같은, 테스트 기기(200)의 다양한 특징을 제어할 수 있다. 또한, 시스템 처리 장치(206)는 하술되는 바와 같은 다양한 검사 동작을 수행하도록 프로그램될 수 있다.

[0034] 테스팅 층(204)은 디바이스 테스팅과 같은 하나 이상의 특정화된 태스크 전용의 내장형 처리 장치(208)를 포함하고 있다. 이러한 테스팅 층(204)이 (테스트 기기(100)(도 1)의 제2 계층(104)에 도시된 바와 같이) 하나 보다 많은 내장형 처리 장치를 포함할 수 있지만, 단순히하기 위해 오직 하나의 내장형 처리 장치가 도시되어 있다. 일부 예에서, 이러한 내장형 처리 장치(208)는 테스트 기기 제어 및 네트워크 통신과 같은 논-테스트 기능을 수행하지 않는다. 테스트 기기는 또한 하나 이상의 DUT와 교대로 통신할 수 있는, 도 1에 도시된 제3 계층(110)과 같은 하나 이상의 추가 층을 포함할 수 있다.

[0035] 시스템 층(202)은 또한 시스템 애플리케이션 프로그래밍 인터페이스(API)(210)를 포함한다. 일반적으로, API는 애플리케이션(예를 들어, 시스템 처리 장치(206) 및 내장형 처리 장치(208)중 하나 이상에서 실행되는 소프트웨어)에 접근하고 상호작용하기 위한 데이터 구조, 함수, 프로토콜, 루틴 및 툴의 세트이다. 애플리케이션 프로그래밍 인터페이스는 언어 종속 또는 언어 독립적일 수 있다. 언어 종속 API는 특정 프로그래밍 언어에서만 유용하고 언어 독립 API는 특정 언어, 시스템 또는 프로세스에 구속되지 않는다. 일부 예에서, 언어 독립 API는 하나 보다 많은 프로그래밍 언어로부터 접근될 수 있다(예를 들어, 이러한 프로그래밍 언어를 사용하여 언어 독립 API의 함수가 호출될 수 있다). 시스템 API(210)는 시스템 API(210)과 연관된 다수의 함수(예를 들어, 함수 1 및 함수 2)를 저장하는 시스템 디바이스 함수 라이브러리(212)를 포함하고 있다. 함수(예를 들어, 함수 1)은 호출될 때, 시스템 처리 장치(206)가 하나 이상의 동작을 실행하도록 할 수 있다. 예를 들어, 함수(예를 들어, 함수 1)는 처리 장치가 FPGA(예를 들어, 제3 계층(110)과 연관된 FPGA)와 연관된 레지스터에 접근하도록 할 수 있다. 다른 함수는 기기의 상태를 감시하도록 전원 또는 온도 센서(예를 들어, 기기 내측에 있다)에 접근할 수 있다.

[0036] 일부 예에서, 사용자(210)는 DUT를 검사하기 위해 테스트 기기(200)에 의해 사용될 수 있는 테스트 로직을 개발하기를 원할 수 있다. 시스템 층(202)은 시스템 디바이스 함수 라이브러리(212)를 포함하거나 접근하는 개발 환경을 제공할 수 있다. 일부 시스템 층이 보통 (예를 들어, 테스트 로직이 테스팅 층(204) 내의 내장형 처리 장치(208)에서 실행되고 함수 문법 및 유용성이 테스트 기기(200)의 층들 사이에서 다를 수 있기 때문에) 사용자가 테스트 로직을 생성하는 능력을 제공하는 개발 환경을 제공할 수 없지만, 여기에 기술된 기술에 의하면 사용자는 테스트 기기(200)에 접속된 DUT(예를 들어, 제3 계층(110)과 같은 제3 층에 접속된 DUT)를 검사하기 위해 테스팅 층(204)에서 실행될 수 있는 테스트 로직을 시스템 층(202)에서 생성할 수 있다.

[0037] 일부 예에서, 시스템 층(202)에서 테스트 로직을 생성할 수 있도록 하기 위해, 시스템 층(202) 및 테스팅 층(204)에 서로 대응하는 함수를 포함하는 각 API가 제공될 수 있다. 예를 들어, 내장된 장치 API(214)는 시스템 디바이스 함수 라이브러리의 함수(예를 들어, 각각 함수 1 및 함수 2)에 대응 함수(예를 들어, 대응 함수 1 및 대응 함수 2)를 저장하는 내장형 디바이스 함수 라이브러리(216)를 포함할 수 있다. 이러한 설계에 의해 예를 들어, 사용자(201)는 테스팅 층(204)과 연관된 내장형 처리 장치(208)에 의해 실행될 수 있는 테스트 로직을 시스템 층(202)에서 개발할 수 있다. 예를 들어, 내장형 디바이스 함수 라이브러리(216)의 함수의 적어도 일부가

시스템 디바이스 함수 라이브러리에서 각 상대 함수를 가지고 있기 때문에, 시스템 층(202)에서 개발된 테스트 로직은 내장형 디바이스 함수 라이브러리(216)의 소망의 함수를 적절하게 호출하는 함수 호출을 포함할 수 있다. 그래서, 적어도 부분적으로 시스템 처리 장치(206)에서 실행될 수 있고 필요할 때 테스팅 층(204)(또는 다른 층)의 함수를 적절하게 호출하는 테스트 로직이 생성될 수 있다. 예를 들어, 시스템 디바이스 함수 라이브러리(212)의 일부 함수는 이들의 내장형 디바이스 함수 라이브러리(216) 상대자에 랩퍼(wrapper)일 수 있다.

[0038] 시스템 층(202) 및 테스팅 층(204)에 유사한 함수를 제공함으로써 예를 들어, 테스트 레벨(204)에서 테스트 로직을 개발하는 것이 익숙하지 않은 개발자가 (예를 들어, 개발자가 보다 익숙할 수 있는 개발 환경에서) 시스템 층(202)에서 사용가능한 테스트 로직을 개발할 수 있다. 그러나, 일부 예에서, 일부 테스트 로직 또는 그 일부는 반응시간 또는 타이밍 가변성 제약과 같은 특정 테스트 필요사항의 성능 수요로 인해 테스팅 층(204)에서 개발되어야 한다.

[0039] 도 3은 시스템 층(302) 및 테스팅 층(304)을 포함하는 테스트 기기(300)를 도시하고 있다. 도 2에 관하여 상술된 테스트 기기(200)와 같이, 테스트 기기(300)는 실행을 위해 테스팅 층(304)에 전송될 수 있는 테스트 로직(예를 들어, 테스트 로직(316)을 사용자(301)가 생성할 수 있는 특징을 포함하고 있다. 예를 들어, 사용자(301)는 시스템 층(302)와 연관된 개발 환경(313)에서 테스트 로직(316)을 개발할 수 있다. 일부 예에서, 개발 환경(313)은 시스템 층(302)과 상이한 층 또는 머신에서 실행될 수 있다. 테스트 로직이 시스템 API(310)과 연관된 시스템 디바이스 함수 라이브러리(312)에 의해 제공된 함수(예를 들어, 함수 1 및 함수 2)를 포함할 수 있지만, 사용자(301)는 테스팅 층(304)과 연관된 내장형 처리 장치(308)에서 테스트 로직(316)의 일부 또는 모두가 실행되기를 원할 수 있다.

[0040] (예를 들어, 내장형 디바이스 함수 라이브러리(322)가 아닌) 시스템 함수 라이브러리(312)로부터의 함수를 사용하는 테스트 로직(316)의 실행을 돕기 위해, 시스템 층(302)은 테스트 로직(316)을 내장형 처리 장치(308)에 의해 실행될 수 있는 포맷으로 변경, 변환 또는 수정하기 위해 테스트 로직 변환 엔진(314)을 사용할 수 있다. 일부 예에서, 테스트 로직 변환 엔진(314)은 시스템 함수 라이브러리(312)의 함수와 내장형 디바이스 함수 라이브러리(322)의 대응 함수 사이의 연관성을 명시하는 하나 이상의 함수 맵핑(315)을 포함한다. 맵핑(315)을 사용하여, 테스트 로직 변환 엔진은 내장형 처리 장치(308)에서 실행되기에 적합한 변경된 테스트 로직(318)을 제공하도록 테스트 로직(316)을 수정할 수 있다. 일부 예에서, 테스트 로직 변환 엔진(314)은 시스템 함수 라이브러리(312)와 연관된 함수(예를 들어 함수 1)를 내장형 디바이스 함수 라이브러리(322)의 상대 함수(예를 들어, 대응 함수 1)로 대체하기 위해 맵핑(315)을 사용할 수 있다.

[0041] 테스트 로직 변환 엔진(314)이 변경된 테스트 로직(318)을 생성한 후에, 시스템 층(302)의 시스템 처리 장치(306)는 변경된 테스트 로직(318)을 실행하기 위해 내장형 처리 장치(308)에 제공할 수 있다. 내장형 처리 장치(308)가 변경된 테스트 로직(318)을 실행하기 전에, 테스팅 층(304)은 에러 체킹, 번역 또는 다른 사전 처리 동작(예를 들어, 테스팅 층(304)(구체적으로 내장형 디바이스 함수 라이브러리(322))이 변경된 테스트 로직(318)을 에러 체크할 수 있다)과 같은 하나 이상의 테스크를 실행할 수 있다. 일부 예에서, 이러한 체크의 일부는 이러한 테스트 로직이 실행할 때의 테스트 로직의 행동에 관한 것이고 이러한 체크의 일부는 변경된 테스트 로직(318) 이진수 자체에 관한 것이다. 이러한 체크는 예를 들어, 변경된 테스트 로직(318)이 로딩될 때, 실행될 수 있다. 또한, 일부 예에서, 내장형 처리 장치(308)에서 실행되기 전에 이러한 변경된 테스트 로직(318)을 수정하는 것이 바람직할 수 있다(예를 들어, 반응시간 또는 응답 시간 사양과 같은 테스트 성능 필요조건은 테스팅 층(304)에서 이러한 변경된 테스트 로직(318)에 추가 수정을 행할 것을 요구할 수 있다). 테스팅 층(304)에서의 변경된 테스트 로직(318)의 추가 수정을 돕기 위해, 테스팅 층(304)은 또한 시스템 층(302)로부터 전송된 변경된 테스트 로직(318)에 사용자(301)가 접근하고 변경할 수 있는 자체 개발 환경을 포함할 수 있다. 일부 예에서, 제1 계층은 제2 계층 보다는 제1 계층과 연관된 보다 큰 반응시간 및 타이밍 가변성을 가질 수 있다. 또한, 일부 경우에, 테스트 로직에 의한 액션은 대기 또는 가변적인 제1 계층의 타이밍보다 빠르게 종료하는 UUT 응답을 트리거할 수 있다. 이러한 경우에, 일부 SW 로직 최적화는 제1 계층에서 주목할만한 성능 향상을 얻지 못할 수도 있지만 제2 계층에서 극적인 향성을 얻을 수도 있다. 일부 예에서, 시스템 층(202)(또는 302)은 또한 시스템 API(210/310) 및 내장된 장치 API(214/314)중 하나 또는 모두와 상호작용하도록 사용될 수 있는 사용자 인터페이스(203)(도 2)를 제공할 수 있다. 예를 들어, 사용자 인터페이스는 양측 개발 환경(313, 324)으로의 공통 인터페이스로서 동작할 수 있어서 사용자(301)가 시스템 층(302) 또는 테스팅 층(304)에서 테스트 로직을 수정할 수 있다. 따라서, 상술된 기술에 의해 사용자(301)는 시스템 층(302)에서 (예를 들어, 사용자(301)에 익숙한 개발 환경에서) 테스트 로직을 조기 생성할 수 있고, 그다음, 이러한 테스트 로직

은 추가 수정 및/또는 실행을 위해 테스팅 층(304)에 전송되는 변경된 테스트 로직으로 변환될 수 있다. 또한 이러한 기술에 의해 사용자(301)는 보다 강력하고 익숙한 디버깅 툴을 사용하여 시스템 층(302)에서 조기 테스트 로직을 디버깅할 수 있다. 이러한 기술은 도 2에 관련하여 상술된 기술 대신에 또는 조합되어 사용될 수 있다. 예를 들어, 시스템 및 테스팅 층들의 함수 라이브러리에 대응 함수가 존재함으로써, 테스트 로직 변환 엔진(314)(및 그 연관된 맵핑(315))은 테스트 로직(316)에서 시스템 층 함수를 대신해야 하는 테스팅 층(304)에서의 적합한 함수를 식별할 수 있다.

[0042] 도 4는 제1 처리 시스템에서 테스트 프로그램을 생성하고 제2 처리 시스템에서 최종 실행을 위해 수정할 수 있는 프로세스(400)를 도시하고 있다. 이러한 프로세스(400)는 예를 들어, 테스트 기기에서 실행될 수 있다. 테스트 프로그램이 제1 처리 시스템(402)에서 수신된다. 예를 들어, 사용자는 시스템 층(302)과 연관된 개발 환경을 사용하여 테스트 프로그램을 생성할 수 있다. 사용자는 또한 네트워크 커넥션 또는 적합한 저장 장치(예를 들어, 유니버설 시리얼 버스(USB) 썸(thumb) 드라이브와 같은 휴대용 저장 장치)를 통해 제1 처리 시스템에 기존의 테스트 프로그램(302)을 업로드할 수 있다. 일부 예에서, 제1 처리 시스템은 테스트 기기로 인터페이스된 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실시하도록 프로그램가능하고, 테스트 기기의 동작을 제어하도록 프로그램된다.

[0043] 제2 처리 시스템(404)에서 실행가능한 변경된 테스트 프로그램이 생성된다. 일부 예에서, 제2 처리 시스템은 장치 테스트 전용이고 장치를 검사하기 위해 하나 이상의 테스트 프로그램을 실행하도록 프로그램가능하다. 제1 처리 시스템은 제1 애플리케이션 프로그래밍 인터페이스(API)를 가질 수 있고 제2 처리 시스템은 제2 API를 가질 수 있는데, 제1 API 및 제2 API는 상이한 API이고, 제1 API 및 제2 API는 적어도 일부 중복되는 함수를 갖고 있다. 테스트 프로그램이 제1 API의 하나 이상의 함수를 포함할 수 있지만, 변경된 테스트 프로그램을 생성하는 단계는 제1 API의 하나 이상의 함수에 상응하는 제1 API의 하나 이상의 함수를 식별하는 단계를 포함할 수 있다. 변경된 테스트 프로그램을 생성하는 단계는 또한 제1 API의 하나 이상의 함수를 제2 API의 하나 이상의 대응 함수로 대체하는 단계를 포함한다.

[0044] 변경된 테스트 프로그램은 제2 처리 시스템에 전송된다. 예를 들어, 변경된 테스트 프로그램이 제2 처리 시스템에 실행되기에 적합한 포맷으로 생성된 후에, 변경된 테스트 프로그램은 제2 처리 시스템에 제공될 수 있다. 제2 처리 시스템은 또한 제2 처리 시스템에서 이러한 변경된 테스트 프로그램을 수정할 수 있는 개발 환경을 제공할 수 있다.

[0045] 일부 상기 예가 변경된 테스트 로직을 테스팅 층의 하나의 내장형 처리 장치에 제공하는 단계를 기술하고 있지만, 시스템 층은 변경된 테스트 로직을 테스팅 층의 다수의 내장형 처리 장치에 제공할 수 있다. 일부 예에서, 하나 보다 많은 내장형 처리 장치가 하나 이상의 공유 API와 연관될 수 있다. 마찬가지로, 하나 이상의 시스템 처리 장치가 하나 이상의 공유 API와 연관될 수 있다.

[0046] 시스템 처리 장치, 내장형 처리 장치 또는 프로그램가능한 로직을 사용하여 테스트를 수행하는 방법이 상술되었다. 그러나, 테스트는 여기에 기술된 바와 같이 시스템 처리 장치, 내장형 처리 장치, 또는 프로그램가능한 로직의 조합을 사용하여 수행될 수 있다. 예를 들어, 이러한 상이한 요소들의 각각은 동일한 장치 또는 그 부분을 검사하기 위해 하나 이상의 테스트 프로그램에서 동시에 실행될 수 있다. 마찬가지로, 이러한 상이한 요소들은 예를 들어, 시스템 처리 장치(예를 들어, 도 1의 102)가 검사 시퀀스의 제1 부분을 수행하고, 내장형 처리 장치(예를 들어, 도 1의 106)가 동일한 검사 시퀀스의 제2 부분을 수행하고, 프로그램가능한 로직(예를 들어, 도 1의 FPGA(111))이 동일한 검사 시퀀스의 제3 부분을 수행하도록 검사를 조정할 수 있다. 임의의 적합한 조정이 여기에 기술된 테스트 기기의 상이한 프로그램 가능한 요소들 사이에서 이루어질 수 있다.

[0047] 또한, 일부 실시예에서, 하나의 계층의 처리를 피할 수 있다. 예를 들어, 검사는 내장형 처리 장치가 아닌 시스템 처리 장치(예를 들어, 102) 및 프로그램가능한 로직(예를 들어, FPGA(111))을 사용하여 수행될 수 있다. 이러한 실시예에서, 시스템 처리 장치와 프로그램가능한 로직 사이의 통신은 내장형 처리 장치를 통과할 수 있거나 내장형 처리 장치 계층을 완전히 우회할 수 있다.

[0048] 일부 실시예에서, 3개 보다 많은 계층의 처리 장치가 존재할 수 있다. 예를 들어, 2개 계층의 내장형 처리 장치가 존재할 수 있다(그 결과, 예를 들어, 총 4개의 계층이 된다). 예를 들어, 단일 내장형 처리 장치가 단일 장치의 검사를 조정하는데 사용될 수 있고, (이러한 단일 내장형 처리 장치에 의해 명령되는) 상이한 내장형 처리 장치가 이러한 단일 장치의 상이한 측면 또는 특징을 검사하는데 사용될 수 있다.

[0049] 일부 실시예에서, 하나 이상의 계층의 처리 장치가 도 1의 시스템으로부터 제거될 수 있다. 예를 들어, 일부

실시예는 내장형 처리 장치의 계층을 포함하지 않을 수 있다. 이러한 시스템 예에서, 오직 시스템 처리 장치(예를 들어, 도 1의 102) 및 프로그램가능한 로직(예를 들어, FPGA(111 내지 113))이 존재할 수 있다. 이와 관련하여, 임의의 적합한 조합의 계층이 여기에 기술된 테스트 기기에 채용될 수 있다.

[0050] 일부 실시예에서, 시스템 처리 장치(예를 들어, 도 1의 102)가 테스트 기기의 외부에 있을 수 있다. 예를 들어, 외부 컴퓨터는 테스트 기기의 동작을 제어하기 위해 채용될 수 있고, 여기에 기술된 방식으로 테스트 기기의 내장형 처리 장치 및 프로그램가능한 로직과 상호작용할 수 있다. 다른 실시예에서, 시스템 처리 장치는 테스트 기기의 일부일 수 있거나 테스트 기기로부터 떨어져 있다(예를 들어, 네트워크를 통해 테스트 기기에 접속되어 있다).

[0051] 일부 실시예에서, 프로그램가능한 로직은 논-프로그램가능한 로직으로 대체될 수 있다. 예를 들어, FPGA를 사용하기 보다는, 하나 이상의 주문형 집적 회로(ASIC)가 여기에 기술된 프로그램가능한 로직 대신에 또는 더하여 테스트 기기에 통합될 수 있다.

[0052] 여기에 기술된 기능, 또는 그 일부, 및 그 다양한 수정(이후로 "함수")은 여기에 기술된 하드웨어에 제한되지 않는다. 이러한 함수의 모두 또는 일부는 적어도 일부, 예를 들어, 프로그램가능한 프로세서, 컴퓨터, 멀티플 컴퓨터, 및/또는 프로그램가능한 로직 컴포넌트와 같은 하나 이상의 데이터 처리 장치에 실행되거나 이러한 데이터 처리 장치의 동작을 제어하기 위해, 하나 이상의 비일시적인 기계 판독가능 매체와 같은, 예를 들어, 정보 캐리어에 접촉식 구현된 컴퓨터 프로그램과 같은 컴퓨터 프로그램 제품을 통해 구현될 수 있다.

[0053] 컴퓨터 프로그램은 컴퓨터 번역되거나 해석된 언어를 포함하는 임의의 형태의 프로그래밍 언어로 기록될 수 있고, 독립형 프로그램, 또는 모듈, 컴포넌트, 서브루틴, 또는 컴퓨팅 환경에서 사용하기에 적합한 다른 유닛을 포함하는 임의의 형태로 전개될 수 있다. 컴퓨터 프로그램은 하나의 컴퓨터 또는 하나의 사이트 또는 다수의 사이트에 분포되고 네트워크에 의해 상호접속된 다수의 컴퓨터에서 실행되기 위해 전개될 수 있다.

[0054] 이러한 함수의 모두 또는 일부를 구현하는 것과 연관된 액션은 이러한 함수를 수행하기 위해 하나 이상의 컴퓨터 프로그램을 실행하는 하나 이상의 프로그램가능한 프로세서에 의해 수행될 수 있다. 이러한 함수의 모두 또는 일부는 예를 들어, FPGA 및/또는 ASIC(주문형 집적 회로)와 같은 전용 로직 회로로서 구현될 수 있다.

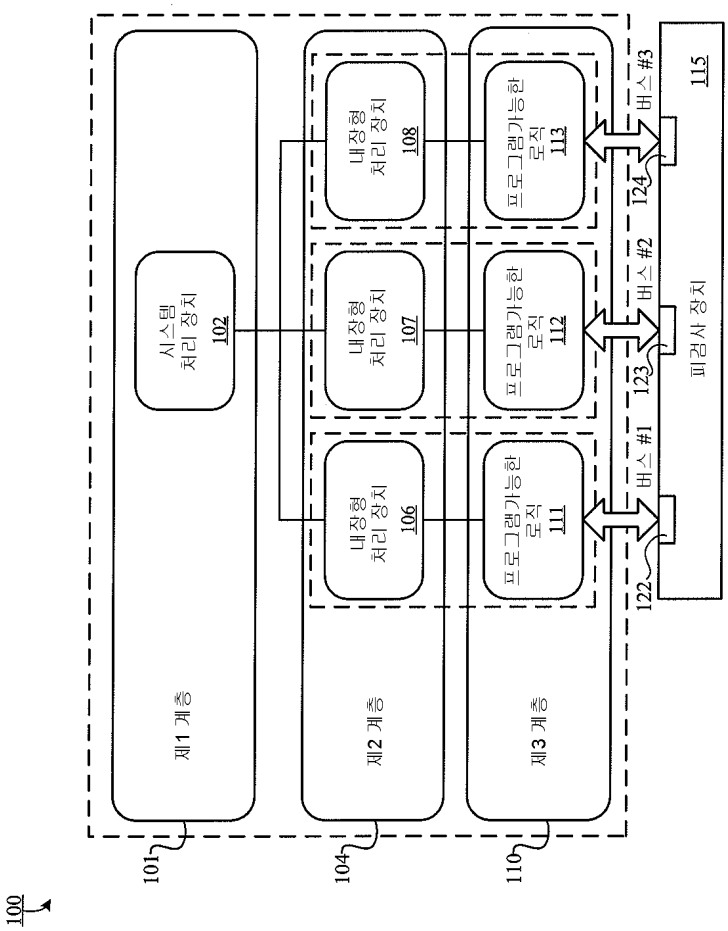
[0055] 컴퓨터 프로그램의 실행을 위해 적합한 프로세서는 예를 들어, 범용 및 전용 마이크로프로세서 모두 및, 임의의 종류의 디지털 컴퓨터의 어느 하나 이상의 프로세서를 포함한다. 일반적으로, 프로세서는 롬 또는 램 또는 모두로부터 명령어 및 데이터를 수신할 것이다. 컴퓨터의 컴포넌트는 명령어를 실행하기 위한 프로세서 및 명령어 및 데이터를 저장하기 위한 하나 이상의 메모리 장치를 포함한다.

[0056] 여기에 기술된 상이한 실시예의 컴포넌트는 위에서 구체적으로 제시되지 않은 다른 실시예를 형성하기 위해 조합될 수 있다. 컴포넌트는 도 1 내지 도 4에 도시된 회로에서 그 동작에 역효과를 주지 않고 배제될 수 있다. 또한, 다양한 별개의 컴포넌트가 여기에 기술된 기능을 수행하기 위해 하나 이상의 개별적인 컴포넌트로 조합될 수 있다.

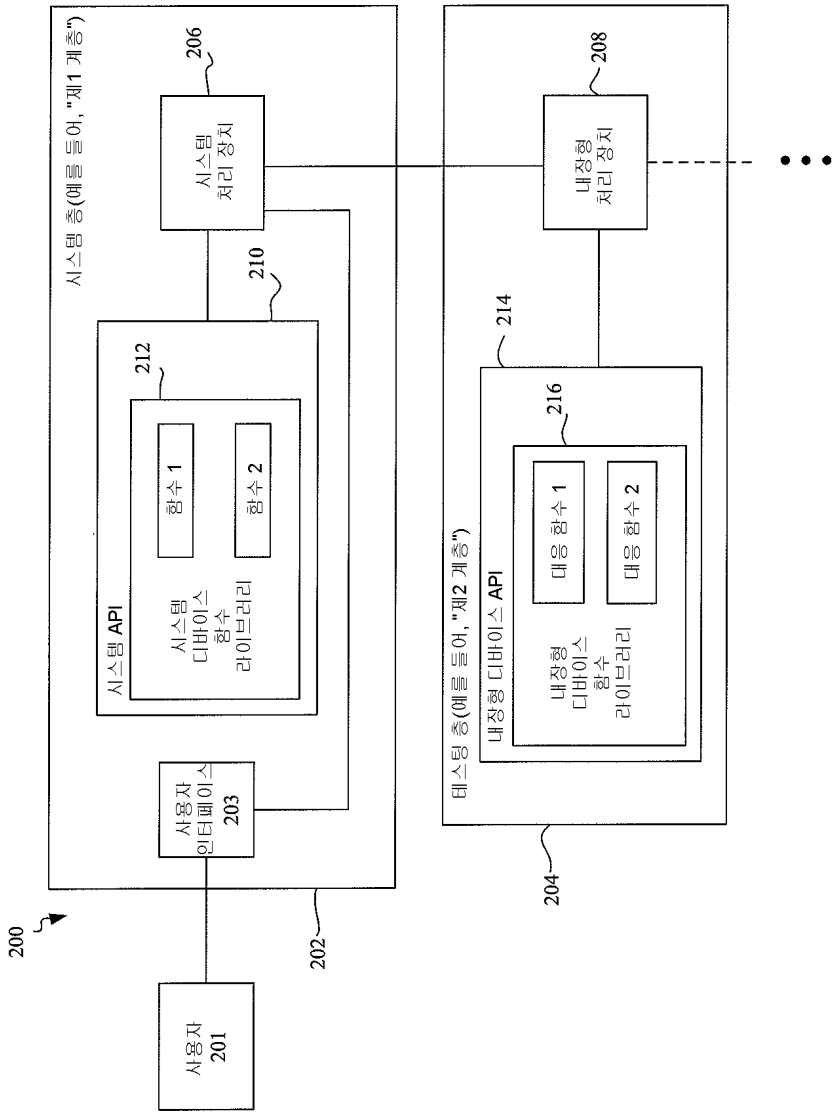
[0057] 여기에 구체적으로 기술되지 않은 다른 실시예 역시 다음의 청구범위에 포함되어 있다.

도면

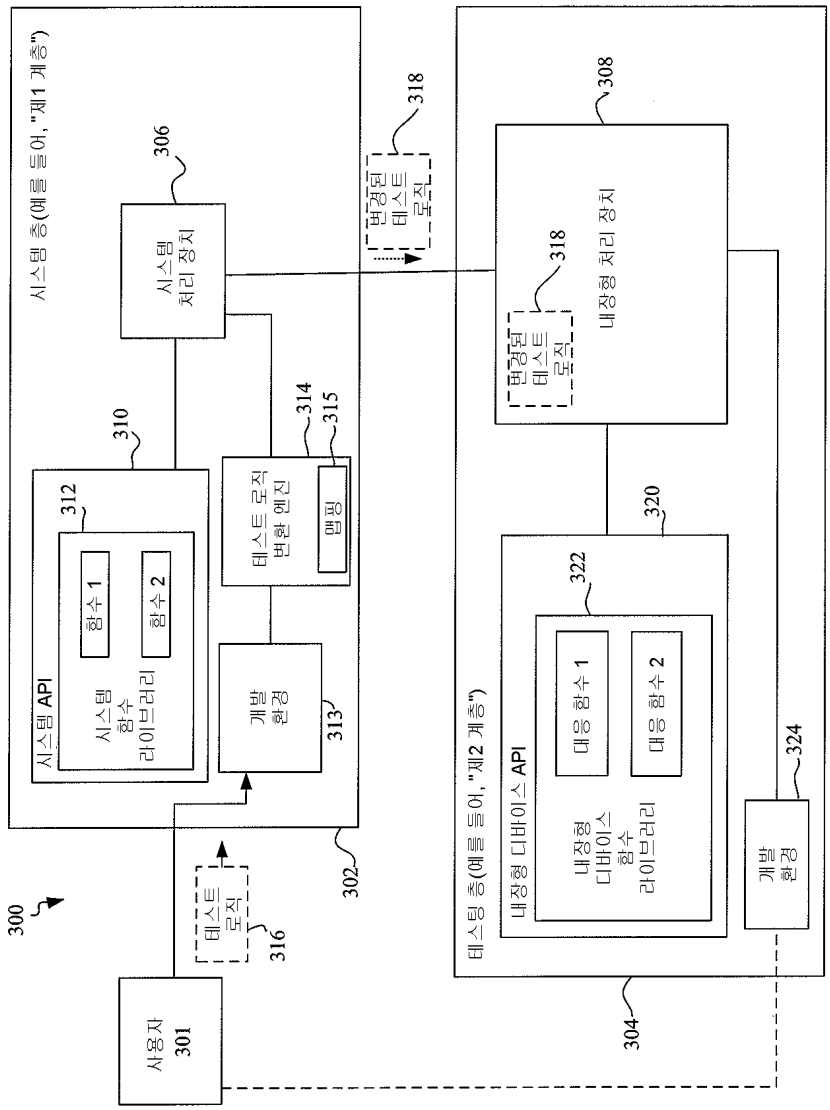
도면1


100
101
제1 계층
시스템 처리 장치 102
104
제2 계층
내장형 처리 장치 106
내장형 처리 장치 107
내장형 처리 장치 108
110
제3 계층
프로그램가능한 로직 111
프로그램가능한 로직 112
프로그램가능한 로직 113
버스 #1
버스 #2
버스 #3
122
123
124
115
피검사 장치

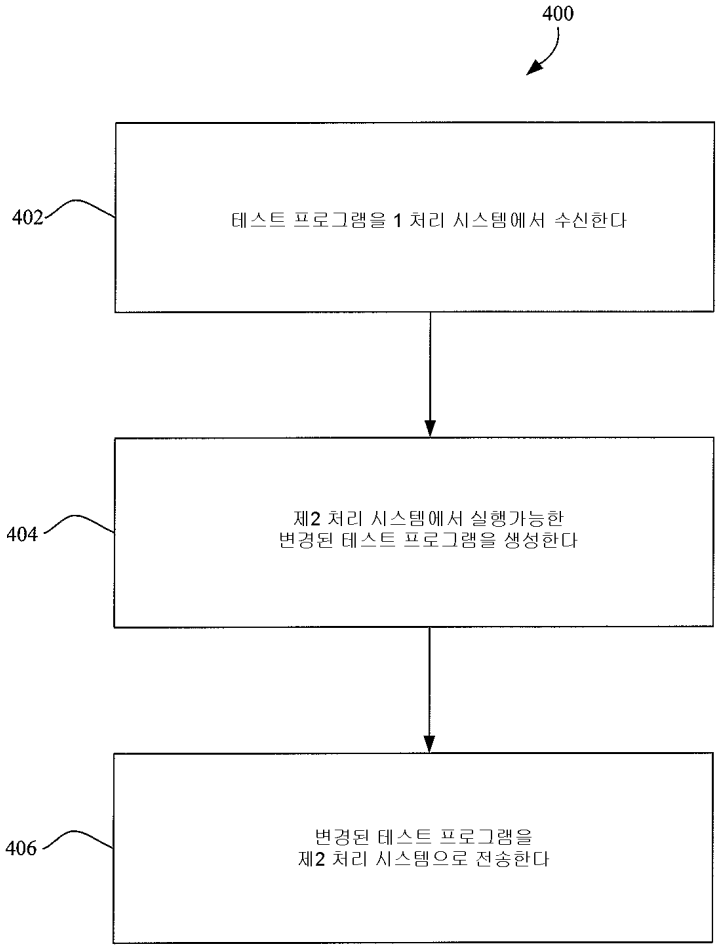
도면2


200
202
204
206
208
210
212
214
216
사용자
201
사용자
인터페이스
203
시스템 층(예를 들어, "제1 계층")
시스템
처리 장치
시스템 API
시스템
디바이스
함수
라이브러리
함수 1
함수 2
테스팅 층(예를 들어, "제2 계층")
내장형
처리 장치
내장형 디바이스 API
내장형
디바이스
함수
라이브러리
대응 함수 1
대응 함수 2

도면3


300
사용자
301
테스트
로직
316
302
시스템 층(예를 들어, "제1 계층")
시스템 API
시스템
함수
라이브러리
함수 1
함수 2
312
310
시스템
처리 장치
306
테스트 로직
변환 엔진
맵핑
314
315
개발
환경
313
변경된
테스트
로직
318
테스팅 층(예를 들어, "제2 계층")
304
내장형 디바이스 API
내장형
디바이스
함수
라이브러리
대응 함수 1
대응 함수 2
322
320
내장형 처리 장치
308
변경된
테스트
로직
318
개발
환경
324

도면4


400
402
테스트 프로그램을 1 처리 시스템에서 수신한다
404
제2 처리 시스템에서 실행가능한
변경된 테스트 프로그램을 생성한다
406
변경된 테스트 프로그램을
제2 처리 시스템으로 전송한다