(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2008-0098609
(43) 공개일자 2008년11월11일

(51) Int. Cl.
G01R 31/28 (2006.01)
(21) 출원번호 10-2008-7020067
(22) 출원일자 2008년08월14일
심사청구일자 없음
번역문제출일자 2008년08월14일
(86) 국제출원번호 PCT/US2007/060660
국제출원일자 2007년01월18일
(87) 국제공개번호 WO 2007/103591
국제공개일자 2007년09월13일
(30) 우선권주장
11/355,681 2006년02월16일 미국(US)

(71) 출원인
프리스케일 세미컨덕터, 인크.
미합중국 텍사스 (우편번호 78735) 오스틴 윌리암 캐논 드라이브 웨스트 6501
(72) 발명자
모리슨, 개리 알.
미국, 텍사스 78727-4513, 오스틴, 스테이튼 드리이브 13036
라이온, 조세 에이.
미국, 텍사스 78727, 오스틴, 안토이네트 플레이스 12209
(뒷면에 계속)
(74) 대리인
이범래

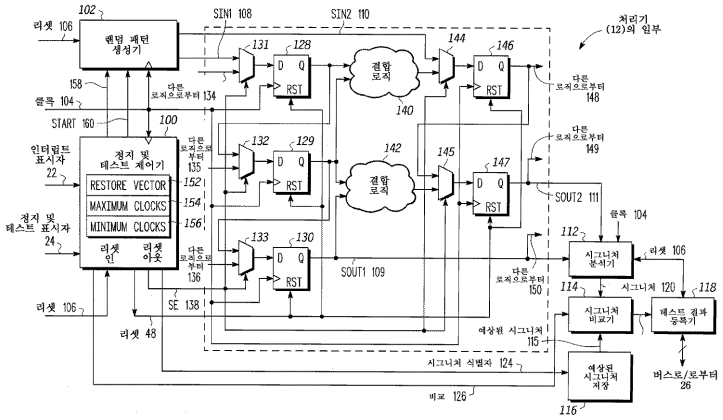
전체 청구항 수 : 총 25 항

(54) 데이터 처리 시스템을 테스트하기 위한 방법 및 장치

(57) 요 약

처리기(12)의 적어도 하나의 로직 블록을 테스트하는 방법은, 처리기에 의한 사용자 애플리케이션의 실행 동안, 처리기가 정지 및 테스트 표시자(24)를 생성하는 단계를 포함한다. 정지 및 테스트 표시자의 생성에 응답하여, 사용자 애플리케이션의 실행을 정지하고, 필요하다면 처리기의 적어도 하나의 로직 블록의 상태를 저장한다. 상기 방법은 처리기의 적어도 하나의 로직 블록을 테스트하기 위해 테스트 스티뮬러스를 적용하는 단계를 더 포함한다. 테스트 스티뮬러스는 사용자 애플리케이션의 실행 동안과 같이, 정상 동작 동안 처리기의 스캔 테스팅을 수행하도록 스캔 체인들에 이동될 수 있다.

대 표 도



(72) 발명자

모이어, 윌리엄 씨.

미국, 텍사스 78620, 드리핑 스프링스, 미도우 릿지 드라이브 1111

레이폴드, 안토니 엠.

미국, 텍사스 78749, 오스틴, 베일 밸리 드라이브 7700

특허청구의 범위

청구항 1

처리기의 적어도 하나의 로직 블록(logic block)을 테스트하는 방법에 있어서:

상기 처리기에 의한 사용자 애플리케이션의 실행 동안, 상기 처리기는 상기 정지 및 테스트 표시자를 발생시키고, 상기 정지 및 테스트 표시자의 발생에 응답하여 상기 사용자 애플리케이션의 실행을 정지시키고, 필요한 경우, 상기 처리기의 상기 적어도 하나의 로직 블록의 상태를 저장하는 단계; 및

상기 처리기의 상기 적어도 하나의 로직 블록을 테스트하는 테스트 스티뮬러스(test stimulus)를 적용하는 단계를 포함하는, 로직 블록 테스트 방법.

청구항 2

제 1 항에 있어서,

상기 정지 및 테스트 표시자를 발생시키기 위해 상기 처리기에 의해 정지 및 테스트 명령을 디코딩하는 단계를 더 포함하는, 로직 블록 테스트 방법.

청구항 3

제 2 항에 있어서,

상기 테스트 스티뮬러스를 적용하기 전에 스캔 체인 구성을 결정하기 위해 상기 정지 및 테스트 명령을 디코딩하는 단계를 더 포함하는, 로직 블록 테스트 방법.

청구항 4

제 1 항에 있어서,

상기 테스트 스티뮬러스를 적용하는 단계는 복수의 클록 사이클들에 대한 적어도 하나의 스캔 체인으로 상기 테스트 스티뮬러스를 시프트 인하는 단계를 더 포함하는, 로직 블록 테스트 방법.

청구항 5

제 4 항에 있어서,

상기 적어도 하나의 스캔 체인은 상기 처리기의 상기 적어도 하나의 로직 블록에 대응하는, 로직 블록 테스트 방법.

청구항 6

제 4 항에 있어서,

상기 테스트 스티뮬러스를 시프트 인(shift in)하기 전에 상기 처리기의 상기 적어도 하나의 로직 블록을 리셋하는 단계를 더 포함하는, 로직 블록 테스트 방법.

청구항 7

제 4 항에 있어서,

테스트 결과들의 제 1 세트에 대응하는 제 1 시그니처를 생성하기 위해, 상기 적어도 하나의 스캔 체인 밖으로 제 1 검사 지점에 대응하는 상기 테스트 결과들의 제 1 세트를 시프트 아웃(shift out)하고, 시그니처 분석기(signature analyzer)로 상기 테스트 결과들의 제 1 세트를 수신하는 단계로서, 상기 제 1 검사 지점은 제 1 복수의 클록 사이클들에 대응하는, 상기 시프트 아웃 및 수신 단계, 및 상기 제 1 시그니처를 제 1 예상된 시그니처와 비교하는 단계를 더 포함하는, 로직 블록 테스트 방법.

청구항 8

제 7 항에 있어서,

테스트 결과들의 제 2 세트에 대응하는 제 2 시그니처를 생성하기 위해, 상기 적어도 하나의 스캔 체인 밖으로 제 2 검사 지점에 대응하는 상기 테스트 결과들의 제 2 세트를 시프트 아웃하고 상기 시그니처 분석기로 상기 테스트 결과들의 제 2 세트를 수신하는 단계로서, 상기 제 2 검사 지점은 제 2 복수의 클록 사이클들에 대응하는, 상기 시프트 아웃 및 수신 단계, 및 상기 제 2 시그니처를 제 2 예상 시그니처와 비교하는 단계를 더 포함하는, 로직 블록 테스트 방법.

청구항 9

제 4 항에 있어서,

상기 적어도 하나의 스캔 체인은 적어도 제 1 메모리 소자, 메모리 소자들의 체인, 및 제 2 메모리 소자를 포함하고, 상기 제 1 메모리 소자의 출력은 상기 메모리 소자들의 체인의 입력 및 적어도 하나의 로직 게이트의 입력에 접속되고, 상기 적어도 하나의 로직 게이트의 출력은 상기 제 2 메모리 소자의 입력에 접속되는, 로직 블록 테스트 방법.

청구항 10

제 9 항에 있어서,

상기 적어도 하나의 로직 게이트는 배타적-OR 게이트 또는 배타적-NOR 게이트 중 적어도 하나인, 로직 블록 테스트 방법.

청구항 11

제 1 항에 있어서,

상기 테스트 스티뮬러스를 이용하여 상기 처리기의 상기 적어도 하나의 로직 블록을 작동하는 단계를 더 포함하는, 로직 블록 테스트 방법.

청구항 12

제 1 항에 있어서,

인터럽트의 수신에 응답하여, 상기 처리기에서 상기 사용자 애플리케이션을 실행하는 단계를 더 포함하는, 로직 블록 테스트 방법.

청구항 13

제 12 항에 있어서,

상기 사용자 애플리케이션을 실행하는 단계는 복원 벡터(restore vector)에서 시작하는 상기 사용자 애플리케이션을 실행하는 단계를 포함하는, 로직 블록 테스트 방법.

청구항 14

제 13 항에 있어서,

상기 사용자 애플리케이션 실행 단계는, 상기 처리기의 상기 적어도 하나의 로직 블록의 상태가 저장된 경우 상기 처리기의 상기 적어도 하나의 로직 블록의 상태를 복원하는 단계를 더 포함하는, 로직 블록 테스트 방법.

청구항 15

제 1 항에 있어서,

상기 복수의 클록 사이클들은 미리 결정된 최대값 및 미리 결정된 최소값 중 적어도 하나를 갖는, 로직 블록 테스트 방법.

청구항 16

제 1 항에 있어서,

테스트 결과들에 대응하는 적어도 하나의 시그니처를 생성하기 위해 상기 적어도 하나의 스캔 체인 밖으로 테스

트 결과들을 시프트 아웃하고 시그니처 분석기로 상기 테스트 결과들을 수신하는 단계를 더 포함하는, 로직 블록 테스트 방법.

청구항 17

제 16 항에 있어서,

상기 적어도 하나의 시그니처를 대응하는 예상된 시그니처와 비교하는 단계를 더 포함하는, 로직 블록 테스트 방법.

청구항 18

처리기의 적어도 하나의 로직 블록을 테스트하는 장치에 있어서:

상기 처리기에 의한 사용자 애플리케이션의 실행 동안, 정지 및 테스트 표시자를 발생시키고, 상기 정지 및 테스트 표시기의 발생에 응답하여, 상기 사용자 애플리케이션의 실행을 정지하고, 필요한 경우 상기 처리기의 상기 적어도 하나의 로직 블록의 상태를 저장하도록 구성된 테스트 제어기;

적어도 하나의 스캔 체인으로서, 상기 테스트 제어기는 또한 상기 처리기의 상기 적어도 하나의 로직 블록을 테스트하기 위해 복수의 클록 사이클들에 대한 상기 적어도 하나의 스캔 체인으로 테스트 스티뮬러스를 입력하고 상기 적어도 하나의 스캔 체인 밖으로 테스트 결과들을 출력하도록 구성되는, 상기 적어도 하나의 스캔 체인;

상기 테스트 스티뮬러스를 발생시키도록 구성되는 패턴 생성기; 및

상기 출력 결과들에 대응하는 적어도 하나의 시그니처를 발생시키도록 구성되고 상기 적어도 하나의 시그니처를 대응하는 예상된 시그니처와 비교하는 시그니처 분석기를 포함하는, 로직 블록 테스트 장치.

청구항 19

제 18 항에 있어서,

테스트의 성공 및 실패를 표시하는 적어도 하나의 값을 저장하도록 구성된 테스트 결과 레지스터를 더 포함하는, 로직 블록 테스트 장치.

청구항 20

제 19 항에 있어서,

상기 테스트 결과 레지스터는 또한 상기 복수의 클록 사이클들을 정량하는 값을 저장하도록 구성되는, 로직 블록 테스트 장치.

청구항 21

제 19 항에 있어서,

상기 테스트 결과 레지스터는 또한 마지막 완료된 검사 지점에 대한 다수의 클록 사이클들에 대응하는 값을 저장하도록 구성된, 로직 블록 테스트 장치.

청구항 22

제 18 항에 있어서,

테스트 성공 또는 실패 표시하는 복수의 값들을 저장하도록 구성되는 테스트 결과 레지스터를 더 포함하고, 상기 복수의 값들의 각각의 값은 검사 지점에 대응하는, 상기 로직 블록 테스트 장치.

청구항 23

제 18 항에 있어서,

상기 테스트 제어기는 또한 상기 적어도 하나의 시그니처가 상기 대응하는 예상된 시그니처에 매칭하지 않게 하는 강제된 에러 신호(forced error signal)를 발생시키도록 구성되는, 로직 블록 테스트 장치.

청구항 24

제 23 항에 있어서,

상기 강제된 에러 신호는 상기 패턴 생성기의 출력에 변동을 일으키는, 로직 블록 테스트 장치.

청구항 25

제 18 항에 있어서,

상기 정지 및 테스트 표시자의 발생 전에 상기 처리기의 상기 적어도 하나의 로직 블록의 레지스터들을 미리 결정된 값으로 설정하는 단계를 더 포함하는, 로직 블록 테스트 장치.

명 세 서

기 술 분 야

<1> 본 발명은 통상 데이터 처리 시스템들에 관한 것이고, 특히 데이터 처리 시스템을 테스팅하는 것에 관한 것이다.

배 경 기 술

<2> 데이터 처리 시스템들의 테스팅은 적절한 동작을 보장하기 위해 중요하다. 테스팅은 사용자 애플리케이션에서 데이터 처리 시스템을 이용하기 전에 제조 후에 공장에서 수행될 수 있다. 이는 최종 사용자가 적합하게 기능하는 제품을 받는 것을 보장한다. 그러나, 최종 사용자에 의해 데이터 처리 시스템의 동작 동안, 이는 또한 제품의 정상 동작 동안 일어날 수 있는 결함들이 검출될 수 있도록 데이터 처리 시스템을 계속적으로 테스트하는 것이 바람직할 것이다.

발명의 상세한 설명

<3> 데이터 처리 시스템이 제조되어 최종 사용자에게 전달된 후, 정상 동작 동안 테스팅은 데이터 처리 시스템이 여전히 적절하게 기능하는지를 보장하기에 바람직할 수 있다. 예를 들면, 데이터 처리 시스템은 차량 내에서 사용될 수 있고, 이러한 데이터 처리 시스템의 테스팅은 사용자가 차량의 구입한 후 바람직하다. 테스트 코드(test code)의 프래그먼트들(fragments)은 데이터 처리 시스템을 테스팅하기 위해 동작하는 동안 주기적으로 실행될 수 있다. 그러나, 이러한 테스트 코드는 일반적으로 비균일하고 불완전한 테스트 적용 범위(test coverage)를 제공한다. 이러한 테스트 코드는 또한 일반적으로 데이터 처리 시스템을 테스팅하기에는 느리고 비효율적이다. 또한, 이들 테스트 코드들은 시간이 걸리고 복잡한 결함을 가져야 한다. 스캔 테스팅(scan testing)은 다른 대안이다. 스캔 테스팅은 쉽게 더 높은 테스트 적용 범위를 제공하도록 이용될 수 있다. 그러나, 종래 스캔 테스팅은 동작시 수행되는 경우, 처리기의 현 상태가 스캔 테스팅이 수행되는 각 시간마다 소실된다. 여기에 기재되는 바와 같이, 본 발명의 실시예들은 소프트웨어에 대하여 정상 동작 동안 규제된 데이터 처리 시스템의 침입적 테스팅(intrusive testing)을 허용하고, 여기서 침입적 테스팅은 데이터 처리 시스템의 상태에 영향을 미치거나 그 상태를 잃게 하는 테스팅을 말할 수 있다. 예를 들면, 일 실시예는 정상 동작 동안 테스트하기 위해 로직 내장 자체 테스트(logic built in self test; LBIST)와 결합하여 스캔 테스트 방법을 이용한다. 일 실시예에서, 정지 및 테스트 명령은 테스팅을 시작하기 위해 이용된다.

<4> 여기에 이용된 바와 같이, 용어 "버스"는 하나 이상의 다양한 형태의 정보를 이동시키기 위해 이용될 수 있는 데이터, 어드레스들, 제어, 또는 상태와 같은 복수의 신호들 또는 도체들(conductors)을 지칭하는데 이용된다. 여기에 논의되는 도체들은 단일 도체, 복수의 도체들, 일방향 도체들, 또는 양방향 도체들에 관하여 예시되거나 기재될 수 있다. 그러나, 다른 실시예들은 도체들의 이행을 변경할 수 있다. 예를 들면, 개별적인 일방향 도체들은 양방향 도체들보다 덜 이용될 수 있고, 그 반대의 경우일 수도 있다. 또한, 복수의 도체들은 다수의 신호들을 연속적으로 이동시키거나 멀티플렉싱 방식으로 한번에 이동시키는 단일 도체로 대체될 수 있다. 유사하게는, 다수 신호들을 전달하는 단일 도체들은 이들 신호들의 서브세트들을 전달하는 다양한 다른 도체들로 분할될 수 있다. 그러므로, 많은 옵션들이 신호들을 이동시키기 위해 존재한다.

<5> 용어들 "확정하다(assert)" 또는 "설정하다(set)" 및 "무효화하다(negate)" 또는 "비확정하다(deassert)"는 신호, 상태 비트, 또는 유사한 장치를 논리적 참 또는 논리적 거짓 상태로 각각 렌더링하는 것을 지칭할 때 이용된다. 논리적 참 상태가 로직 레벨 1이면, 논리적 거짓 상태는 논리 레벨 0이다. 그리고, 논리적 참 상태가 로직 레벨 0이면, 논리적 거짓 상태는 1이다. 그러므로, 여기에 기재된 각각의 신호는 긍정 또는 부정 로직으

로 설계될 수 있고, 여기서 부정 로직은 신호 이름 위의 바 또는 명칭 뒤의 별표에 의해 표시된다. 부정 로직 신호의 경우, 신호는 액티브 로우(active low)이고, 여기서 논리 참 상태는 로직 레벨 0에 대응한다. 긍정 로직 신호의 경우, 신호는 액티브 하이(active high)이고, 논리 참 상태는 로직 레벨 1에 대응한다. 여기에 기재된 신호들 중 어떤 것도 부정이거나 긍정 로직 신호들 중 하나로 설계될 수 있다는 것을 주의하라. 그러므로, 긍정 로직 신호들로 기재된 이들 신호들은 부정 로직 신호들로서 실행될 수 있고, 부정 로직 신호들로서 기재된 이들 신호들은 긍정 로직 신호들로서 실행될 수 있다.

<6> 본 발명은 첨부하는 도면들에 의해 한정되는 것이 아닌 예시로서 설명되는 것이고, 유사한 참조 부호들은 유사한 요소들을 지시하는 것이다.

<7> 당업자는 도면들의 요소들이 간결함과 명확함을 위해 도시되고 필수적으로 비례하여 그려지지 않았음을 인식한다. 예를 들면, 도면들의 몇몇 요소들의 치수들은 본 발명의 실시예들의 이해를 향상시키기 위해 다른 요소들에 비해 과장될 수 있다.

실 시 예

<12> 도 1은 본 발명의 일 실시예에 따른 데이터 처리 시스템(10)을 블록도 형태로 도시한다. 데이터 처리 시스템(10)은 처리기(12), 정지 및 테스트 회로(stop and test circuitry; 14), 인터럽트 제어기(interrupt controller; 20), 다른 모듈들(16), 메모리(18), 및 버스(26)를 포함한다. 처리기(12), 인터럽트 제어기(20), 정지 및 테스트 회로(14), 다른 모듈들(16), 및 메모리(18)는 모두 버스(26)에 양방향으로 결합된다. 다른 모듈들(16)은 예를 들면, 다른 메모리들, 처리기들, 보조 처리기들, 입력/출력 장치들, 타이머들, 또는 어떤 다른 형태의 주변기기들과 같은 어떤 형태 및 수들의 모듈들을 포함할 수 있다. 대안으로, 데이터 처리 시스템(10)에 다른 모듈들이 존재하지 않을 수 있다. 메모리(18)는 예를 들면, 읽기용 메모리(ROM), 랜덤 액세스 메모리(RAM) 등과 같은 어떤 형태의 메모리일 수 있다. 일 실시예에서, 처리기(12)에 의해 실행되는 사용자 애플리케이션들이 메모리(18)에 저장된다. 또한, 예시된 실시예에서, 처리기(12)는 (이하에 또한 기재되는 바와 같이 선택적인) 다른 모듈들 모니터(other modules monitor; 50)를 포함한다.

<13> 정지 및 테스트 회로(14)는 인터럽트 제어기(20) 및 처리기(12)에 결합된다. 인터럽트 제어기(20)는 인터럽트 표시자(interrupt indicator; 22)를 정지 및 테스트 회로(14)에 제공한다. 정지 및 테스트 회로(14)는 처리기(12)에 리셋(reset; 48)을 제공하고, 제어기(12)로부터 정지 및 테스트 표시자(24)를 수신한다. 또한, 데이터 처리 시스템(10)은 멀티플렉서들(30, 32) 및 디멀티플렉서(demultiplexer; 28)를 포함하고, 이들 각각은 정지 및 테스트 인에이블(stop and test enable; 44)을 수신한다. 처리기(12)는 정지 및 테스트 인에이블(44)에 기초하여 정지 및 테스트 스캔 아웃(1-N)(34)으로서 정지 및 테스트 회로(14)에 제공되거나 외부 스캔 아웃(1-N)(37)로서 제공된 스캔 아웃(1-N)(40)을 디멀티플렉서(28)에 출력한다. 멀티플렉서(30)는 정지 및 테스트 회로(14)로부터 정지 및 테스트 스캔 인(1-N)(35) 및 외부 스캔 인(1-N)(38)을 수신하고 정지 및 테스트 인에이블(44)에 기초하여 이들 중 하나를 스캔 인(1-N)(41)으로서 처리기(12)에 제공한다. 멀티플렉서(32)는 정지 및 테스트 회로(14) 및 외부 스캔 인에이블(39)로부터 정지 및 테스트 스캔 인(1-N)(35)을 수신하고, 정지 및 테스트 인에이블(44)에 기초하여 이들 중 하나를 스캔 인에이블(42)로서 처리기(12)에 제공한다. 도 1의 N은 0보다 큰 어떤 정수를 나타내고, 처리기(12)에 존재하는 스캔 체인들의 수에 대응한다는 것을 주의하라. 그러므로, 멀티플렉서(30)는 N 멀티플렉서들을 포함할 수 있고, 디멀티플렉서(28)는 N 디멀티플렉서들을 포함할 수 있다는 것을 주의하라. 또한, 대안의 실시예에서, 디멀티플렉서(28)는 존재하지 않을 수 있다. 예를 들면, 스캔 입력들은 스캔 테스팅이 수행되지 않고 있을 때 간단히 무시될 수 있다.

<14> 외부 스캔 아웃(1-N)(37), 외부 스캔 인(1-N)(38), 및 외부 스캔 인에이블(39)은 데이터 처리 시스템(10)의 외부 패드들 또는 핀들에 각각 라우팅될 수 있다. 대안으로, 이들 외부 신호들의 어떤 것도 외부 패드들 또는 핀들의 수보다 작은 수로 멀티플렉싱될 수 있다. 예를 들면, 체인들의 모든 스캔은 N 패드들 또는 핀들을 통한 입력보다 동일한 패드 또는 핀을 통해 입력될 수 있다. 또한, 대안의 실시예에서, 다른 회로는 멀티플렉서들(30, 32) 및 디멀티플렉서(28)의 기능성을 구현하기 위해 제공될 수 있다. 또 다른 실시예에서, (외부 스캔 아웃(37) 및 외부 스캔 인(38)과 같은) 외부 스캔 입력들 및 출력들이 모두에 제공되지 않을 수 있고, 모든 스캔 입력들 및 출력들은 정지 및 테스트 회로(14)로부터 및 그에 제공된다.

<15> 동작시, 정지 및 테스트 인에이블(44)이 확인되지 않을 때, 멀티플렉서들(30, 32)은 외부 스캔 인(1-N)(38) 및 외부 스캔 인에이블(39)을 각각 선택하고, 외부 스캔 인(1-N)(38)은 스캔 인(1-N)(41)으로서 처리기(12)에 제공되고 외부 스캔 인에이블(39)은 스캔 인에이블(42)로서 제공된다. 유사하게는, 정지 및 테스트 인에이블(44)이 확정되지 않을 경우, 디멀티플렉서(38)는 처리기(12)로부터 외부 스캔 아웃(1-N)(37)으로 스캔 아웃(1-N)(40)을

제공한다. 이러한 방식으로, 외부 테스터는 테스트 처리기(12)를 스캔하기 위해 처리기(12)의 스캔 입력 및 출력들에 결합될 수 있다. 예를 들면, 처리기(12)의 제조 또는 데이터 처리 시스템(10)의 제조 후, 처리기(12)는 외부 스캔 아웃(1-N)(37), 외부 스캔 인(1-N)(38), 및 외부 스캔 인에이블(39)에 결합된 종래의 테스터를 이용하여 공장 테스트될 수 있다. 그러므로, 종래 스캔 테스트 방법은 이러한 공장 테스팅을 수행하기 위해 이용될 수 있다. 그러나, 정지 및 테스트 회로(14)는, 최종 사용자에 의해 이용되는 동안과 같이, 실제 동작 동안 개선된 처리기(12)의 테스팅을 제공하기 위해 이용될 수 있다. 정지 및 테스트 회로(14)의 동작은 도 2 및 도 3을 참조하여 이하에 더 설명될 것이다. (데이터 처리 시스템(10)의 다른 부분들의 동작은 본 기술에 알려진 바와 같이 동작하고, 그러므로 여기에 기재된 실시예들의 이해에 관련된 데이터 처리 시스템(10) 및 처리기(12)의 그들 일부분들은 더 자세하게 설명될 것임을 주의하라.)

<16> 도 2는 본 발명의 일 실시예에 따른 처리기(12)의 일부(점선 내부) 및 정지 및 테스트 회로(14)의 일부(점선 외부)를 도시한다. 일반적으로, 처리기(12)의 회로는 결합 로직 또는 회로로부터 입력들을 수신하는 복수의 플립 플롭들 또는 래치들(latches)로서 도시될 수 있고 결합 로직 또는 회로에 출력들을 제공할 수 있다. 예를 들면, 도 2는 D 플립 플롭들(128-130, 146-147) 및 결합 로직(140, 142)을 포함하는 처리기(12)의 회로의 작은 부분을 도시한다. 결합 로직(140, 142)의 각각은 다양한 다른 기능들을 수행하기 위해 어떤 형식의 결합 로직도 포함할 수 있다. 예를 들면, 140, 142 각각은 (예를 들면, AND 게이트들, OR 게이트들, NOR 게이트들, XOR 게이트들 등과 같은) 어떤 수의 로직 게이트들도 포함할 수 있다.

<17> 플립 플롭(128) 및 플립 플롭(129)은 결합 로직(140)에 그들의 출력을 제공하고 결합 로직(140)은 멀티플렉서(144)에 출력을 제공하고, 멀티플렉서(144)는 플립 플롭(146)에 그의 출력을 제공한다. 유사하게는, 플립 플롭(129) 및 플립 플롭(130)은 그들의 출력들을 결합 로직(142)에 제공하고 결합 로직(142)은 멀티플렉서(145)에 출력을 제공하고, 멀티플렉서(145)는 그것의 출력을 플립 플롭(147)에 제공한다. 플립 플롭(130)은 또한 그것의 출력 Sout1(109)를 시그니처 분석기(signature analyzer; 112) 및 다른 로직(150)(예를 들면, 도 2에 도시되지 않은 처리기(12)내 다른 결합 로직)에 제공한다. 멀티플렉서(144)는 또한 Sin2(110)을 입력으로 수신한다. 플립 플롭들(128-130)의 각각은 멀티플렉서들(131-133) 각각으로부터 데이터 입력을 수신한다. 멀티플렉서(131)는 랜덤 패턴 생성기(102)로부터 입력 Sin1(108)을 수신하고, 다른 로직(134)(예를 들면, 도 2에 도시되지 않은 처리기(12)내 다른 결합 로직)으로부터 다른 입력을 수신한다. 멀티플렉서(132)는 입력으로서 플립 플롭(128)의 출력을 수신하고 다른 로직(135)(예를 들면, 도 2에 도시되지 않은 처리기(12)내 다른 결합 로직)으로부터 다른 입력을 수신한다. 멀티플렉서(133)는 입력으로서 플립 플롭(129)의 출력을 수신하고 다른 로직(136)(예를 들면, 도 2에 도시되지 않은 처리기(12)내 다른 결합 로직)으로부터 다른 입력을 수신한다. 또한, 플립 플롭(146)은 그것의 출력을 멀티플렉서(145)의 입력 및 다른 로직(148)(도 2에 도시되지 않은 처리기(12)내 다른 결합 로직)에 제공한다. 플립 플롭(147)은 그것의 출력 SOUT2(111)을 시그니처 분석기(112) 및 다른 로직(149)(도 2에 도시되지 않은 처리기(12)내 다른 결합 로직)에 제공한다.

<18> 처리기(12)는 어떤 형식의 구성을 포함할 수 있다는 것을 주의하라. 예를 들면, 어떤 플립 플롭들의 출력들은 어떤 수의 결합 로직 부분들에 제공될 수 있고, 입력을 동일한 플립 플롭에 제공하는 결합 로직 부분에 피드백될 수도 있다. 유사하게는, 각각의 플립 플롭은 어떤 결합 로직 부분으로부터 그것의 입력을 수신할 수 있다. 처리기(12)는 처리기(12)의 기능을 수행하기 위해 필요한 만큼 어떤 수의 플립 플롭들 및 어떤 양의 결합 로직도 포함할 수 있다. 또한, 처리기(12)는 함께 결합된 어떤 수의 로직 블록들을 포함하는 것으로 나타낼 수 있고, 각각의 로직 블록은 적어도 하나의 스캔 체인 또는 스캔 체인의 일부분을 포함할 수 있다. 그러므로, 모든 처리기(12) 또는 처리기(12)의 하나 이상의 로직 블록들이 스캔 테스팅될 수 있다.

<19> 도 2의 점선의 외부에 정지 및 테스트 회로(14)의 일부가 도시되었고, 이는 랜덤 패턴 생성기(102), 정지 및 테스트 제어기(100), 시그니처 분석기(112), 시그니처 비교기(114), 예상된 시그니처 저장소(116), 및 테스트 결과 레지스터들(118)을 포함한다. 랜덤 패턴 생성기(102)는 Sin1(108) 및 Sin2(110)을 처리기(12)에 제공한다. Sin1(108) 및 Sin2(110)는 정지 및 테스트 인에이블(44)이 확정될 경우, 멀티플렉서들(30)을 통해 정지 및 테스트 회로(14)로부터 수신된 도 1에 도시된 처리기(12)로의 두 개의 스캔 인(1-N)(41) 입력들일 수 있다는 것을 주의하라. Sout1(109) 및 Sout2(111)는 시그니처 분석기(112)에 제공된다. Sout1(109) 및 Sout2(111)는 정지 및 테스트 인에이블(44)이 확정될 때, 디멀티플렉서(28)를 통해 처리기(12)로부터 정지 및 테스트 회로(14)로 제공되는 도 1에 도시된 처리기(12)로부터의 두 개의 스캔 아웃(1-N)(40) 출력들일 수 있다는 것을 주의하라. 또한, 정지 및 테스트 제어기(100)는 모든 멀티플렉서들(131-133, 144, 145)에 스캔 인에이블(SE)(138)을 제공한다. SE(138)는 정지 및 테스트 인에이블(44)이 확정될 때 멀티플렉서(32)를 통해 정지 및 테스트 회로(14)로부터 수신된 도 1에 도시된 처리기(12)에 대한 스캔 인에이블(42)일 수 있다는 것을 주의하라.

<20> 도 2에서, 정지 및 테스트 회로(14) 및 처리기(12)의 사이의 출력들 및 입력들은 도 2를 너무 복잡하게 하지 않도록 멀티플렉서들(28, 30, 32) 없이 직접 접속들로서 도시된다. 또한, 외부 스캔 접속들이 처리기(12)를 요구하지 않는 경우, 데이터 처리 시스템(10)은 또한 멀티플렉서들(28, 30, 32)을 포함하지 않을 수 있고, 정지 및 테스트 회로(14) 및 처리기(12) 간의 입력들 및 출력들은 도 2에 도시된 바와 같이 직접 접속될 것이다.

<21> 또한 도 2를 참조하면, 정지 및 테스트 제어기(100)는 리셋(106)을 수신하고 모든 플립 플롭들(128-130, 146, 147)에 리셋(48)을 제공한다. 일 실시예에서, 리셋(106)은 버스(26)를 통해 수신된 신호이다. 리셋(106)은 또한 시그니처 분석기(112), 랜덤 패턴 생성기(102), 및 테스트 결과 레지스터(118)에 제공된다. 정지 및 테스트 제어기(100)는 또한 복원 벡터(restore vector; 152), 최대 클록들(154), 및 최소 클록들(156)을 포함한다. 이들은 정지 및 테스트 제어기(100)에 위치한 레지스터들 또는 다른 저장 회로일 수 있다(또는 정지 및 테스트 회로(14) 또는 데이터 처리 시스템(10)내의 어느 곳에도 위치될 수 있다). 정지 및 테스트 제어기(100)는 인터럽트 제어기(20)로부터 인터럽트 표시자(22)를 수신하고, 강제된 에러 신호(forced error signal; 158) 및 시작(160)을 랜덤 패턴 생성기(102)에 제공하고, 시그니처 식별자(signature identifier; 124)를 예상된 시그니처 저장소(116)에 제공하고, 비교(126)를 시그니처 비교기(114)에 제공한다. 시그니처 비교기(114)는 시그니처 분석기(112)로부터 시그니처(120)를 수신하고 예상된 시그니처 저장소(116)로부터 예상된 시그니처(115)를 수신하고, 두 개를 비교하고 비교 결과들(122)을 테스트 결과 레지스터(118)에 제공한다. 테스트 결과 레지스터(118)는 또한 버스(26)에 양방향으로 결합될 수 있다. 테스트 결과 레지스터(118) 및 예상된 시그니처 저장소(116)는 레지스터들로서 각각 구현될 수 있거나 어떤 다른 저장 회로로서 구현될 수 있다는 것을 주의하라. 예를 들면, 그들은 데이터 처리 시스템(10)내 메모리에 위치될 수 있다. 또한, 테스트 결과 레지스터(118) 및 예상된 시그니처 저장소(116)는 데이터 처리 시스템(10)내 어느 곳에도 위치될 수 있다.

<22> 본 기술에서 공지된 바와 같이 생성될 수 있는 클록(104)은 랜덤 패턴 생성기(102), 정지 및 테스트 제어기(100), 시그니처 분석기(112), 및 처리기(12)(예를 들면, 플립 플롭들(128-130, 146, 147)의 각각에)에 제공된다. 클록(104)은 데이터 처리 시스템(10)내의 유일한 클록을 나타낼 수 있거나 데이터 처리 시스템(10) 및 처리기(12)의 하나의 클록 도메인을 나타낼 수 있다. 이러한 경우에는, 처리기(12)의 다른 부분들은 예를 들면 다른 클록 도메인 내에 위치될 수 있고, 그러므로 (클록(104)과는 다른) 다른 클록을 수신한다. 클록 도메인 클록들 각각은 동일한 시스템 클록으로부터 생성될 수 있거나 개별적으로 생성될 수 있다. 구성과 관계없이 이들 클록들은 본 기술에 공지된 바와 같이 생성되고 제공될 수 있다.

<23> 동작시, 멀티플렉서들(131-133, 144, 145)은 처리기(12)의 스캔 테스팅을 허용한다는 것을 주의하라. 이들 멀티플렉서들은 다양한 스캔 체인들을 생성하기 위해 이용된다(도 2에 도시된 두 개의 체인들, 하나는 입력 Sin1(108) 및 출력 Sout1(109)를 갖고, 다른 것은 입력 Sin2(110) 및 Sout2(111)을 갖는다.). 예를 들면, 하나의 스캔 체인은 플립 플롭들(128-130)을 포함하고, SE(138)이 확정될 경우, 스캔 체인 입력 Sin1(108)은 입력으로서 플립 플롭(128)에 제공되고, 플립 플롭(128)의 출력은 입력으로서 플립 플롭(129)에 제공되고, 플립 플롭(129)의 출력은 입력으로서 플립 플롭(130)에 제공되고, 플립 플롭(130)의 출력은 스캔 체인 출력 Sout1(109)를 제공한다. 유사하게는, 처리기(12)의 다른 스캔 체인은 플립 플롭들(146, 147)을 포함하고, 여기서 SE(138)이 확정될 경우, 스캔 체인 입력 Sin2(110)은 입력으로서 플립 플롭(146)에 제공되고, 플립 플롭(146)의 출력은 입력으로서 플립 플롭(147)에 제공되고, 플립 플롭(147)의 출력은 스캔 체인 출력 Sout2(111)을 제공한다. 그러므로, SE(138)이 확정될 경우, 스캔 입력들 및 출력들은 처리기(12)의 모든 부분 또는 일부분들을 테스트하기 위해 각각 시프트 인 및 아웃될 수 있다.

<24> SE(138)가 확정된 경우, 처리기(12)는 정상으로 동작하고, 여기서 입력들 플립 플롭들(128-130, 146, 147)은 처리기(12)의 결합 로직으로 제공되고, 출력은 처리기(12)의 결합 로직에 제공된다는 것을 주의하라. 즉, SE가 확정되지 않은 정상 동작 동안, 플립 플롭들의 입력들 및 출력들은 체인화되지 않고, 그래서 상기에 설명된 스캔 체인들을 형성하지 않는다.

<25> 예시된 실시예에서, mux-D 스캔 체인 구성이 예시되었고, 여기서 멀티플렉서들은 D 플립 플롭들이 정상 동작을 위해 구성될 수 있고 테스팅 동안 스캔 체인들로서 구성될 수 있도록 D 플립 플롭들 각각의 입력들에 이용된다는 것을 주의하라. 그러나, 다른 스캔 체인 구성들 및 다른 메모리 소자들이 이용될 수 있다는 것을 주의하라. 예를 들면, 본 기술에서 공지된 레벨-센서티브 스캔 설계(Level-Sensitive Scan Design; LSSD)가 이용될 수 있다. 처리기(12)의 스캔 체인 구성은 변경될 수 있다는 것을 또한 주의하라. 예를 들면, 처리기(12)는 단일 스캔 체인이거나 어떤 수의 스캔 체인들일 수 있다. 또한, 단일 또는 어떤 수의 스캔 체인들은 상기에 기재된 바와 같이, 각각의 클록 도메인에 대해 존재할 수 있다. 또한, 각각의 스캔 체인은 어떤 임의의 길이일 수 있다(예를 들면, 어떤 임의의 수의 플립 플롭들 또는 다른 메모리 소자들을 포함함). 또한, 몇몇 실시예들에서, 스

캔 체인 구조는 설정가능할 수 있고, 스캔 체인 접속들은 처리기(12)의 일부분 내의 스캔 체인들의 수 및/또는 길이를 변경하는 것을 허용하도록 변경될 수 있다. 이러한 구성은 처리기(12)의 플립 플롭들의 출력의 교호하는 상호 접속을 수행하기 위해 멀티플렉서들(131, 132, 133, 144, 145)로의 입력들 및 제어들을 변경함으로써 달성될 수 있다. 이러한 제어는 정지 및 테스트 제어기(100)에 의해 제공될 수 있거나, 데이터 처리 시스템(10) 내의 어느 부분으로부터 제공될 수 있다.

<26> 정상 동작 동안 SE(138)가 무효화될 때, 처리기(12)의 플립 플롭들은 처리기의 현재 상태를 저장한다는 것을 주의하라. 각각의 클록 펄스에 대하여, 하나의 플립 플롭으로부터 다른 것으로 결합 로직을 통해 전달되는 값들에 기초하여, 플립 플롭들의 값들이 그에 따라 업데이트된다. 예를 들면, 정상 동작 동안 SE(138)가 무효화될 때, 플립 플롭들(128, 129)의 출력들은 입력을 생성하기 위해 결합 로직(140)을 통해 멀티플렉서(144)로 전달한다. 클록(104)이 플립 플롭(146)을 클록킹할 때, 이러한 결합 로직(140)의 출력은 (멀티플렉서(144)에 대한 선택 신호, SE(138)가 무효화되기 때문에) 멀티플렉서(144)를 통해 플립 플롭(146)에 저장된다. 스캔 테스팅이 수행되는 동안 시프팅할 때(SE(138)가 확정된 경우), 플립 플롭들은 (상기 기재된 바와 같이) 스캔 체인들을 형성하고, 다수의 스캔 체인 입력들(즉, 테스트 스티뮬러스(test stimulus) 또는 테스트 입력들)이 시프트 인된다. 다수의 입력들이 시프트 인되었을 때, SE(138)가 무효화되고, 다음 클록은 테스팅 동안 함수 사이클을 생성하고, 결합 로직을 통해 전달하는 시프트 인된 테스트 입력들로부터 기인된 테스트 결과들은 스캔 체인들에 캡처링된다. 이후, SE(138)는 다시 확정되고 캡처링된 출력이 시프트 아웃된다. 이러한 방식으로, 입력들(테스트 스티뮬러스)은 SE(138)가 확정될 때 처리기(12)에 제공되고, 클록은 함수 사이클을 제공하도록 SE(138)가 무효화될 때 제공되고, 출력들은 SE(138)가 다시 확정될 때 처리기(12)에서 판독된다.

<27> 일 실시예에서, 입력들의 수는 SE(138)가 확정될 때, 시프트 인되고, 동일한 수의 출력들이 시프트 아웃된다. 즉, 각각의 시간에 테스트 입력은 시프트 인되고, 테스트 출력은 동시에 시프트 아웃된다. 또한, 어떤 수의 테스트 입력들 및 출력들도 함수 사이클에 대해 허용되도록 SE(138)가 무효화되기 전에 각각 시프트 인되고 시프트 아웃될 수 있다. 이러한 수는 1 이상의 어떤 정수일 수 있다. 예를 들면, 일 실시예에서, (하나의 출력이 시프트 아웃되는 동안) 하나의 입력이 시프트 인된 후, SE(138)이 무효화되고, 함수 주기는 다음 클록에 생긴다. 대안의 실시예에서, 다섯 개의 입력들은 함수 사이클에 대해 허용되도록 SE(138)가 무효화되기 전에 (다섯 개 출력들이 시프트 아웃되는 동안) 시프트 인될 수 있다. 그러므로, 스캔 테스팅 동안, 클록(104)은 테스트 입력/출력 시프팅 사이클들에 대해 허용하는 바와 같이, 입력들 및 출력들은 SE(138)이 확정될 때 스캔 체인들로 시프트되거나 그의 외부로 시프트되고, 함수 사이클들에 대해 허용하는 바와 같이, SE(138)가 무효화되고, 결합 로직을 통해 전달하는 테스트 인된 테스트 스티뮬러스에 기인하는 테스트 결과들이 캡처링된다.

<28> 예를 들어, 도 2를 참조하면, 스캔 테스팅 동안, SE(138)이 확정되고 특정 수의 테스트 입력들은 스캔 체인들로 스캐닝된다. 즉, 클록(104)의 특정 수의 클록 사이클들은 입력 테스트 입력들을 캐시 체인(cache chain)으로 입력하기 위해 이용된다. 예를 들면, 클록(104)의 제 1 테스트 입력/출력 시프트 사이클에서, 테스트 스티뮬러스는 Sin1(108) 및 Sin2(110) 각각을 통해 플립 플롭들(128, 146)의 각각으로 시프트된다. 이러한 클록(104)의 제 1 테스트 입력/출력 시프트 사이클동안, 값들은 또한 Sout1(109) 및 Sout2(111) 각각을 통해 플립 플롭들(130, 147)로부터 시프트 아웃된다. 클록(104)의 다음 테스트 입력/출력 시프트 사이클 동안, 플립 플롭들(128, 146) 내의 값들은 플립 플롭들(129, 147) 각각으로 시프트되고, 플립 플롭(129)내의 값은 플립 플롭(130)으로 시프트된다. 플립 플롭들(130, 147)의 값들은 Sout1(109) 및 Sout2(111) 각각과 같은 출력이다. 어떠한 수의 이들 테스트 입력/출력 시프트 사이클들도 바람직하게 생길 수 있다. 바람직한 수의 테스트 입력/출력 사이클들 후에, 함수 사이클은 SE(138)을 무효화함으로써 수행된다. 이러한 함수 사이클에서, 다른 로직(134), 다른 로직(135), 다른 로직(136), 결합 로직(140), 및 결합 로직(142)으로부터 전달하는 값들은 클록(104)을 갖는 플립 플롭들(128-130, 146, 147) 각각에 캡처링된다. 다음 테스트 입력/출력 시프트 사이클들 동안, 이들 캡처링된 값들은 (새로운 입력들이 시프트 인될 때) 시프트 아웃된다.

<29> 일반적으로 스캔 테스팅이 시작되기 전에, 플립 플롭들은 예를 들면 논리 레벨 0과 같이 공지된 값으로 리셋된다는 것을 주의하라. 또한, 스캔 테스팅이 시작하기 전에, 플립 플롭들은 스캔 체인들의 수 및 스캔 체인 길이들을 규정하기 위해 수개의 대안들 중 하나로 구성될 수 있다. 구성 제어는 도 2의 정지 및 테스트 제어기(100)로부터 또는 데이터 처리 시스템(10) 내 어느 곳으로부터 제공될 수 있다.

<30> 시프트 인된 테스트 입력들에 기초하여, 특정 결과들이 예상된다. 그러므로, 스캔 체인들의 시프트 아웃된 출력값들은 이후 시프트 인된 테스트 입력들에 대응하는 예상된 시그니처와 비교되는 시그니처를 생성하기 위해 시그니처 분석기에 제공될 수 있다. 그들이 매칭하지 않을 경우, 불합격이 표시된다. 일 실시예에서, 입력들의 미리 결정된 벡터들은 이들 미리 결정된 벡터들 각각이 대응하는 예상된 시그니처를 갖는 경우 시프트 인될

수 있다는 것을 주의하라. 대안으로, LBIST는 의사 랜덤 번호 생성기(pseudo random number generator)가 의사 랜덤 벡터들(pseudo random vectors)(테스트 입력들), 즉 미리 결정되는 벡터들이 아닌 벡터들을 생성하기 위해 이용될 수 있는 경우, 이용될 수 있다. 이들 벡터들은 또한 테스트의 합격 또는 불합격을 결정하기 위해 이용될 수 있는 연관된 예상된 시그니처들을 갖는다.

<31> 스캔 테스팅은 시스템의 전체 안정성을 제공하는 좋은 방법임이 주의하라. 즉, 모두, 대부분을 쉽게 테스트하거나, 장치의 큰 부분이 간단히 테스트되게 하기 위해 이용될 수 있다(즉, 테스트중인 장치, 또는 도 2의 예의 처리기(12)인 DUT). 그러나, 스캔 테스팅은 DUT의 상태들을 저장하는 동일한 플립 플롭들의 이용을 요구하기 때문에, 매우 침입적이고, 즉 DUT의 현재 상태는 스캔 테스팅시 잃게 된다. 또한, LBIST는 스캔 테스팅에서 의사 랜덤 패턴들(pseudo random patterns)의 이용을 통해 DUT의 전체 안정성을 테스트하기 위한 편리한 방법이다. LBIST는 DUT에 의사 랜덤 수들에 기초하지 않는 스캔 스티뮬러스를 제공한다. 이전에 언급된 스캔 스티뮬러스의 이용을 통해 획득된 장애 적용 범위는 비선형 범위 그래프로 나타내어질 수 있고, 예를 들면, 테스트 적용 범위의 증가는 어떤 수의 클록들 후에 빠르게 드롭 오프(drop off)하기 시작한다. 획득된 실제 적용 범위는 LBIST 생성 동안 이용되는 툴에 의해 결정되고 합격/불합격 상태(pass/fail status)는 예상된 시그니처와 실제 시그니처를 비교함으로써 결정된다. 일반적으로 높은 장애 범위를 얻기 위한 두 개의 방법들이 있다: 관리(controllability) 및/또는 식별가능성(observability) 지점들을 설계에 삽입하는 것 또는 더 긴 시퀀스를 반복. 그러므로, LBIST 및 스캔 테스팅은 일반적으로 장치들을 최종 사용자들에 배송하기 전에(또한, 최종 사용자에 의한 정상 동작 전에) 공장에서 행해지고, 여기서 테스팅이 침입적인지의 여부와 무관하다.

<32> 도 2 및 도 3에 관하여 기재될 바와 같이, 본 발명의 실시예는 정상 동작 동안 이러한 침입적 테스팅 형식을 고려한다. 즉, 처리기(12)가 최종 사용자에 의해 이용될 때와 같이 정상 동작 동안 스캔 테스트되게 하는 소프트웨어가 기록될 수 있다. 예를 들면, 일 실시예에서, 정지 및 테스트 명령과 같은 소프트웨어 명령은 처리기(12)의 스캔 테스팅을 시작하기 위해 사용자 애플리케이션에 이용될 수 있다. 대안으로, 소프트웨어 명령은 처리기(12)의 스캔 테스팅을 일으키는 몇몇 동작이 될 수 있다. 예를 들면, 비트를 특정 레지스터 또는 어드레스 위치를 기록하는 명령은 처리기(12)의 스캔 테스팅을 시작할 수 있다. 이러한 방식으로, 처리기(12)에서 소프트웨어 실행은 그 자신이 처리기(12)의 스캔 테스팅을 시작하는 정지 및 테스트 표시자의 발생이 될 수 있다. 그러므로, 모든 또는 대부분의 처리기(12)의 전체 신뢰 레벨은 본 기술에서 현재 실행된 바와 같이 복잡한 소프트웨어 테스트 기능들을 기록할 필요가 없이 정상 동작 동안 소프트웨어 제어를 통해 달성될 수 있다. 상기된 바와 같이, 이들 소프트웨어 테스트 기능들은 소프트웨어를 갖는 처리기(12)의 부분들만 테스트되고, 스캔 테스팅과 다른 제한된 적용 범위만 제공된다.

<33> 또한, 도 1에 예시된 바와 같이, 처리기(12)는 정지 및 테스트 절차가 수행될 때를 결정하기 위해 데이터 처리 시스템(10) 내, 다른 모듈들(16)과 같은 다른 모듈들을 모니터링하는 다른 모듈 모니터(50)를 포함할 수 있다. 그러므로, 일 실시예에서, 다른 모듈 모니터(50)는 데이터 처리 시스템(10)의 하나 이상의 모듈들의 특정 활동성 또는 활동성들에 응답하여 정지 및 테스트 표시자를 처리기(12)에 제공할 수 있다. 정지 및 테스트 표시자에 응답하여, 처리기(12)의 모두 또는 일부의 스캔 테스팅이 시작될 수 있고, 처리기(12) 외부의 회로의 일부분들은 처리기(12) 대신 또는 그에 부가하여 테스트될 수 있다. 예를 들면, 다른 모듈 모니터(50)는 또한 정지 및 테스트 절차 동안 다른 모듈들(16) (또는 다른 모듈들(16)의 일부들)이 스캔 테스팅되는 것에 관한 표시를 제공할 수 있다.

<34> 도 2 및 도 3의 설명들은 DUT인 처리기(12)에 관하여 제공될 것임을 주의하라. 그러나, 여기의 설명들은 또한 (처리기(12)의 하나 이상의 로직 블록들과 같은) 처리기(12)의 어떤 일부인 DUT 또는 데이터 처리 시스템(10)의 어떤 다른 일부에 적용한다. 즉, 도 2의 스캔 체인들은 처리기(12)이기보다는 데이터 처리 시스템(10)의 몇몇 다른 모듈일 수 있다. 일 실시예에서, 정지 및 테스트 회로(14)는 공유되고 데이터 처리 시스템(10)의 다른 모듈들 중 어떤 것에 대해 테스팅 처리기(12)에 이용될 수 있거나, 개별 정지 및 테스트 회로는 여기에 논의되는 방식으로 정상 동작 동안 스캔 테스트될 수 있는 다른 모듈들 및 처리기(12)를 위하여 이용될 수 있다.

<35> 도 3은 본 발명의 일 실시예에 따른 정상 동작 동안 스캔 테스팅의 방법을 예시한다. 쉬운 설명을 위해, 도 1 및 도 2에 관하여 기재될 것이다; 그러나, 이는 어떤 시스템 및 DUT에도 적용될 수 있고, 꼭 데이터 처리 시스템(10) 및 처리기(12)에만 적용될 수 있는 것은 아니다. 흐름은 블록(200)에서 개시되고, 여기서 처리기(12)를 포함하는 데이터 처리 시스템(10)이 제조된다. 흐름은 블록(202)으로 진행되고, 여기서 데이터 처리 시스템(10)의 공장 테스팅이 수행된다. 이러한 공장 테스팅은 예를 들면 스캔 테스팅, LBIST, 또는 다른 형식의 테스팅을 포함할 수 있고, 일반적으로 어떤 최종 사용자에 시스템을 공급하기 전에 수행된다. 이러한 공장 테스팅은, 장치가 최종 사용자 애플리케이션에서 아직 이용되지 않았기 때문에 침입적일 수 있다(즉, 장치의 상태를

스크램블링할 수 있다). 또한, 이러한 공장 테스팅을 위해 일반적으로 더 많은 시간이 이용될 수 있다. 또한, 이러한 공장 테스팅의 타이밍은 최종 사용자 애플리케이션에 의해 제어될 필요가 없다.

<36> 이후 흐름은 블록(204)으로 진행되고, 여기서 데이터 처리 시스템(10)은 최종 사용자 애플리케이션에 포함될 수 있다. 예를 들면, 데이터 처리 시스템(10)은 자동차, 휴대용 장치 등에 포함될 수 있다. 일 실시예에서, 블록(202)에 기재된 공장 테스팅은 처리기(12)에서만 수행될 수 있다. 이러한 경우, 이후 블록(204)에서 처리기(12)는 데이터 처리 시스템(10)과 같은 데이터 처리 시스템에 포함될 수 있고 또한 최종 사용자 애플리케이션에 포함될 수 있다. 일 실시예에서, 데이터 처리 시스템(10)은 그 자체가 최종 사용자 애플리케이션일 수 있다.

<37> 흐름은 지점 A를 통해 블록(204)으로부터 블록(206)으로 진행하고, 여기서 사용자 애플리케이션이 실행된다. 예를 들면, 처리기(12)가 자동차와 같은 최종 사용자 애플리케이션일 때, 사용자 애플리케이션은 처리기(12)에서 실행된다. 이러한 점에서, 데이터 처리 시스템(10)은 공장 밖에 있는 최종 사용자의 손에 있다. 이후 흐름은 판정 다이아몬드(208)로 진행하고, 여기서 사용자 애플리케이션이 정지 및 테스트 절차를 표시했는지의 여부가 결정된다. 상기에 논의된 바와 같이, 이는 다양한 방법들로 사용자 애플리케이션에 의해 행해질 수 있다. 일 실시예에서, 사용자 애플리케이션내 정지 및 테스트 소프트웨어 명령은 정지 및 테스트 절차를 표시하기 위해 디코딩되었을 수 있다. 대안으로, 특정 레지스터 또는 메모리 위치에 기입하는 사용자 애플리케이션내 소프트웨어 명령은 정지 및 테스트 절차를 표시하는 처리기(12)내 회로에 의해 검출되었을 수 있다. 어느 경우에나, 정지 및 테스트 절차가 처리기(12)내에 표시될 때, 정지 및 테스트 표시자(24)는 처리기(12)로부터 정지 및 테스트 회로(14)로 제공된다. 전자의 정지 및 테스트 소프트웨어 명령의 실시예에서, 처리기(12)의 명령 디코더는 정지 및 테스트 표시자(24)를 제공할 수 있고, 후자의 특정 위치에 대한 기록의 실시예에서, 특정 위치를 폴링(poll)하거나 이와는 달리 기록을 검출하는 회로는 정지 및 테스트 표시자(24)를 제공할 수 있다. (대안으로, 정지 및 테스트 절차는 상기에 기재된 바와 같이 다른 모듈 모니터(50)로 표시될 수 있다.)

<38> 도 3으로 되돌아가서 참조하면, 사용자 애플리케이션이 정지 및 테스트 절차를 표시하지 않은 경우, 흐름은 지점 A로 돌아가서, 사용자 애플리케이션들의 실행을 계속한다. 정지 및 테스트 절차가 표시되는 경우, 흐름은 블록(210)으로 진행하고, 여기서 사용자 애플리케이션의 실행이 정지되고 처리기(12)의 상태가 저장된다. 필요한 경우, 처리기(12)의 플립 플롭들의 모두 또는 일부의 값들은 정지 및 테스트 절차 후 동일 지점으로부터 실행을 계속할 수 있도록 저장될 수 있다. 대안으로, 비록 사용자 애플리케이션은 정지 및 테스트 절차가 표시될 때 유휴 상태에 있을 수 있지만, 상태가 저장되기를 요구하지 않는다. 또 다른 대안의 실시예에서, 단지 수개의 플립 플롭들이 저장될 필요가 있고, 따라서 시간 및 리소스들을 절약한다. 이러한 대안의 실시예에서, 소프트웨어는 처리기(12)의 부분들이 저장되는지를 표시하기 위해 이용될 수 있다. 또한, 상태들의 저장이 요구되는 경우, 추가의 소프트웨어 명령들이 어떤 플립 플롭들이 저장되고 어디에 저장되었는지를 표시하기 위해 사용자 애플리케이션에 제공될 수 있거나, 추가의 소프트웨어 명령은 제공될 수 있고, 그것은 모든 플립 플롭들이 저장된 것을 표시한다. 대안으로, 정지 및 테스트 절차의 표시시, 특수화한 하드웨어 회로는 플립 플롭들의 모두 또는 일부의 현재 상태의 빠른 저장을 수행하기 위해 이용될 수 있다.

<39> 상태가 선택적으로 저장된 후, 흐름은 블록(212)으로 진행되고, 여기서 정지 및 테스트 표시자(24)는 예를 들면 정지 및 테스트 회로(14)에 제공된다. 흐름은 블록(214)으로 진행하고, 여기서 리셋이 수행된다. 예를 들면, 정지 및 테스트 회로(14)는 처리기(12)의 플립 플롭들 및 예를 들면, 랜덤 패턴 생성기(102), 시그니처 분석기(112), 및 테스트 결과 레지스터(118)와 같은 데이터 처리 시스템(10) 내 리셋이 필요한 다른 회로에 리셋(48)을 제공한다. 이는 예를 들면, 정지 및 테스트 절차가 공지된 값들로 시작된다는 것을 보장한다. 일 실시예에서, (리셋(48)과 같은) 리셋은 모든 것을 리셋하지 않고, 예를 들면, 소프트웨어가 레지스터들을 초기화하기 위해 이용될 수 있다.

<40> 이후 흐름이 블록(216)으로 진행되고, 여기서 클록 카운터 검사 지점은 현재 검사 지점으로 선택된다. 즉, 테스팅이 사용자 애플리케이션의 실행 동안과 같은 정상 동작 동안 수행되고 있기 때문에, 정지 및 테스트 절차가 표시되는 각각의 시간에 완전한 테스팅이 허용될 시간이 없을 수 있다. 그러므로, 테스트 결과들은, 특정한 수의 클록들이 먼저 달성된 경우, 비교되고 분석되기 위해서만 필요하다. 다음은, 예를 들면, LBIST에 의해 제공된 DUT의 장애 적용 범위에 대해 클록들의 수에 관련하는 샘플 테이블이다. (다음 테이블은 예로서만 제공되고, 여기서 실제값들은 설계에 따라 변할 것이다.)

<41>

클록들의 수	적용 범위

<42>

20	5%

<43> 100 10%

<44> 500 30%

<45> 2000 70%

<46> 그러므로, 500 클록들이 제공될 수 있는 경우, 30% 적용 범위가 달성되는 등이다. 그러므로, 일 실시예에서, 테스트 결과들은 특정 클록 검사 지점들을 달성할 때만 업데이트된다. 일 실시예에서, 클록 검사 지점들은 예를 들면, 5%, 10%, 30% 등 바람직한 적용 범위 레벨들에 따라 설정되고, 테스트 결과들은 (5%에 대응하는) 20 클록들 후, (10%에 대응하는) 100 클록들 후, (30%에 대응하는) 500 클록들 후 등에 업데이트된다. 또한, 현재 실시예에서, 상기 테이블의 클록들의 수는 (상기된 바와 같이) 함수 클록 사이클들을 지칭하고, 여기서 테스트 입력/출력 시프트 사이클들에 대해 제공된 클록들은 카운트되지 않는다. 그러나, 대안의 실시예에서, 테이블은 클록들의 수가 제공된 모든 클록들을 나타내도록 설정될 수 있다. 그러므로, 블록(216)에서, 제 1 클록 카운트 검사 지점이 예를 들면 20 클록들과 같이 선택된다.

<47> 흐름은 블록 218로 진행하고, 여기서 테스트 스티뮬러스가 SE(138)이 확정될 때 x 클록들에 대한 시그니처 분석기(1112)에 테스트 결과들을 시프트 아웃하는 동안 시프트 인된다. 상기된 바와 같이, 어떤 수 x의 테스트 입력/출력 시프트 사이클들이 수행될 수 있고, 여기서 x는 1 이상의 어떤 정수일 수 있다. 또한, 예시된 실시예에서, LBIST는 정지 및 테스트 절차 동안 수행되고 있다. 그러므로, 도 2를 참조하면, 정지 및 테스트 표시자(24)에 응답하여, 정지 및 테스트 제어기(100)는 (일 실시예에서, 의사 랜덤 패턴 생성기인) 랜덤 패턴 생성기(102)에 시작(160)을 제공하고 SE(138)을 확정한다. 시작(160)에 응답하여, 랜덤 패턴 생성기(102)는 Sin1(108) 및 Sin2(110)을 통해 의사 랜덤 입력들을 스캔 체인들에 제공하는 것을 시작한다. (이들 입력들은 Sin1(108) 및 Sin2(110)을 통해 제공되고, 출력들은 또한 Sout1(109) 및 Sout2(111)를 통해 시프트 아웃되고 있다는 것을 주의하라.)

<48> 흐름은 블록(220)으로 진행하고, 여기서 시스템 클록(예를 들면, 클록(104))은 테스트 출력들이 블록(218)으로 시프트 인되는 테스트 스티뮬러스에 기초하여 스캔 체인들에 캡처링되도록 (함수 클록 사이클을 제공하도록) SE(138)가 무효화될 때 제공된다. 또한, 클록 카운트가 증가되고, 함수 사이클들에 대해 제공된 클록들의 트랙을 유지한다. 일 실시예에서, 이러한 클록 카운트는 테스트 결과 레지스터(118)에 보존되고, 여기서 테스트 결과 레지스터(118)는 예를 들면 클록 카운트 필드(clock count field)를 포함할 수 있다. 즉, 일 실시예에서, 테스트 결과 레지스터(118)는 클록 사이클들을 정량화하는 값을 저장할 수 있다. 대안으로, 이러한 클록 카운트는 이하에 더 설명되는 바와 같이 데이터 처리 시스템(10) 내 어느 곳에도 저장될 수 있고, 테스트 결과 레지스터(118)의 클록 카운트 필드는 클록 카운트 검사 지점이 도달될 때 클록 카운트만을 저장할 수 있다.

<49> 흐름은 판정 다이아몬드(221)로 진행하고, 여기서 인터럽트가 수신되었는지의 여부가 결정된다. 예를 들면, 일 실시예에서, 데이터 처리 시스템(10)은 인터럽트가 데이터 처리 시스템(10)에 수신될 때를 결정하는 인터럽트 제어기(20)를 포함한다. 인터럽트가 수신될 때,인터럽트 표시자(22)를 정지 및 테스트 회로(14)에 제공한다. 현재 실시예에서, 어떤 인터럽트는 최종 및 정상 동작에 대해 정지 및 테스트 절차가 재개하게 한다. 대안으로, 단지 특정 인터럽트들 또는 특정 우선 순위 레벨의 인터럽트들은 최종 및 정상 동작에 대해 정지 및 테스트 절차가 재개하게 할 수 있다. 다시 도 3을 참조하면, 인터럽트가 수신되었을 경우(인터럽트 표시자(22)에 의해 정지 및 테스트 회로(14)에 표시된 경우), 흐름은 지점 B를 통해 블록(234)으로 진행하고, 여기서 다른 리셋이 수행된다. 이러한 리셋은 블록(214)에 관하여 설명된 리셋과 유사하다. 흐름은 블록(236)으로 진행하고, 여기서 실행은 복원 벡터 위치에서 시작한다. 즉, 복원 벡터는 처리기(12)에 의한 실행이 재개하는 곳을 표시하기 위해 이용될 수 있다. 이러한 복원 벡터는 도 2의 복원 벡터(152)일 수 있고, 정지 및 테스트 제어기(100)(또는 정지 및 테스트 회로(14) 중 어떤 곳)에 저장될 수 있다. 대안으로, 이러한 복원 벡터의 효과는 처리기(12)의 로직에 삽입될 수 있고, 이러한 경우, 정지 및 테스트 제어기(100)의 복원 벡터(152)는 불필요할 것이다. 이후 흐름은 블록(238)으로 진행하고, 여기서 상태가 블록(210)에 저장될 경우, 상태는 복원된다. 이후 흐름은 지점 A로 리턴하고, 여기서 사용자 애플리케이션들은 실행을 계속한다.

<50> 판정 다이아몬드(221)에서, 인터럽트가 제 1 클록 검사 지점에 도달하기 전에 수신된 경우, 테스트 결과들은 판정 다이아몬드(208)로부터 블록(210)으로의 yes 분기로 표시된 정지 테스트 절차에 대해 저장되지 않을 수 있다. 판정 다이아몬드(221)에서, 인터럽트가 수신되지 않았을 경우, 흐름은 판정 다이아몬드(222)로 진행하고, 여기서 (예를 들면, 테스트 결과 레지스터(118)의) 클록 카운터가 제 1 클록 카운트 검사 지점(즉, 현재 검사 지점)에 도달되었거나 초과되었는지의 여부가 결정된다. (일 실시예에서, 상기된 바와 같이, 테스트 결과 레지스터(118)의 클록 카운트 필드는 클록 카운트 검사 지점이 도달될 때, 클록 카운트만을 저장하고, 따

라서, 테스트 결과 레지스터(118)는 항상 도달된 최신의 클록 카운트 검사 지점을 나타낼 수 있다는 것을 주의하라. 이러한 실시예에서, 블록(220)에서 증가되고 판정 다이아몬드(222)에서 현재 검사 지점이 도달되거나 초과되었는지를 결정하기 위해 이용되는 실제 클록 카운트는, 처리기(12) 또는 데이터 처리 시스템(10)의 어느 곳에도 저장될 수 있거나, 테스트 결과 레지스터(118)의 다른 필드에 저장될 수도 있다.) 판정 다이아몬드(222)에서, 현재 클록 카운트 검사 지점이 도달되거나 초과되지 않았을 경우, 현재 검사 지점에 도달을 시도하기 위하여 흐름은 더 많은 테스트 스티뮬러스가 시프트 인되고 더 많은 테스트 결과들이 시프트 아웃되는 블록(218)으로 리턴하고, 다른 함수 클록 사이클이 수행될 블록(220)으로 리턴한다.

<51> 판정 다이아몬드(222)에서, 클록 카운트가 현재 검사 지점에 도달하거나 초과된 경우, 흐름은 블록 224로 진행되고, 여기서 시프트 아웃된 테스트 출력들로부터 계산된 시그니처(예를 들면, Sout1(109) 및 Sout2(111))를 예상된 시그니처와 비교하여, 이에 따라 합격/불합격 표시자가 조정된다. 일 실시예에서, 테스트 결과 레지스터는 또한 테스트들이 계속 합격을 표시하는 한 로직 레벨 1을 유지하고, 테스트가 불합격될 때(즉, 테스트 결과들이 예상된 시그니처와 매칭하지 않을 때) 로직 레벨이 제로가 되는 단일 비트일 수 있는 합격/불합격 표시자들을 포함한다. 대안으로, 합격/불합격 표시자는 로직 레벨 1이 상기 테이블의 각각의 클록 카운트 검사 지점에 대하여 테스트 결과 레지스터(118)내에 제공될 수 있고, 여기서 로직 레벨 1이 대응하는 검사 지점에서 합격을 표시할 것이고, 로직 레벨 0은 대응하는 검사 지점에서 불합격을 표시할 것이고, 이와 반대로도 가능하다. 대안으로, 합격/불합격 결과들을 저장하는 다른 방법들이 이용될 수 있다.

<52> 도 2를 참조하면, Sout1(109) 및 Sout2(111)을 통해 스캔 체인들로부터 시프트 아웃된 테스트 결과들이 시그니처 분석기(112)에 제공된다는 것을 주의하라. 시그니처 분석기(112)는 이들 출력들에 기초하여 시그니처(120)를 생성하고 정지 및 테스트 제어기에 의해 비교를 인에이블하기 위해 제공되는 비교(126)에 기초하여 시그니처 비교기(114)에 시그니처(120)를 제공한다. 또한, 예상된 시그니처 저장소(116)는 비교를 위해 예상된 시그니처(115)를 시그니처 비교기(114)에 제공한다. 예상된 시그니처 저장소(116)로부터 예상된 시그니처는 정지 및 테스트 제어기(100)로부터 수신된 시그니처 식별자(124)에 기초하여 선택된다. 그러므로, 정지 및 테스트 제어기(100)는 랜덤 패턴 생성기(102)를 제어하고, 이에 따라 예상된 시그니처의 선택을 제어한다. 본 기술에 공지된 종래의 의사 랜덤 패턴 생성기 및 시그니처 분석기가 이용될 수 있다는 것을 주의하라. 또한, 저장소(116)에 저장된 예상된 시그니처들은 또한 본 기술에 공지된 바와 같이 생성될 수 있다. 예를 들면, 예상된 시그니처들은 DUT의 시뮬레이션에 의해 계산될 수 있다.

<53> 또한 도 2를 참조하면, 시그니처 비교기(114)는 시그니처(120)와 예상된 시그니처(115)를 비교하고 비교 결과(122)를 테스트 결과 레지스터(118)에 제공한다. 이후, 테스트 결과 레지스터(118)의 적합한 합격/불합격 표시자는 비교 결과(122)를 반영하기 위해 필요되는 만큼 업데이트될 수 있다. 예를 들면, 비교 결과(122)가 매칭을 표시하는 경우, 합격/불합격 표시자는 (로직 레벨 1로 설정되는 것과 같이) 합격을 표시하도록 설정된다. 그러나, 비교 결과(122)가 매칭하지 않는 것을 표시하는 경우, 합격/불합격 표시자는 (로직 레벨 0으로 클리어되는 것과 같이) 불합격을 표시하도록 설정된다.

<54> 다시 도 3을 참조하면, 흐름은 판정 다이아몬드(226)로 진행하고, 여기서 블록(224)에 비교의 결과로서 합격 또는 불합격이 표시되는지의 여부가 결정된다. (예를 들면, 비교 결과(122)에 의해 표시된 바와 같이 시그니처(120)가 예상된 시그니처(115)와 매칭된 것을 표시하는) 합격이 된 경우, 흐름은 판정 다이아몬드(228)로 진행한다. 판정 다이아몬드(228)에서, 인터럽트가 수신되었는지의 여부가 다시 결정된다. 상기에 기재된 바와 같은 이러한 인터럽트는 인터럽트(22)를 통해 정지 및 테스트 회로(14)에 표시될 수 있다. 판정 다이아몬드(221)에서 인터럽트에 제공된 동일한 설명들이 또한 판정 다이아몬드(228)에 적용된다. 인터럽트가 수신되었을 경우, 흐름은 블록(234)으로 진행하고, 여기서 상기 기재된 바와 같이 흐름은 블록들(234, 236, 238, 206) 등을 통해 진행한다. 그러므로, 인터럽트가 수신될 때, 테스팅은 계속하지 않고, 정상 동작(예를 들면, 사용자 애플리케이션의 재개하는 실행)이 재개한다.

<55> 그러나, 인터럽트가 판정 다이아몬드(228)에서 수신되지 않았다면, 흐름은 판정 다이아몬드(230)로 진행하고, 여기서 최대 클록 카운트가 도달되었는지의 여부가 결정된다. 예를 들면, 일 실시예에서, (예를 들면, 정지 및 테스트 제어기(100) 또는 정지 및 테스트 회로(14)내 어느 곳에 저장된, 도 2의 최대 클록들(154)에 대응할 수 있는) 최대 클록 카운트가 이용될 수 있다. 최대 클록 카운트가 달성되었을 경우, 인터럽트가 수신되었는지의 여부가 결정되도록 흐름은 판정 다이아몬드(228)로 리턴한다. (또한, 이하에 더 논의될 바와 같이, 블록(227)에서와 같이, 불합격이 표시되었을 경우, 흐름은 또한 판정 다이아몬드(228)에 리턴한다.) 최대 클록 카운트가 달성되었을 경우, 테스팅은 계속되지 않고, 대신 흐름음 판정 다이아몬드들(228, 230) 사이에서 인터럽트가 수신될 때까지 계속한다. 이 지점에서, 흐름은 판정 다이아몬드(228)로부터 블록(234)으로 진행하고, 여기서 상

기된 바와 같이 정상 동작이 재개한다. (대안의 실시예에서, 최대 클록 카운트가 달성되었을시, 정지 및 테스트 절차는 정지하고 흐름은 블록(234)으로 진행하고, 여기서 사용자 애플리케이션의 실행이 인터럽트를 기다리지 않고 재개한다.) 판정 다이아몬드(230)에서, 최대 클록 카운트가 아직 달성되지 않았을 경우(그리고 인터럽트가 아직 정지 및 테스트 절차를 정지하기 위해 수신되지 않았기 때문에), 흐름은 블록(232)으로 진행하고, 여기서 다음 클록 카운트 검사 지점은 현재 검사 지점으로서 선택된다. 예를 들어, 상기 테이블을 참조하면, 다음의 클록 카운트 검사 지점은 100 클록들의 다음의 엔트리를 지칭할 수 있다. 즉, (블록(224)에서) 테스트 결과들이 업데이트될 다음의 시간은 현재 클록 카운트가 인터럽트가 수신되기 전에 100 클록들에 도달하는 경우일 수 있다.

<56> 일 실시예에서, 예를 들면, 충분한 수의 함수 클록들이 제공되기 전에 인터럽트가 발생되었기 때문에 클록 카운트가 클록 카운트 검사 지점에 도달하지 않은 경우, 다음의 정지 및 테스트 절차는 시작에서부터 다시 시작되어야 하고, 예를 들면 다음 정지 및 테스트 절차를 위한 현재 클록 카운트 검사 지점은 (상기에 주어진 동일한 테이블에서 20 클록들인) 제 1 클록 검사 지점으로 다시 설정된다.

<57> 다시 판정 다이아몬드(226)를 참조하면, 불합격이 발생한 경우(예를 들면, 비교 결과(122)에 의해 표시된 바와 같이 시그니처(120)가 예상된 시그니처(150)와 매칭하지 않는 것을 나타내는 경우), 흐름은 블록(227)으로 진행하고, 여기서 불합격이 표시된다. 불합격의 표시는 블록(224)에 관하여 상기에 기재된 테스트 결과 레지스터(118)의 합격/불합격 표시자의 조정을 지칭할 수 있거나, 또한 불합격을 표시하는 추가의 출력을 제공하는 것을 포함할 수 있다는 것을 주의하라. 이후 흐름은 판정 다이아몬드(228)로 진행하고, 여기서 인터럽트가 수신되었는지의 여부가 결정된다. 흐름은 판정 다이아몬드(228)에 관하여 상기에 이미 기재된 바와 같이 진행한다. 예시된 실시예에서, 흐름은 인터럽트가 수신될 때까지 판정 다이아몬드들(228, 230) 사이에서 계속하여, 흐름이 블록(234)으로 계속하게 하고, 따라서 정상 동작이 재개한다(예를 들면, 사용자 애플리케이션의 실행의 재개)는 것을 주의하라. 대안의 실시예에서, 판정 다이아몬드(230)에서, 최대 클록 카운트가 달성된 경우 또는 불합격이 표시되었을 경우, 흐름은 즉시 블록(234)으로 계속할 수 있고 사용자 애플리케이션의 실행은 정지 및 테스트 절차를 종료하기 위해 인터럽트를 기다리지 않고 재개할 수 있다.

<58> 대안적인 실시예에서, 도 3의 흐름은 또한 최소 클록 검사를 포함할 수 있다. 예를 들면, 도 2에 도시된 바와 같이, 정지 및 테스트 제어기(100)(또는 다른 경우로는 정지 및 테스트 회로(14))에 저장된 최소 클록들(156)은 정지 및 테스트 절차 동안 이용될 수 있다. 일 실시예에서, 최소의 클록들의 수는 어떤 결과들이 업데이트되기 전에 수신된다. 다른 실시예에서, 최소의 클록들의 수는 (판정 다이아몬드(221)에서와 같이) 인터럽트가 수신되는 경우를 검사하기 전에 수신된다. 또한, 다른 대안의 실시예에서, 최대 클록들 카운트가 이용될 수 있다. 이러한 경우에는, 판정 다이아몬드(230)는 단지 불합격이 발생되었는지의 여부에 대한 검사를 포함할 것이다.

<59> 도 3은 정지 및 테스트 절자 동안 수행되는 LBIST 테스팅에 관하여 개시된다. 그러나, 대안의 실시예들에서, 다른 형식들의 스캔 테스팅이 수행될 수 있다. 대안으로, 스캔 테스팅과 다른 또는 그에 더하여 처리기의 상태를 스크램블링하는 테스팅을 포함하는 다른 테스팅의 형식들이 정지 및 테스트 절차 동안 수행될 수 있다. 이들 경우들에서, 정지 및 테스트 회로(14)는 테스팅의 바람직한 형식 또는 그 형식들을 달성하기 위해 도 2에 도시된 것으로부터 다른 회로 또는 추가의 회로를 포함할 수 있다. 또한, 도 3의 흐름은 일 실시예이고, 대안의 흐름들이 추가의 흐름들은 추가의 단계들을 포함하거나, 더 적은 단계들을 포함하거나, 또한 단계들을 조합하여 이용될 수 있다는 것을 주의하라.

<60> 또한, 상기된 설명들은 테스트 스티뮬러스가 스캔 체인들로 이동되고 테스트 출력들이 스캔 체인들의 밖으로 이동된다. 그러나, 대안의 실시예들에서, 테스트 스티뮬러스는 다르게 적용될 수 있다. 예를 들면, 그들은 스캔 체인 또는 스캔 체인들에 병렬로 로딩될 수 있다. 유사하게는, 스캔 테스트 출력들은 스캔 체인 또는 스캔 체인들로부터 병렬로 판독될 수 있다.

<61> 그러므로, 예를 들면, 도 3의 흐름도와 같이, 본 발명의 얼마나 다양한 실시예들이 데이터 처리 시스템, 처리기, 또는 그의 일부분들이 공장을 벗어난 후 침입적으로 테스팅될 수 있는 정상 동작 동안 및 최종 사용자 애플리케이션과 같은 정상 동작 동안 규제된 침입적 테스팅을 허용하는지가 인식될 수 있다. 이는 처리기 자체가 유휴(idle)이거나 이와는 달리 침입적으로 테스트가 가능할 때의 "인식을 갖는" 처리기(12)의 소프트웨어이기 때문에, (예를 들면, 정지 및 테스트 표시자(24)의 이용을 통해) 침입적 테스팅이 수행되는 방법 및 시간을 제어할 수 있게 하는 처리기(12)에서 구동하는 소프트웨어에 대한 능력은 최종 사용자 애플리케이션의 정상 동작 동안 개선된 테스팅을 허용할 수 있다. 유사하게는, 이는 일반적으로 버스(26)에 결합된 주변 장치 또는 보조 처리 장치가 유휴이거나 유휴일 수 있을 때의 인식을 갖는 (처리기(12) 또는 버스(26)에 결합된 다른 처리기

상에 구동하는) 소프트웨어이다. 예를 들면, 소프트웨어는, 프로토콜을 수행하거나 흐름 제어에 의해, 시스템 능력을 모니터링함으로써 이러한 인식을 가질 수 있다. 그러므로, 소프트웨어는 (침입적 테스팅을 통해) 장치의 상태가 스크램블링될 수 있고 이후 (예를 들면, 리셋 및 복원 동작을 통해) 유기적인 방법으로 그로부터 회복될 때를 인식하기 때문에, 소프트웨어는 처리기(12) 또는 데이터 처리 시스템(10)의 다른 부분들, 또는 둘 모두의 침입적 테스팅을 수행할 수 있다. 일 실시예에서, 처리기(12)로부터의 정지 및 테스트 표시자(24)는 또한 데이터 처리 시스템(10)의 모듈 또는 일부가 침입적으로 테스트될 수 있다는 것을 표시할 수 있다.

<62> 도 2의 예시된 실시예에서, 정지 및 테스트 제어기(100)는 랜덤 패턴 생성기(102)에 강제된 에러 신호(158)를 제공한다. 이러한 신호는 테스팅 회로 자체가 올바르게 동작하고 있는 것을 테스트하고 보장하기 위해 이용될 수 있다. 예를 들면, 강제된 에러 신호(158)는 불합격이 보장되는 것(적어도 하나의 시그니처는 예를 들면, 랜덤 패턴 생성기(102)의 출력에 변경을 일으킴으로써 대응하는 예상된 시그니처에 매칭하지 않는 것)으로 설정될 수 있다. 그러나, 테스트 결과 레지스터(118)가 불합격을 표시하지 않는 경우, 테스팅 회로 그 자체가 문제일 수 있다. 그러므로, 강제된 에러 신호(158)는 테스팅 회로 그 자체의 적합한 동작을 보장하기 위해 공지된 결과를 생성하기 위한 방법을 제공한다. 소프트웨어 명령은 이러한 강제된 에러 신호를 제공하기 위해 이용될 수 있거나, 또는 대안으로, 이는 적합한 동작을 보장하기 위해 주기적으로 하드웨어에 의해 행해질 수 있다.

<63> 도 4는 처리기(12)에 제시될 수 있는 스캔 체인의 대안적인 실시예를 도시한다. 도 4의 스캔 체인은 입력들을 더 먼 스캔 체인들의 단부들로 더 빠르게 전달하기 위해 추가의 XOR 게이트들을 이용한다. 도 4는 (결합 로직(312, 314, 316, 318, 320) 각각으로부터의 입력들을 수신하는) 정상 동작시 동작하도록 구성될 수 있고, 스캔 체인(다른 플립 플롭으로부터 수신된 입력들)으로서 동작하도록 구성될 수 있는 플립 플롭들(301-306)을 포함한다. 도 4의 일 실시예에서, 각각의 플립 플롭은 도 2의 플립 플롭들의 입력들에 위치되는 멀티플렉서들의 기능을 내부에 포함한다는 것을 주의하라. 도 4의 각각의 플립 플롭은 스캔 인에이블(SE) 및 클록 입력을 수신한다. SE가 무효화될 때, 각각의 플립 플롭은 "D" 입력을 통해 수신된 입력들에 작용하여, 정상 동작을 허용한다. SE가 확정될 때, 각각의 플립 플롭은 스캔 체인으로서 동작하고, 여기서 입력들은 "SD" 입력을 통해 스캔 체인으로 전달된다. 비록 도 4에 도시되지 않았지만, 플립 플롭들의 출력들은 또한 결합 로직에 제공될 것이다. 이들 동일한 형태의 플립 플롭들은 또한 도 2의 실시예에서 이용될 수 있다는 것을 주의하라.

<64> 도 4의 스캔 체인은 또한 배타적-OR(XOR) 게이트들(308, 310)을 포함한다. XOR 게이트(308)는 플립 플롭(301)의 출력으로부터 제 1 입력을 수신하고, 플립 플롭(303)의 출력으로부터 제 2 입력을 수신한다. XOR 게이트(310)는 플립 플롭(304)의 출력으로부터 제 1 입력을 수신하고, 플립 플롭(305)의 출력으로부터 제 2 입력을 수신한다. XOR 게이트들은 테스트 입력들이 더 적은 수의 클록들을 갖는 스캔 체인의 더 아래로 전달되게 한다. 즉, (예를 들면, 플립 플롭(301)과 같은) 몇몇 플립 플롭들의 출력들이 XOR 게이트를 통해 (예를 들면, 플립 플롭(304)과 같은) 스캔 체인의 더 아래로 플립 플롭들에 피드 포워딩되기(fed forward) 때문에, 이들 포워딩된 값들을 수신하는 플립 플롭들은 이전의 스캔 체인의 플립 플롭들 모두를 통해 전달하기 위해 테스트 입력들을 기다리는 것보다 더 빠르게 상태를 변경할 수 있다. 이러한 방식으로, 스캔 체인의 더 먼 섹션들의 테스팅은 이들 더 먼 플립 플롭들이 상태들을 변경하는 한, 스티뮬러스가 더 많은 회로에 제공되어, 더 많은 테스팅 범위를 허용하기 때문에 더 적은 클록 사이클들로 수행될 수 있다.

<65> 예를 들면, 배타적-NOR(XNOR) 게이트와 같은 다른 로직 게이트들은 XOR 게이트들(308, 310)을 대체하거나 그에 추가하여 이용될 수 있다는 것을 주의하라. 또한, 어떤 수의 플립 플롭들도 게이트들의 입력들 간에 존재할 수 있고, 도 4에 도시된 수에는 제한이 없다. 예를 들면, XOR 게이트들(308, 310)과 같은 게이트들의 배치는 특정 설계에 대한 최선의 스캔 스티뮬러스를 생성하기 위해 선택될 수 있다.

<66> 또한, 처리기(12)의 하나 이상의 스캔 체인들은 제 1 메모리 소자, 메모리 소자들의 체인, 및 제 2 메모리 소자를 포함하는 것으로 기재될 수 있고, 여기서 제 1 메모리 소자의 출력은 메모리 소자들의 체인의 입력 및 적어도 하나의 로직 게이트의 입력에 접속되고, 적어도 하나의 로직 게이트의 출력은 제 2 메모리 소자의 입력에 접속된다. 예를 들면, 메모리 소자들은 스캔 체인 내 플립 플롭들일 수 있고, 여기서 로직 게이트들은 도 4에 관하여 설명된 XOR 게이트들일 수 있다. 메모리 소자들의 체인은 어떤 수의 메모리 소자들도 포함할 수 있다.

<67> 상기 명세에서, 본 발명은 특정 실시예들에 관하여 기재되었다. 그러나, 당업자는 다양한 변형들 및 변경들이 이하 청구항들에 기재된 바와 같이 본 발명의 범위로부터 벗어남이 없이 행해질 수 있다는 것을 인식한다. 예를 들면, 블록도들은 또한 도시된 것들과 다른 블록들을 포함할 수 있거나, 더 많거나 더 적은 블록들을 가질 수 있거나, 더 많거나 적은 단계들을 가지거나, 다르게 구성되거나, 다수의 단계들 또는 단계들로 분리될 수 있는 단계들 또는 다른 것과 동시에 수행될 수 있는 단계들을 가질 수 있는 단계들을 가질 수 있다. 따라서, 명

세서 및 도면들은 제한되는 의미보다는 예시적으로 간주되는 것이고, 모든 이러한 변경들은 본 발명의 범위 내에 포함되도록 의도된다.

<68> 이점들, 다른 이익들, 및 문제들에 대한 해결책들이 특정 실시예들에 관하여 상기에 기재되었다. 그러나, 이익들, 이점들, 문제들에 대한 해결책들 및 어떤 이익, 이점 또는 생성되거나 더 분명하게 되게 하는 해결책을 생성할 수 있는 소자(들)는 청구항들의 일부 또는 모두의 중대하거나, 요구되거나, 필수적인 특징 또는 소자로서 해석되지 않는다. 여기에 사용된 바와 같이, 용어 "포함하다", "포함하는" 또는 그의 다른 변형들은 비-배타적 포함을 적용하도록 의도되고, 소자들의 리스트를 포함하는 공정, 방법, 제품, 또는 장치는 단지 이들 요소들을 포함할 뿐만 아니라 이러한 공정, 방법, 제품, 또는 장치에 명확히 리스팅되지 않거나 TQ입되지 않은 다른 소자들을 포함할 수 있다.

도면의 간단한 설명

<8> 도 1은 본 발명의 일 실시예에 따른 데이터 처리 시스템을 블록도 형식으로 도시하는 도면.

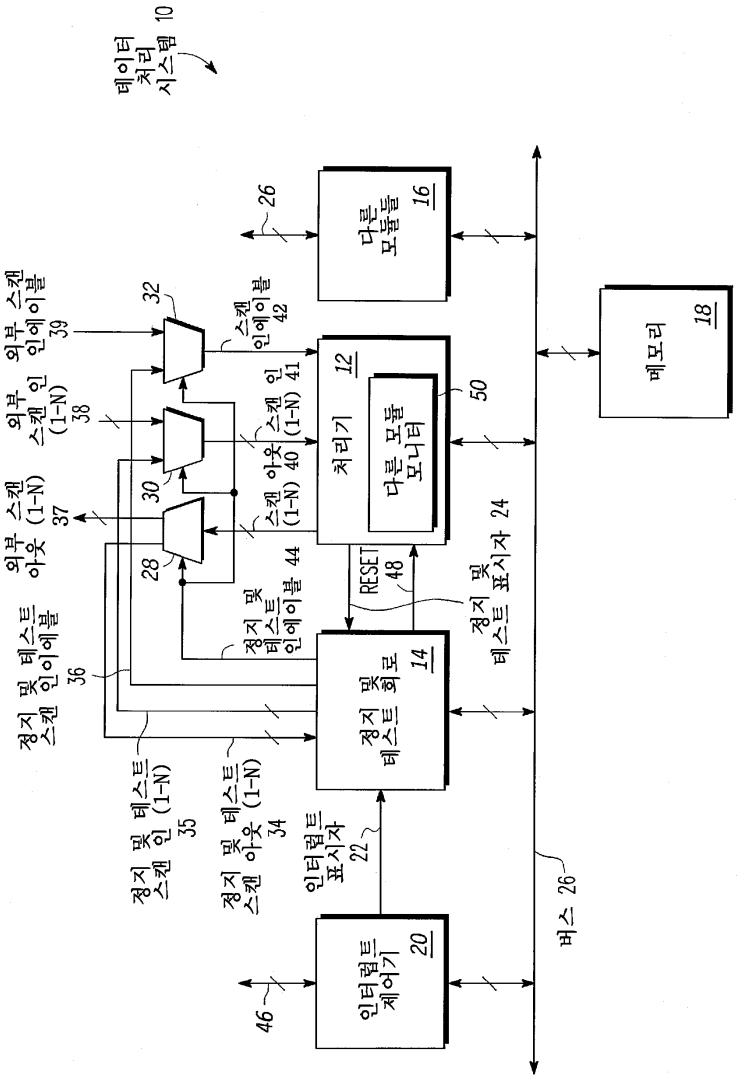
<9> 도 2는 본 발명의 일 실시예에 따른 도 1의 처리기와 정지 및 테스트 회로의 일부분들을 블록도 형식으로 도시하는 도면.

<10> 도 3은 본 발명의 일 실시예에 따른 도 1의 처리기를 테스팅하는 방법을 순서도 형식으로 도시하는 도면.

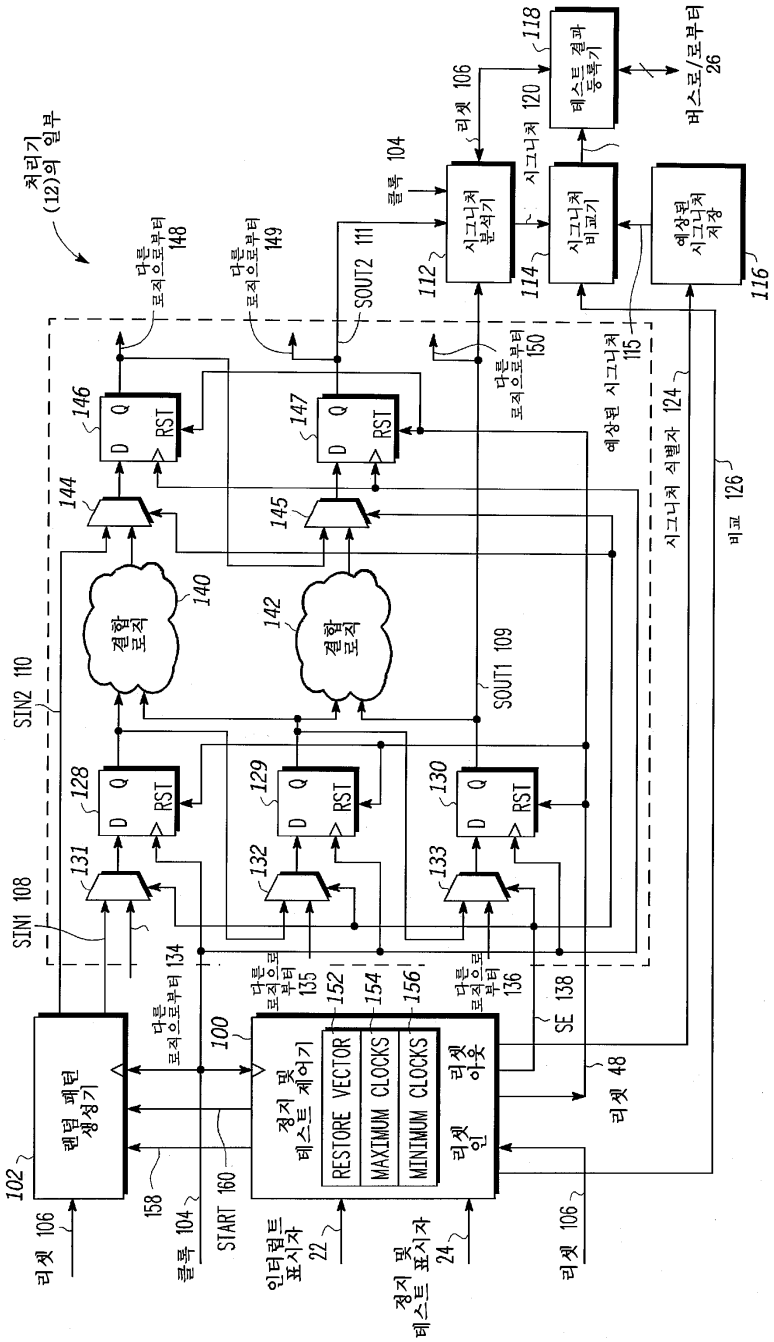
<11> 도 4는 본 발명의 실시예에 따른 도 1의 처리기의 일부분을 블록도 형식으로 도시하는 도면.

도면

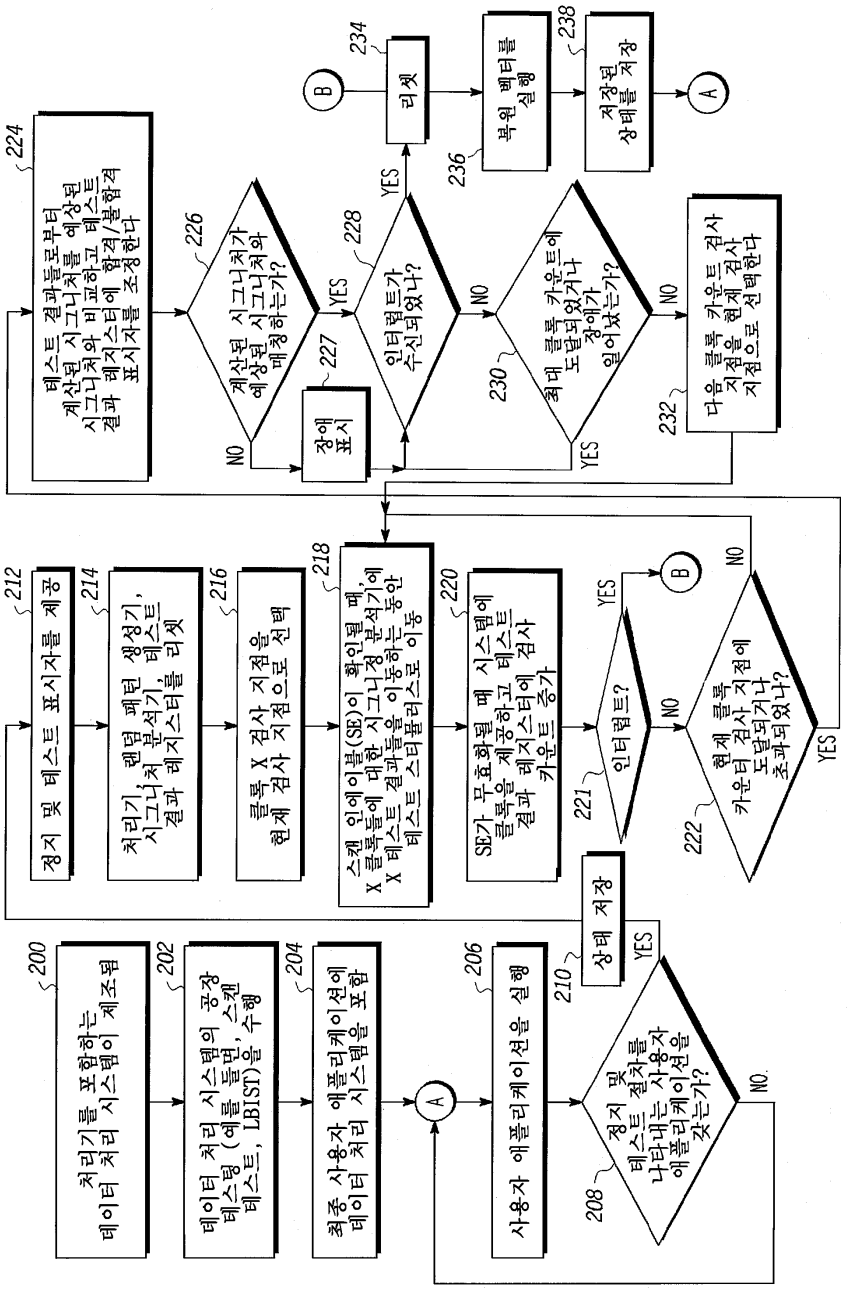
도면1


데이터 처리 시스템 10
다른 모듈들 16
26
메모리 18
처리기 12
다른 모듈 모니터
50
32
스캔 인에이블 42
외부 스캔 인에이블 39
외부 스캔 인 (1-N) 38
외부 스캔 아웃 (1-N) 37
30
28
스캔 인 (1-N) 41
스캔 아웃 (1-N) 40
정지 및 테스트 인에이블 44
RESET 48
정지 및 테스트 표시자 24
정지 및 테스트 인에이블 36
정지 및 스캔 인 (1-N) 35
정지 및 스캔 아웃 (1-N) 34
정지 및 테스트 회로 14
인터럽트 표시자 22
인터럽트 제어기 20
46
버스 26

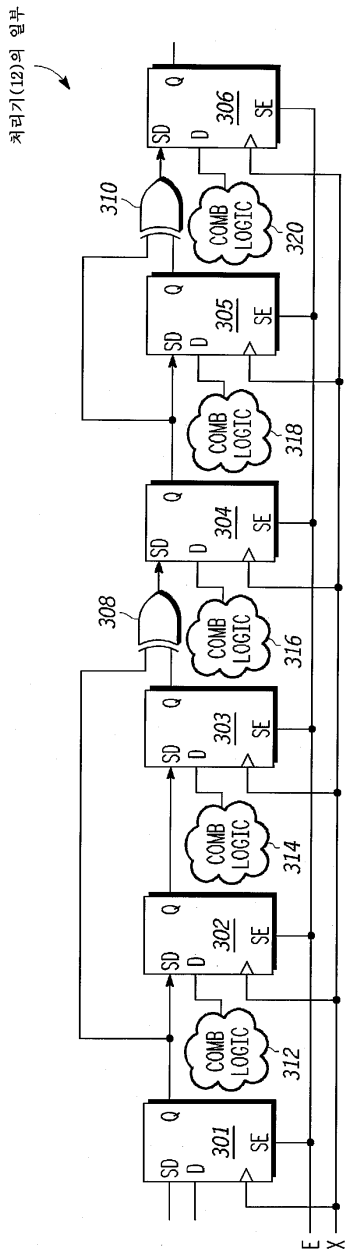
도면2


처리기 (12)의 일부
102
랜덤 패턴 생성기
100
경지 및 테스트 제어기
RESTORE VECTOR
MAXIMUM CLOCKS
MINIMUM CLOCKS
리셋 인
리셋 아웃
SIN1 108
SIN2 110
SOUT1 109
SOUT2 111
SE 138
START 160
158
리셋 106
클록 104
인터럽트 표시자 22
경지 및 테스트 표시자 24
다른 로직으로부터 134
다른 로직으로부터 135
다른 로직으로부터 136
다른 로직으로부터 148
다른 로직으로부터 149
다른 로직으로부터 150
152
154
156
128
129
130
131
132
133
144
145
146
147
결합 로직
140
142
시그니처 분석기
112
시그니처 비교기
114
예상된 시그니처 저장
116
테스트 결과 응축기
118
시그니처 120
예상된 시그니처 115
시그니처 식별자 124
비교 126
버스로/로부터 26
리셋 48

도면3


200
처리기를 포함하는 데이터 처리 시스템이 제조됨
202
데이터 처리 시스템의 공장 테스팅(에를 들면, 스캔 테스트, LBIST)을 수행
204
최종 사용자 애플리케이션에 데이터 처리 시스템을 포함
206
사용자 애플리케이션을 실행
208
정지 및 테스트 절차를 나타내는 사용자 애플리케이션을 갖는가?
210
상태 저장
212
정지 및 테스트 프시자를 제공
214
처리기, 랜덤 패턴 생성기, 시그니처 분석기, 테스트 결과 레지스터를 리셋
216
클록 X 검사 지점을 현재 검사 지점으로 선택
218
스캔 인에이블(SE)이 확인될 때, X 클록들에 대한 시그니쳐 분석기에 X 테스트 결과들을 이동하는 동안 테스트 스티뮬러스로 이동
220
SE가 무효화될 때 시스템에 클록을 제공하고 테스트 결과 레지스터에 검사 카운트 증가
221
인터럽트?
222
현재 클록 카운터 검사 지점에 도달되거나 초과되었나?
224
테스트 결과들로부터 계산된 시그니쳐를 예상된 시그니쳐와 비교하고 테스트 결과 레지스터에 합격/불합격 표시자를 조정한다
226
계산된 시그니처가 예상된 시그니처와 매칭하는가?
227
장애 표시
228
인터럽트가 수신되었나?
230
최대 클록 카운트에 도달되었거나 장애가 일어났는가?
232
다음 클록 카운트 검사 지점을 현재 검사 지점으로 선택한다
234
리셋
236
복원 벡터를 실행
238
저장된 상태를 저장
A
B
YES
NO

도면4


처리기(12)의 일부
SD
D
Q
SE
301
302
303
304
305
306
COMB LOGIC
312
314
316
318
320
308
310
E
X