(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2008년07월14일
(11) 등록번호 10-0846089
(24) 등록일자 2008년07월08일

(51) Int. Cl.
G06F 17/50 (2006.01)
(21) 출원번호 10-2001-7004099
(22) 출원일자 2001년03월30일
심사청구일자 2004년09월25일
번역문제출일자 2001년03월30일
(65) 공개번호 10-2001-0085867
(43) 공개일자 2001년09월07일
(86) 국제출원번호 PCT/US1999/022984
국제출원일자 1999년09월30일
(87) 국제공개번호 WO 2000/19343
국제공개일자 2000년04월06일
(30) 우선권주장
60/102,566 1998년09월30일 미국(US)
(56) 선행기술조사문헌
US 4,858,175 A
US 5,519,633 A
US 5,663,076 A
전체 청구항 수 : 총 18 항

(73) 특허권자
카덴스 디자인 시스템즈 인크
미국 캘리포니아 샌어제이 실리 애브뉴 2655 (우
(72) 발명자
창, 헨리
미국94086캘리포니아써니베일사우쓰메리437아파트먼트#18
쿡, 래리
미국95033
캘리포니아로스가토스스패니시랜치로드25399
(뒷면에 계속)
(74) 대리인
남상선

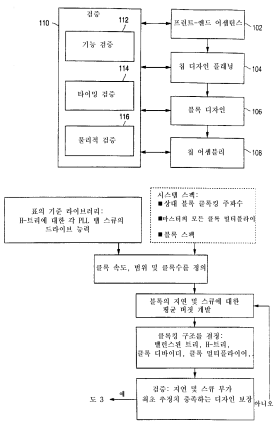
심사관 : 유병철

(54) 설계 블록들 사이에 다수의 글루 로직 엘리먼트들을 분배하는 방법 및 글루 로직 분배 효율을 증가시키는 방법

(57) 요 약

본 발명은 회로 시스템을 설계하는데 사용되는 다수의 미리 설계된 회로 블록들을 선택하고, 미리 설계된 회로 블록들을 고려하여 설계자의 경험을 반영하는 데이터를 수집하고, 상기 설계자의 경험은 처리 방법에 조절가능하며, 설계자의 경험 데이터 및 리스크의 허용 정도에 기초하여 회로 시스템의 설계를 허용하거나 거부하며, 허용에 따라 각각의 회로 블록에 대한 표준 및 수정된 제한들을 포함하는 블록 스펙들을 형성하고, 허용에 따라, 표준 및 수정된 제한들에 따라 그리고 선택된 회로블록 및 처리 방법을 변형시키지 않고서 칩상의 시스템으로서, 칩의 플로어 플랜상에 회로 블록들을 전개하는 블록 스펙들을 형성하는 것을 포함하는 회로 시스템을 설계하는 방법 및 장치에 관한 것이다.

대표도 - 도1



(72) 발명자

헌트,메릴

미국92025캘리포니아에스콘디도레일뷰드라이브2460

케이이,우디안

미국95014캘리포니아쿠퍼티노헌터스톤플레이스1086

렌나드,크리스토퍼,케이.

미국94087캘리포니아써니베일사우쓰베르나르도애브뉴745아파트먼트#319-에이

마틴,그랜트

미국94566캘리포니아플리전튼레이븐로드2424

패터슨,피터

미국92692
캘리포니아미션비에죠퍼시픽힐스드라이브25585

트렁,코안

미국95035캘리포니아밀피타스글렌뷰드라이브2206

벤카트라마니,쿠마르

미국95070
캘리포니아사라토가비아리얼드라이브19495

(81) 지정국

국내특허 : 알바니아, 아르메니아, 오스트리아, 오스트레일리아, 아제르바이잔, 보스니아 헤르체고비나, 바베이도스, 불가리아, 브라질, 벨라루스, 캐나다, 스위스, 중국, 쿠바, 체코, 독일, 덴마크, 에스토니아, 스페인, 핀란드, 영국, 그루지야, 헝가리, 이스라엘, 아이슬랜드, 일본, 케냐, 키르키즈스탐, 북한, 대한민국, 카자흐스탄, 세인트루시아, 스리랑카, 리베이라, 레소토, 리투아니아, 룩셈부르크, 라트비아, 몰도바, 마다가스카르, 마케도니아공화국, 몽고, 말라위, 멕시코, 노르웨이, 뉴질랜드, 슬로베니아, 슬로바키아, 타지키스탄, 투르크맨, 터어키, 트리니아드토바고, 우크라이나, 우간다, 우즈베키스탄, 베트남, 폴란드, 포르투칼, 루마니아, 러시아, 수단, 스웨덴, 싱가포르, 아랍에미리트, 남아프리카, 가나, 감비아, 인도네시아, 인도, 시에라리온, 세르비아 앤 몬테네그로, 짐바브웨

AP ARIPO특허 : 케냐, 레소토, 말라위, 수단, 스와질랜드, 우간다, 시에라리온, 가나, 감비아, 짐바브웨, 탄자니아

EA 유라시아특허 : 아르메니아, 아제르바이잔, 벨라루스, 키르키즈스탐, 카자흐스탄, 몰도바, 러시아, 타지키스탄, 투르크맨

EP 유럽특허 : 오스트리아, 벨기에, 스위스, 독일, 덴마크, 스페인, 프랑스, 영국, 그리스, 아일랜드, 이탈리아, 룩셈부르크, 모나코, 네덜란드, 포르투칼, 스웨덴, 핀란드, 사이프러스

OA OAPI특허 : 부르키나파소, 베닌, 중앙아프리카, 콩고, 코트디브와르, 카메룬, 가봉, 기니, 말리, 모리타니, 니제르, 세네갈, 차드, 토고, 기니 비사우

특허청구의 범위

청구항 1

디바이스 설계가 다수의 이미 존재하는 설계 블록들을 포함하는 집적 회로 디바이스 설계 방식에서의 수행을 위해, 글루(glue) 로직 분배 효율을 증가시키는 방법으로서,

선택된 글루 로직 엘리먼트를 카피하여 상기 선택된 엘리먼트와 그의 카피를 포함하는 복제 엘리먼트 세트를 형성하는 단계; 및

상기 다수의 설계 블록들에 상기 복제 엘리먼트 세트를 분배하는 단계를 포함하는 글루 로직 분배 효율 증가 방법.

청구항 2

디바이스 설계가 다수의 이미 존재하는 설계 블록들을 포함하는 집적 회로 디바이스 설계 방식에서의 수행을 위해, 상기 설계 블록들 사이에 다수의 글루 로직 엘리먼트들을 분배하는 방법으로서,

제 1글루 로직 엘리먼트가 다수의 로드들을 구동하는 출력 네트를 포함할 경우, 상기 제 1글루 로직 엘리먼트를 다수의 파생(derivative) 글루 로직 엘리먼트들로 분할하는 단계; 및

상기 다수의 설계 블록들에 상기 파생 글루 로직 엘리먼트들을 분배하는 단계를 포함하는 글루 로직 엘리먼트 분배 방법.

청구항 3

제 2항에 있어서,

상기 각각의 파생 글루 로직 엘리먼트는 하나의 단일 출력 로드만을 포함하는 것을 특징으로 하는 글루 로직 엘리먼트 분배 방법.

청구항 4

제 2항에 있어서,

상기 제 1글루 로직 엘리먼트가 다수의 입력들을 포함할 경우, 상기 분할된 엘리먼트는 제 1엘리먼트인 것을 특징으로 하는 글루 로직 엘리먼트 분배 방법.

청구항 5

제 2항에 있어서,

상기 파생 글루 로직 엘리먼트는 단지 두 개의 입력들을 포함하는 것을 특징으로 하는 글루 로직 엘리먼트 분배 방법.

청구항 6

디바이스 설계가 다수의 이미 존재하는 설계 블록들을 포함하는 집적 회로 디바이스 설계 방식에서의 수행을 위해, 상기 설계 블록들 사이에 다수의 글루 로직 엘리먼트들을 분배하는 방법으로서,

선택된 품질에 대해 상기 다수의 글루 로직 엘리먼트들을 분석하는 단계; 및

상기 분석을 기초로 한 방식으로 선택된 글루 로직 엘리먼트를 선택된 블록에 병합(merge)하는 단계를 포함하는 글루 로직 엘리먼트 분배 방법.

청구항 7

제 6항에 있어서,

상기 선택된 블록은 상기 선택된 글루 로직 엘리먼트에 대한 기능적 관계(functional affinity)를 기초로 한 방식으로 선택되는 것을 특징으로 하는 글루 로직 엘리먼트 분배 방법.

청구항 8

제 7항에 있어서,

상기 기능적 관계는 상기 병합이 상기 회로 디바이스 설계의 적절한 기능을 위해 요구되는 물리적 I/O 엘리먼트들의 수를 감소하는 지의 여부를 포함하는 것을 특징으로 하는 글루 로직 엘리먼트 분배 방법.

청구항 9

제 6항에 있어서,

둘 또는 그 이상의 설계 블록들이 상기 병합을 위한 동등한 후보들이라면, 가장 낮은 핀 밀도를 가진 블록이 선택되는 것을 특징으로 하는 글루 로직 엘리먼트 분배 방법.

청구항 10

제 7항에 있어서,

상기 기능적 관계는 상기 선택된 글루 로직 엘리먼트 및 상기 선택된 블록이 모두 향상된 칩 레벨 타이밍 특성들을 가지는지의 여부를 포함하는 것을 특징으로 하는 글루 로직 엘리먼트 분배 방법.

청구항 11

디바이스 설계가 다수의 이미 존재하는 설계 블록들을 포함하는 집적 회로 디바이스 설계 방식에서의 수행을 위해 글루 로직을 분배하는 방법으로서,

상기 설계 블록들 사이에서 카피되고 분배될 수 없거나 상기 설계 블록들과 병합될 수 없는 다수의 엘리먼트들을 식별하는 단계; 및

상기 식별된 다수의 엘리먼트들을 클러스터링(clustering)하는 단계를 포함하는 글루 로직 분배 방법.

청구항 12

제 11항에 있어서, 상기 각각의 클러스터링된 엘리먼트들은 입력 네트들에 대하여 다중 로드들 및 출력 네트들에 대하여 다중 로드들을 포함하는 것을 특징으로 하는 글루 로직 분배 방법.

청구항 13

제 11항에 있어서, 상기 다수의 엘리먼트들은 유사한 기능을 가진 입력들을 포함하는 것을 특징으로 하는 글루 로직 분배 방법.

청구항 14

디바이스 설계가 다수의 이미 존재하는 설계 블록들을 포함하는 집적 회로 디바이스 설계 방식에서의 수행을 위해, 상기 설계 블록들 사이에 글루 로직을 분배하는 방법으로서,

제 1글루 로직 엘리먼트의 제 1특징을 식별하는 단계;

상기 제 1글루 로직 엘리먼트와 호환가능하게 하는 제 2특징을 가지는 제 2글루 로직 엘리먼트를 식별하는 단계; 및

상기 식별된 제 2글루 로직 엘리먼트와 상기 제 1글루 로직 엘리먼트를 병합하는 단계를 포함하는 글루 로직 분배 방법.

청구항 15

제 14항에 있어서,

상기 제 1특징은 상기 제 1글루 로직 엘리먼트에 의하여 요구되는 핀들의 수를 포함하는 것을 특징으로 하는 글루 로직 분배 방법.

청구항 16

제 14항에 있어서,

상기 제 1특징은 상기 제 1글루 로직 엘리먼트의 입력 구조를 포함하는 것을 특징으로 하는 글루 로직 분배 방법.

청구항 17

제 14항에 있어서,

상기 제 1특징은 상기 제 1글루 로직 엘리먼트의 출력 구조를 포함하는 것을 특징으로 하는 글루 로직 분배 방법.

청구항 18

제 14항에 있어서,

상기 제 2글루 로직 엘리먼트는 설계 블록인 것을 특징으로 하는 글루 로직 분배 방법.

청구항 19

삭제

청구항 20

삭제

청구항 21

삭제

청구항 22

삭제

청구항 23

삭제

청구항 24

삭제

청구항 25

삭제

청구항 26

삭제

청구항 27

삭제

청구항 28

삭제

청구항 29

삭제

청구항 30

삭제

청구항 31

삭제

청구항 32

삭제

청구항 33

삭제

청구항 34

삭제

청구항 35

삭제

청구항 36

삭제

청구항 37

삭제

청구항 38

삭제

청구항 39

삭제

청구항 40

삭제

청구항 41

삭제

청구항 42

삭제

청구항 43

삭제

청구항 44

삭제

청구항 45

삭제

청구항 46

삭제

청구항 47

삭제

청구항 48

삭제

청구항 49

삭제

청구항 50

삭제

청구항 51

삭제

청구항 52

삭제

청구항 53

삭제

청구항 54

삭제

청구항 55

삭제

청구항 56

삭제

청구항 57

삭제

청구항 58

삭제

청구항 59

삭제

청구항 60

삭제

청구항 61

삭제

청구항 62

삭제

청구항 63

삭제

청구항 64

삭제

청구항 65

삭제

청구항 66

삭제

청구항 67

삭제

청구항 68

삭제

청구항 69

삭제

청구항 70

삭제

청구항 71

삭제

청구항 72

삭제

청구항 73

삭제

청구항 74

삭제

청구항 75

삭제

청구항 76

삭제

청구항 77

삭제

청구항 78

삭제

청구항 79

삭제

청구항 80

삭제

청구항 81

삭제

청구항 82

삭제

청구항 83

삭제

청구항 84

삭제

청구항 85

삭제

청구항 86

삭제

청구항 87

삭제

청구항 88

삭제

청구항 89

삭제

청구항 90

삭제

청구항 91

삭제

청구항 92

삭제

청구항 93

삭제

청구항 94

삭제

청구항 95

삭제

청구항 96

삭제

청구항 97

삭제

청구항 98

삭제

청구항 99

삭제

청구항 100

삭제

청구항 101

삭제

청구항 102

삭제

청구항 103

삭제

청구항 104

삭제

청구항 105

삭제

청구항 106

삭제

청구항 107

삭제

청구항 108

삭제

청구항 109

삭제

청구항 110

삭제

청구항 111

삭제

청구항 112

삭제

청구항 113

삭제

청구항 114

삭제

청구항 115

삭제

청구항 116

삭제

청구항 117

삭제

청구항 118

삭제

청구항 119

삭제

청구항 120

삭제

청구항 121

삭제

청구항 122

삭제

명 세 서

기 술 분 야

<1> 본 발명은 일반적으로 집적 회로("IC") 디바이스 설계에 관한 것이며, 특히 미리 설계된 회로 블록들을 재사용하는 시스템의 설계에 관한 것이다.

<2> 본 출원은 여기서 참조되는 1998년 9월 30일에 출원된 "블록 기반 설계 방법"이란 명칭의 미국 특허 출원 No. 60/102,566을 우선권으로 주장한다.

배 경 기 술

<3> 최근에, 실리콘 처리 기술의 지속적인 혁신에 의해 집적 회로 디바이스의 비용이 크게 절감되고 성능 및 기능성은 크게 증가되었으며, 이로 인해 전자 제조 및 정보 처리 산업의 발전이 고무되었다. 차례로, 이러한 고속의

성장 산업들은 집적 회로 설계 시스템 개발자들에게 여전히 더 고속이며 비용이 싼 디바이스들을 제조할 것을 요구하고 있다. 그 결과, 설계 산업은 다음과 같은 급격한 변화를 겪고 있다.

<4> (1) 칩 설계는 더 크고 더 복잡해지고 있다. 예를 들어, 1997년에, 전형적인 집적 회로는 100-500K 게이트들을 포함하였다. 1998년에는, 전형적인 디바이스는 100만에서 200만 게이트들을 포함하였다. 1999년의 기술은 상기의 경향에 따라 400만에서 600만 게이트들이 설치되는 디바이스를 선보였다.

<5> (2) 칩 설계는 더욱 애플리케이션에 특정되고 있다. IC 설계의 초기에는, 디바이스 제조업자들은 최종 사용자들이 전자 제품에 설계했던 여러 "오프-더-쉘프 (off-the-shelf)" 칩들을 생산하였다. 현재에는, 전자 제품 제조업자들은 특정 기능을 수행하기 위해 더욱 자주 주문형 칩 설계를 주문한다.

<6> (3) 전자 제품 개발은 현재 주로 고객 주문에 의해 이루어지며, 상기 개발은 제품 수명 사이클을 짧게 하며, 따라서 허용된 설계 시간 및 자원들이 감소된다. 예를 들어, 1997년에, 평균 설계 사이클은 12-18 개월 사이였다. 1998년에는, 상기 평균 시간은 10-12 개월로 감소했으며 1999년에는 상기 산업은 8-10 개월 설계 사이클 시간을 요청하고 있다.

<7> (4) 설계 시간 제한은 대응하는 설계 노력을 요구한다. 이전에는, 업스트림 시스템 소자들에 대한 중요한 설계 결정들은 다운스트림 시스템 소자 설계들이 검증될 때까지 대기할 수 있었다. 설계 관리자들은 더 이상 순차적으로 설계 작업을 수행하는 여유를 가질 수 없다. 여러 시스템 소자들은 동시에 개발되어야 할 수도 있다. 따라서, 설계 관리자들은 적어도 몇몇의 시스템 소자 설계들이 완성되기 전에 중요한 예측을 수행하도록 요청받는다.

<8> 상기 요구들에 대응하여, 전자 시스템 설계는 시스템이 다수의 현존하는 소자 설계 블록들(기술 분야에서 "지능 특성 블록들" 또는 "IP 블록들"과 같이 지칭됨)을 집적함으로써 설계되는 블록 기반 설계("BBD")로서 기술분야에 공지된 방법으로 진행하고 있다. 상기 미리 설계된 블록들은 내부 설계 팀으로부터 얻어지거나 또는 다른 설계 회사들로부터 허가받을 수 있으며, 기본적으로 다른 설계 구조 및 환경들에 의해 지원될 수 있다. 더욱이, 사전 설계된 블록들은 다른 설계 요구 및 제한들에 맞추어 개발될 수 있다.

<9> BBD를 사용하여 설계자들에 의해 직면한 또 다른 도전 과제는 프런트-엔드(프로젝트 허용) 지연들 및 시스템 설계 실행가능성을 결정하는데의 불확실성에 의해 발생되는 리스크이다. 현재의 ASIC(application-specific integrated circuit) 설계들은 주로 RTL(register transfer level) 단계에 머물러 있으며, 심지어 더욱 초기에는, 특정 레벨에서 고객들에 의해 설계자에 제공되었다. 상기 설계들은 그후에 효율적인 경비로 설치를 제공하도록 요청된 영역, 성능 및 전력 교환에 따라 이용가능한 합성 기술의 제한에 기초한 방법으로 구분된다. 상기 방법에서, 설계자는 입력으로서 시스템 스펙을 허용하고 궁극적으로 물리적 설치(설계 장소, 라우팅 및 검증을 포함)를 위해 네트리스트 레벨 설계를 제공한다. 설계 스펙이 의도된 처리 기술 또는 클로킹, 전력 및 크기 스펙을 포함하는 이용할 수 있는 처리 기술의 능력 내에 있다면, 이용가능한 설계 방법은 논리적으로 예측할 수 있고 이용가능한 회로 설계 툴로 우수하게 동작한다.

<10> 그러나, RTL 레벨 설계 및 시스템 레벨 설계 동작들은 일반적으로 결합될 수 없거나 느슨하게 결합되는데, 이것은 시스템 레벨 기능 정의로부터 ASIC(RTL) 레벨로의 결부된 링크가 없다는 것을 의미한다. RTL 레벨 설계는 해설(paper) ASIC 스펙에 기초하여 개발되며 ASIC 인터페이스 주위에 형성된 새로 형성된 테스트 도구에 의해 검증된다. 따라서 ASIC 설계에 대해 이용가능한 설계 및 설치 방법은 효율적인 블록 집적을 방해하는 다수의 문제점들을 안고 있다.

<11> 첫째로, 현재의 방법은 타이트한 시간대 시장(time-to-market) 제한내에서 계층적 검증 및 짧은 어셈블리 시간을 제공하면서, 서로 다른 설계 고려사항을 갖는 다수의 자원들에 의해 제공된 다수의 설계 블록들을 집적하는 능력을 포괄적으로 평가하고 이를 보장하는 하향식 방법을 제공하지 않는다.

<12> 또한, 기존의 ASIC 설계 방법은 확장성을 제공하지 않는다. 상당한 수의 기존의 방법들은 평면 설계에 초점이 맞추어진다. 상기 방법은 100만개 이상의 게이트들을 갖는 시스템에 대해 상위 레벨 설계를 어셈블링하도록 요청된 지속시간에 있어서 상당한 문제점들을 초래하였다.

<13> 게다가, 현재의 ASIC 설계 방법들은 사전 설계된 회로 블록들의 재사용에 적합하지 않다. 이용가능한 설계안들은 특정 디바이스 구조내의 집적 회로 설계 블록들과 관련된 타이밍, 클록, 버스, 전력, 블록 배열, 검증 및 테스트 문제들을 해결하기 위한 길잡이를 제공하지 않는다. 따라서, 재사용을 차단하기 위한 포괄적인 접근이 없이는, 현재의 방법들은 특별하고 예측불가한 설계 방법을 발생시키고 설계 구현 실행가능성을 감소시키며, 전달하는 비용 및 시간을 증가시키며, 종종 설계된 시스템에 맞추기 위해 성능이 떨어지는 변형물들을 사전 설계된

회로 블록들에 트리거시킨다. 더욱이, 현재의 방법들은 성능 교환 분석 및 클록 주파수와 같은 중요한 설계 파라미터들의 피드백 및 성공적인 그리고 예측가능한 완성 칩 설계와 설치의 영역 대 위험 요소를 제공하지 못한다.

<14> 따라서, 발전하는 환경을 만족시키고 이용가능한 기술의 결점을 제시할 수 있는 방법이 요청된다.

<15> 또한 회로 설계의 다수 자원들로부터 미리 설계된 회로 블록들을 사용 및 재사용하는 적합한 방법이 요청된다.

<16> IP 블록들의 조합은 또한 블록들이 단일 디바이스상에서 함께 동작하도록 하는 "글루(glue)" 로직에 대한 필요성을 발생시킨다. 글루 로직은 주로 설계 블록들을 상호접속하는 역할을 하며 통상적으로 설계를 통해 분산배치된 블록들사이에 상주한다. 글루 로직 소자들은 칩 설계의 여러 단계들 동안 설계에 부가될 수 있거나, 또는 호스트 블록에 대한 상호접속 메카니즘으로서 동작하는 설계내의 각 블록의 가장 먼 경계에 상주할 수 있다. 자원에 관계없이, 글루 로직은 원래의 블록 설계자에 의해 고려되지 않았을 지연들을 도입하여 블록들사이의 글루 로직의 배치로부터 발생하는 와이어 혼잡성 및 타이밍 복잡성을 최소화하는 설계내에서 최적으로 배치되어야 한다.

<17> 따라서, 기술분야에 있어서 본 발명에 속하는 블록 기반 설계의 글루 로직을 배치하고 분배하는 개선된 방법이 요청된다.

<18> 또한 여러 글루 로직 소자들의 기능적 밀접성을 설명하고 신규한 설계 블록들로 그룹화하는 글루 로직 분배 메카니즘이 요청된다.

<19> 또한 기술분야에 최적화된 양의 글루 로직을 이미 존재하는 설계에 접목하는 글루 로직 분배 메카니즘이 요청된다.

<20> 게다가, 이미 존재하는 ASIC 설계 방법은 사전 설계된 회로 블록들의 재사용에 적합하지 않다. 이용가능한 설계들은 특정 디바이스 구조내의 집적 회로 설계 블록들과 관련된 타이밍, 클록, 버스, 전력, 블록 배열, 검증 및 테스트 문제를 해결하기 위한 길잡이를 제공하지 않는다. 회로 블록들은 다수의 비-조화(inconsistent) 자원들로부터 생기기 때문에, 과제는 상기 회로 블록들을 블록 기반 설계에 적합한 형태로 회로 시스템으로 어떻게 집적하느냐이다.

<21> 따라서, 블록 기반 설계에 적합한 형태의 다수의 비-조화 자원들로부터의 회로 블록들을 상호접속하는데 적합한 방법 및 장치가 요청된다.

<22> 또한 여러 인터페이스들을 갖는 회로 블록들을 표준화된 인터페이스들을 갖는 회로 블록들로 변환하기 위한 인터페이스들을 제공하는 방법 및 장치가 요청된다.

<23> 물론, 단일 칩상의 전체 시스템을 포함하는 모든 IC들은 상기 칩이 성능 요건들을 만족하고 숨겨진 제조 결함이 없는지를 검증하기 위한 일련의 테스트들을 거쳐야 한다. 만일 제조 결함을 놓치게 되면, 결함있는 칩은 어셈블리 처리후에, 더 나쁘게는 현장에서까지 발견되지 않을 수 있다. 고객 만족의 효과란 의미에서 상기 "테스트 면제"의 비용은 생산 라인의 질을 떨어뜨릴 수 있다.

<24> 일반적으로, 결함들을 검출하는 세가지 형태의 테스트들이 있다. DC 파라메트릭 테스트, AC 파라메트릭 테스트 및 기능("PLL") 테스트가 그것이다. DC 파라메트릭 테스트에 있어서, 상기 칩의 입력, 출력, 입력 대 출력 송신, 총 전류 및 전력 소비가 측정된다. AC 파라메트릭 테스트에 있어서, 입력 및 출력 신호들의 상승 및 하강 시간들, 입력 및 출력 단자간의 전파 지연 시간, 최소 클록 펄스폭 및 칩의 동작 주파수가 측정된다. 기능적 테스트에 있어서, 상기 칩은 규정된 동작 조건하에서 설계된대로 기능하는지를 알아보기 위해 테스트된다. 일반적으로, 테스트 패턴을 입력 단자("테스트 벡터")에 적용하고 출력 단자에서 검출된 출력 패턴을 예측된 패턴과 비교하여 기능적 테스트를 수행한다.

<25> 테스트를 위한 설계("DFT") 방법이 제시되기 전에, 설계자들은 칩을 형성하고 어셈블링한 후에, 완성된 설계를 테스트 설계자들에게 넘겼다. 상기 테스트 설계자들은 그후에 패키지 레벨 테스트 로직을 부가하고 상기 칩을 제조업자("fab")에게 보냈다. 상기 제조 테스터들은 그후에 칩을 검사하고 패키지 레벨 로직상에 상기에 기술된 테스트들을 포함하는 보드 테스트 프로토콜을 실행했다. 이용가능한 스캔 설계 방법은 "단일" 테스트 방법을 예측가능하고 지속가능한 테스트 결과를 갖는 전체 칩에 적용하는 고도로 효율적이며 광범위하게 사용되는 방법의 단순 예이다. 다른 특별한 방법들은 스캔할 수 없는 설계 형태를 다루는데 사용될 수 있다.

<26> 오늘날, 이전에 전체 칩에 포함된 로직은 이제 더 큰 칩에 포함되는 단일 가상 소자(VC) 또는 설계 블록으로써

사용된다. 따라서, 테스트들은 더이상 회로 설계가 완성된 후에 예정되지 않을 수 있다. 설계자들은 설계 프로세스를 통해 전체 패키지된 칩 뿐 아니라 각각의 설계 블록을 어떻게 테스트할지를 계획하여야 한다. 설계 프로세스는 따라서 적합한 하나 이상의 테스트 방법들을 적용함으로써 테스트 가능성을 확인하여야 한다.

<27> DFT의 장점들은 공지되어 있다. DFT 로직 및 테스트 벡터 검증 기능은 생산 사이클에서 더 짧은 생산 준비 테스트들을 허용한다. 또한 DFT 스캔 경로들은 칩 및 그렇지 않으면 이용할 수 없는 시스템 상태들로의 액세스를 제공한다. 우수한 DFT 설계는 그로인해 시장 비교 시간을 줄이고 전치 설계 프로세스 및 제조 테스트들의 개발을 용이하게 함으로써 테스트 비용을 감소시킨다.

<28> 따라서, 이용가능한 기술에 의해 나타난 네가지 요청안이 있다.

<29> 첫째로, BBD에 대한 신규한 DFT는 제한된 테스트 액세스를 공유하기 위해, 그리고 전체 SOC 레벨 테스트 목적에 부합하도록 다른 비-유사 테스트 방법들간의 미리 설계된 테스트 데이터를 효율적으로 이용할 수 있어야 한다.

<30> 둘째로, 기술 표준화에 기인한 새로운 결함 형태들 및 새로운 결함 레벨들, 혼합 신호 및 혼합 기술 설계의 새로운 복잡도 및 증가하는 I/O 카운트 및 새로운 패키징 기술들의 발생 문제들에 직면하여야 한다.

<31> 세째로, 본질적으로 통합된 구조적 테스트 모델이 부족한 IP 블록들을 집적하는 어려움에 직면하여야 한다. SOC 레벨 테스트 액세스 및 결합 격리가 요청되며, 현재 이용가능한 DFT 방법에 의해 크게 지원될 수 없는 저전력 설계 기술(즉, 래치 기반, 게이트 클록, 유도 클록, 파이프라인 및 저임계 전압)에 대한 요구가 제시되어야 한다.

<32> 그리고 새로운 DFT 방법은 제한된 또는 부적절한 테스트 정보에 직면할 때도 고유하고 지속적인 테스트 집적 모델을 갖는 시장 압력에 대한 시간을 극복해야 한다.

<33> 이용가능한 기술은 테스트 데이터가 오류들의 세트에 대해 부분적으로 또는 완전히 생성되고 검증될 수 있도록 구조적 정보(즉, 오류 모델들 및 테스트 모델들)를 요구한다. 예를 들어, 스캔 설계 방법은 동기적 설계에만 적용할 수 있으며 단일 스턱-앳-폴트(stuck-at-fault) 모델들만을 검출한다. 게다가, 다른 DFT 해답안은 스캔을 기반으로 하며, 따라서 구조적 정보를 포함하지 않은 어려운 IP 테스트 모델을 공유하고 검증하는 것을 다소 어렵게 한다.

<34> 이용가능한 기술은 또한 공유 및 검증이 가능하더라도(즉, 소프트 IP 모델) 전류 게이트 카운트의 급증을 유지할 수 없는 비선형 계산 모델을 요구한다. 그러나, 소프트 IP들은 반드시 스캔할 수 있거나 합성할 수 있지 않아서, 때때로 예측할 수 없고 다룰 수 없는 테스트 개발을 발생시킨다.

<35> 마지막으로 설계 검증으로 돌아가면, SOC 설계의 다수의 미리 설계된 블록들의 사용에 의해 나타난 문제는 신뢰성 있고 효율적인 기능적 검증 방법의 필요성이다. 유용한 기술에서, 테스트 도구(suite)들은 다중블록 설계를 검증하는데 사용된다. 각 테스트 도구는 블록들이 집적되기 전에 블록들의 각각을 테스트하는데 사용된다. 그 후에, 블록들을 집적한 후에, 시스템 레벨에서 기능적 검증이 가능하도록 테스트 도구를 조절하기 위해 상당한 노력이 요구된다. 테스팅 및 디버깅 프로세스는 최종의 완전한 시스템 검증이 확실하게 제공될 수 있기 전의 다수번 반복되어야 할 필요가 있다.

<36> 상기 문제에 대한 하나의 유용한 방법은 상응하는 행동 모델들에 대한 형성 모듈들의 치환이며, 그로인해 혼합 모드 상황에서 칩 레벨 시뮬레이션 및 테스팅을 허용한다. 상기 방법은 효율적으로 수행된다면 바람직한 결과들을 제공할 수 있고 상기에 기술된 반복적인 블록 기반 시뮬레이션들 보다 비용도 저렴할 수 있지만, 상기 방법은 전체 칩이 신뢰성있는 기능적 검증을 획득하도록 시뮬레이팅되어야 하기 때문에, 여전히 비용이 비싸고 느리다.

<37> 특히 심각한 문제는 버스 구조들을 기능적으로 검증할 필요성에 의해 다중 블록 설계들에 존재한다. 유용한 기술에서, 버스 검증은 두가지 방식 중 하나로 달성된다. 상기 버스는 전체 칩의 완전한 부분으로서 디버깅되고 검증될 수 있거나, 또는 새로 생성된 블록들에 의해 제공된 세부적인 구현을 고려하여 미리 정의된 블록들에 대한 버스 기능 모델을 사용하여 검증될 수 있다. 그러나, 완전한 버스 검증은 느리고 비용이 비쌀 수 있다. 전체 칩은 버스 설계를 검증하기 위해 사용되어야 하고 완전한 버스 검증은 디버깅이 버스관련 버그들이 없다는 것을 확인하기 위한 세부 레벨 및 잠재성 때문에 어렵고 시간을 낭비하게 될 때, 설계 사이클에서 나중에 실행될 수 있을 뿐이다. 버스 기능 모델 방법은 소정의 상기와 같은 문제점들을 용이하게 하지만, 새롭게 생성된 블록들에 대한 구체적인 구현을 요구한다. 게다가, 버스 기능 모델들은 자체적으로 오류가 발생하기 쉽고 신호 흔적을 생성하고 디버그를 어렵게 하거나 불가능하게 하는 "블랙 박스(black boxes)"로서 이용할 수 있다.

발명의 상세한 설명

<38> 이용가능한 기술의 단점을 해결하기 위해, 본 발명은 회로 시스템을 설계하는 방법 및 장치를 제공하는데, 상기 방법은,

<39> (a) 상기 회로 시스템을 설계하는데 이용될 다수의 회로 블록을 선택하는 단계;

<40> (b) 상기 미리 설계된 회로 블록에 대하여, 설계자의 경험을 반영하는 데이터를 수집하는 단계를 포함하며, 상기 설계자의 경험은 처리 방법에 적용가능하고;

<41> (c) 상기 설계자의 경험 데이터 및 허용가능한 정도의 리스크를 기초로 한 방식으로 회로 시스템의 설계를 허용 또는 거부하는 단계;

<42> (d) 허용할 때, 각각의 회로 블록(FEA)에 대한 표준 및 변형된 제한을 포함하는 블록 스펙을 형성하는 단계; 및

<43> (e) 허용할 때, 선택된 회로 블록 및 처리 방법을 변경하지 않고 상기 표준 및 변형된 제한에 따라 칩의 플로어 플랜에 회로 블록을 배치하는 블록 스펙을 형성하는 단계를 포함한다.

실 시 예

<102> 이용가능 기술의 단점을 극복하기 위해, 본 발명은 블록 기반 설계("BBD")에 대한 신규한 방법 및 실행을 개시한다.

<103> 도 1을 참조하면, 본 발명에 따른 블록 기반 설계(BBD) 방법에 기초한 설계 프로세스를 도시하는 흐름도(100)가 도시된다. 도 1에 도시된 바와 같이, 상기 설계 프로세스는 프런트 엔드 허용 설계 단계(102), 칩 플래닝 설계 단계(104), 블록 설계 단계(106), 칩 어셈블리 설계 단계(108) 및 검증 설계 단계(110)를 포함한다.

<104> 프런트 엔드 허용 설계 단계(102)는 시스템 통합자(칩 설계자)가 차세대 설계 프로젝트의 실행가능성을 평가하도록 해준다. 프런트 엔드 허용 설계 단계(102)에서, 설계자는 ASIC를 설계하기 위해 기능적 및 다른 요건들(전달 시간 및 경비와 같은)을 포함하는 고객로부터의 스펙을 수신한다. 상기 고객은 또한 소정의 미리 설계된 회로 블록들 및 상기 회로 블록들에 대한 테스트 벤치들을 제공할 수 있다. 블록들을 공급받은 고객과 함께, 프런트 엔드 허용 설계 단계(102)를 이용하는 설계자는 입력으로서 일부는 제 3 자에 의해 공급될 수 있고, 일부는 레가시(legacy) 회로 블록들일 수 있으며, 일부는 새로 생성될 수 있는 개별 소스들로부터의 회로 블록들을 허용할 수 있다. 상기 선택된 회로 블록들은 소프트, 펌(firm) 또는 하드 설계 상태에 있을 수 있다.(소프트 상태는 RTL 레벨이며; 하드는 GDSII 레벨이며; 및 펌은 게이트 레벨 또는 네트리스트와 같은 소프트와 하드의 중간이다.) 프런트 엔트 허용 설계 단계(102)는 그후에 사용 데이터, 실제 시뮬레이션을 통한 평가 데이터 및/또는 부분 실행 데이터의 필드를 포함하여, 설계자의 유용한 경험들을 수집한다. 프런트 엔드 허용 설계 단계(102)는 그후에 설계자가 고객의 요구조건들, 설계자의 유용한 경험 및 설계자의 허용가능한 정도의 리스크를 포함하는 설계 특성 파라미터들상에 기초한 설계 프로젝트를 허용할 것인지 결정하는 것을 돕기 위한 평가를 제공한다. 게다가, 기능적 스펙에 기초하여, 프런트 엔트 허용 설계 단계(102)는 상기 회로 설계에 사용될 미리 설계된 회로 블록들의 최종 세트를 기술한다.

<105> 프런트 엔드 허용 설계 단계(102)는 세가지 평가 단계들을 제공한다: 대강의 평가, 중간정도의 평가 및 세밀한 평가가 그것이다. 일 단계의 평가가 만족스럽지 않으면, 프런트 엔드 허용 설계 단계(102)는 설계 특성 파라미터들을 세분화하고 다음 단계의 부가 평가를 실행한다.

<106> 제안된 설계 프로젝트가 허용할 수 있는 것으로 나타나면, 프런트 엔드 허용 설계 단계(102)는 이전에 일찍 검출되는 설계의 상기 문제점들을 보증하기 위해 그리고 프로젝트 요건들, 설계자의 유용한 경험 및 선택된 처리 방법에 의해 한정된 경계내의 포괄적 방법으로 상기 문제점들이 해결될 수 있도록 보증하기 위해 포괄적 단계들을 제공한다. 프런트 엔드 허용 설계 단계(102)는 선택된 미리 설계된 회로 블록들, 설계 표준 및 상호의존 설계 제한들을 포함하는 처리 방법을 정의하는 설계 스펙을 생성한다.

<107> 칩 플래닝 설계 단계(104)는 프런트 엔트 허용 설계 단계(102)의 출력으로부터 설계 스펙을 선택된 회로 블록들의 각각에 대한 블록 스펙들로 변환한다. 칩 플래닝 설계 단계(104)에서 수행되는 태스크들은 (1) 칩 설계, 어셈블리 및 지연의 예측가능성, 라우팅 가능성, 영역, 파워 낭비 및 타이밍에 초점을 둔 실행을 위한 플랜들을 개발하는 단계; 및 (2) 제한들을 식별하고 조절하는 단계를 포함한다. 구체적으로, 설계 표준 및 프런트 엔드 허용 설계 단계(102)의 출력으로 제공된 상호의존 제한들에 기초하여, 칩 플래닝 설계 단계(104)는 프런트 엔드

허용에서 규정된 경계조건(요건들 및 제한들과 같은)내에서 칩 플래닝을 제공한다. 독창적인 칩 플래닝 설계 단계(104)는 한번에 하나의 제한을 고려하고 프런트 엔드 허용 설계 단계(102)에 의해 명기된 바와 같은 전체 설계 표준과 맞춘다. 칩 플래닝 설계 단계(104)는 프런트 엔드 허용 설계 단계(102)에 선택된 회로 블록들의 각각에 대한 예산을 형성하고, 회로 블록의 스펙을 교정하고 프런트 엔드 허용 설계 단계(102)에 의해 명기된 처리 방법내에서 제한들을 조절함으로써 이것을 달성한다. 상기 본 발명의 칩 플래닝 설계 단계에 반해, 기존의 방법들은 새로운 기능적 블록들을 생성하거나 또는 설계 시간을 증가시키고 프로젝트 리스크를 높이면서 설계 표준을 맞추기 위해 프로세싱 기술을 변경한다. 칩 플래닝 설계 단계(104)는 또한 하기에 더 상세히 검토되는 글루 로직(즉, 상기 선택된 회로 블록들을 상호접속하도록 요구되는 하드웨어)에 대한 스펙을 생성한다. 칩 플래닝 설계 단계(104)는 칩상에 하나 이상의 영역들을 차지하는 새로운 글루 로직 블록들, 상기 선택된 회로 블록들로 분배된 분배 글루 로직 및 상위 레벨 블록 글루 로직 소자들을 포함하는 세가지 형태의 글루 로직들을 출력으로 제공한다.

<108> 선택된 회로 블록을 이음새 없이 연결하기 위해, 필요한 경우, 블록 설계 단계(106)는 표준 인터페이스를 형성하기 위해 각 회로 블록 주위에 (칼라로 불리는)인터페이스를 내장한다. 회로 블록이 연성, 경성 또는 강성일 수 있으므로 각 칼라는 마찬가지로 연성, 경성 또는 강성일 수 있다. 블록 설계 단계(106) 출력은 이하를 제공한다: (1)칩 내의 모든 회로 블록은 제한이나 비용면에 부응하여, 전술한 칩 설계 플랜 및 구성에 적합하다; (2)칩 어셈블리 설계 단계(108)에 모든 필요한 모델 및 모든 회로의 계획이 제공된다; (3)설계는 개발하는 방법에 대해 가능하며 칩 플랜 설계 단계(104), 통상적인 회로 블록의 적용 및 제 3의 회로 블록에서 발생되는 새로운 회로 블록을 만들어 내기 위해 적용된다; 그리고 (4)설계는 소정의 칩 구조 및 비용면에 적합하다.

<109> 칩 어셈블리 설계 단계(108)는 설계 단계 제작을 위해 최고 레벨 설계를 테이핑 아웃하기 위해 회로 블록을 집적한다. 칩 어셈블리 설계 단계(108)는 소정의 전체 설계 상세부의 완성은 물론 하드 블록 및 칩 버스 라우팅의 최종 배치를 포함한다. 칩 어셈블리 설계 단계(108)는 모든 회로 블록이 칩 플랜 내로 설계, 수정 및 집적되기까지 시작하지 않는다. 칩 어셈블리 설계 단계(108)를 위한 입력은 프런트 엔드 허용 설계 단계(102) 또는 칩 플래닝 설계 단계(104)로부터 수신되는 파워, 영역 및 시간 이득 스펙을 포함한다.

<110> 검증 설계 단계(110)는 각 단계에서 설계가 기능적 스펙에서 설명된 대로 고객 기능 요구 및 프런트 엔드 허용 설계 단계(102)에서 공급된 칩 테스트 벤치를 충족시키는 것을 보장한다. 검증 설계 단계(110)는 기능적 검증(112), 타이밍 검증(114) 및 물리적 검증(116)을 포함한다.

<111> 기능적 검증 단계(112)는 각 설계 단계에서 선택된 회로 블록에 대한 논리적 기능 및 칩 테스트 벤치가 고객 스펙의 기능적 요구를 충족시키는 것을 보장한다. 기능적 검증은 프런트 엔드 허용 설계 단계(102), 칩 플래닝 설계 단계(104), 블록 설계 단계(106) 또는 칩 어셈블리 설계 단계(108) 동안 실행될 수 있다. 타이밍 검증은 설계의 각 단계에서 신호 타이밍이 논리적 기능을 발생시키며 고객의 스펙에서 설명된 테스트를 통과하는데 적절한 것을 보장한다. 타이밍 검증은 프런트 엔드 허용 설계 단계(102), 칩 플래닝 설계 단계(104), 블록 설계 단계(106) 또는 칩 어셈블리 설계 단계(108) 동안 실행될 수 있다. 물리적 검증은 회로 설계를 위해 물리적 레이아웃이 고객 스펙을 만족시키는 것을 보장한다.

<112> 설계 프로세스 동안, 프런트 엔드 허용 설계 단계(102), 칩 플래닝 설계 단계(104), 블록 설계 단계(106) 및 칩 어셈블리 설계 단계(108)는 소정의 기능을 수행하며, 검증 기능(110)을 포함하는 기능적 검증(112), 타이밍 검증(114) 및 물리적 검증(116)을 위해 필요한 정보를 생성한다. 만일 설계 프로세스의 특정 단계에서 검증동안 소정의 에러가 발생할 경우, 이러한 에러는 바람직하게 다음 단계로 넘어가기 전에 교정된다.

<113> 따라서, 칩 어셈블리 설계 단계(108)에서, 설계 프로세스는 칩을 생산하기 위한 최고 레벨 설계를 생성하며, 칩에 대한 설계 및 모든 칩 테스트 벤치에 사용되는 각각의 회로 블록에 대한 칩 테스트 벤치의 검증을 포함한다.

<114> 도2-15는 요약 형식으로 설명될 것이다. 이러한 도면은 이하에서 보다 상세하게 논의될 재료의 자세한 설명을 제공한다.

<115> Ⅱ. 프런트 엔드 허용(102)

<116> 도 2를 참조하면, 흐름도(200)는 본 발명에 따라, 프런트 엔드 허용 설계 단계(102)의 스텝(210-216)을 도시한다.

<117> Ⅲ. 칩 플래닝(104)

<118> 칩 플래닝 설계 단계(104)는 이하의 모듈을 포함한다:

<119> (1) 클록 플래닝;

<120> (2) 버스 식별 및 플래닝

<121> (3) 파워 플래닝;

<122> (4) I/O 및 아날로그/믹싱된 신호 요구;

<123> (5) 테스트 플래닝;

<124> (6) 타이밍 및 플로워 플래닝;

<125> (7) 버스 검증;

<126> 도 3에는 본 발명에 따른 클록-플래닝 모듈이 도시된다.

<127> 도 4에는 본 발명에 따른 버스 식별 및 플래닝 모듈이 도시된다.

<128> 도 5에는 본 발명에 따른 파워-플래닝 모듈이 도시된다.

<129> 도 6에는 본 발명에 따른 I/O 및 아날로그/믹싱된 신호 요구가 도시된다.

<130> 도 7에는 본 발명에 따른 테스트-플래닝 모듈이 도시된다.

<131> 도 8에는 본 발명에 따른 타이밍 및 플로워-플래닝 모듈이 도시된다.

<132> Ⅳ. 블록 플래닝(106)

<133> 도 9에는 본 발명에 따라 블록 설계 단계의 흐름이 도시된다.

<134> Ⅴ. 칩 어셈블리(108)

<135> 도 10에는 본 발명에 따라 칩 어셈블리 설계 단계의 데이터 흐름이 도시된다.

<136> 도 11에는 본 발명에 따라 칩 어셈블리 설계 단계의 태스크 흐름이 도시된다.

<137> Ⅵ. 검증(110)

<138> 도 12, 13, 14 및 15에는 본 발명의 검증 설계 단계를 위한 기능적 검증이 도시된다.

<139> 가능성 판단을 위한 방법

<140> 프런트 엔드 평가로 우선 전환하면, 도 16은 본 발명에 따라 복수의 사전 설계된 회로 블록을 사용하여 회로 설계의 가능성을 판단하기 위해 본 발명의 방법을 도시한다.

<141> 도 16에서, 방법을 위한 입력은 입력으로서 사용 데이터의 필드를 이용하기 위해 처음부터 설계된다. 그러나, 새로운 설계 프로젝트를 판단할 때, 새로운 타입의 입력(1, 2 및 3)은 새로운 설계 프로젝트의 가능성을 판단하는데 사용될 필요가 있다. 방법을 수용하기 위해, 새로운 타입의 입력이 프로세싱되어 방법은 새로운 설계 프로젝트를 위한 가능성 판단을 실행하기 위해 새로운 입력을 사용할 수 있다.

<142> 도 17은 본 발명에 따라 도 16에 도시된 방법을 사용하여 가능성 판단 결과를 도시한다. 도 17은 수직축에 대한 위험 및 수평축을 따르는 시간/비용을 도시한다. 위험 지수에 따라, 이러한 세가지 타입의 새로운 데이터를 사용하는 위험은 사용 데이터의 필드를 사용할 경우 나타나는 위험과 비교할 경우 조금 증가한다. 또한 도 17로부터, 타입(3) 입력은 위험에 대해 가장 큰 영향을 갖는 것을 알 수 있다. 그러나, 새로운 데이터의 이러한 세가지 타입을 사용함으로써 시간/비용 수치에 따라, 시간/비용은 사용 데이터의 필드를 사용함으로써 생성되는 위험과 비교하여 상당히 증가한다. 도 17에 나타난 본 발명의 위험(v), 시간/비용 계산을 고려함으로써, 프리스테이징된 블록은 설계 방법에서 적절한 사용을 위해 사전 설계되며 제한된다. 프리스테이징된 설계 플랜은 바람직하게 예를 들어 블록 제작 피스인 현재의 방법의 일부분이다.

<143> 도 18은 본 발명에 따라, 복수의 사전 설계된 회로 블록을 사용하여 회로 설계의 가능성을 판단하기 위한 방법을 도시한다. 도 18에서, 방법을 위한 입력은 입력으로서의 사용 데이터의 필드를 사용하기 위해 처음부터 설정된다. 그러나, 새로운 설계 프로젝트를 판단할 경우, 새로운 타입의 입력(X, Y, Z)이 새로운 설계 프로젝트의 가능성을 판단하는데 사용되기 위해 필요하다. 새로운 입력 타입을 수용하기 위해, 방법은 새로운 입력이

새로운 설계 프로젝트를 위한 가능성 판단을 수행하기 위해 사용될 수 있도록 수정된다.

<144> 도 19는 본 발명에 따라, 도 18에서 도시된 본 발명의 설명을 사용하여 얻어진 판단된 가능성을 도시한다. 도 19는 수직축을 따르는 위험 및 수평축을 따르는 시간/비용을 도시한다. 위험 수치에 따라, 위험은 세개의 새로운 입력 타입을 사용하는 것이 사용 데이터의 필드를 사용할 경우 제공되는 위험과 비교하여 상당히 증가할 경우 제공된다. 또한 도 19로부터, 타입(Z) 입력이 위험에 대해 가장 큰 영향을 미치는 것을 알 수 있다. 그러나, 시간/비용 수치에 따라, 새로운 입력의 세 타입을 추가로 사용함으로써 제공되는 시간/비용이 사용 데이터의 필드의 사용에 의한 시간/비용과 비교하여 바람직하게 증가한다.

<145> 새로운 타입의 입력은 사전 설계된 입력을 위해 평가 데이터 또는 실행 데이터일 수 있다. 도 16-19에 도시된 결과에 기초하여, 시스템 통합자는 교환 결정을 할 수 있다.

<146> 프런트 엔드 허용에서 가능성 판단

<147> 도 1에서 프런트 엔드 허용(FEA) 설계 단계(102)는 가능성 및 제안된 설계의 판단을 포함한다. 설계는 판단된 표준이 용인할 수 있는 위험 수준 내에 있는 경우 가능하다.

<148> 어느 정도, FEA는 시스템 통합자가 제안된 설계를 수용하는 위험을 가정할 수 있는 포인트에 대한 설계 고안의 프로세스이다. 이처럼, 이는 지식의 결핍, 즉 요구된 설계의 최종 출력에서의 에러를 감소시키는 프로세스이다. 시작 포인트로서, FEA 프로세스는 고객, 설계를 받아들이는 통합자의 위험 프로파일, 사전 설계된 블록의 세트 및 사전 설계된 블록을 가진 경험에 대한 통합자의 사전 지식에 의해 전달된 설계 요구의 세트를 수신한다. 사전 설계된 블록은 다양한 레벨의 레졸루션(강성, 연성 또는 경성)일 수 있다. 레졸루션, 앞선 경험 및 블록의 이해는 블록에 걸쳐서 영역, 파워, 실행 등의 예상에서 에러 한계의 큰 영역을 발생시킨다.

<149> 각각의 블록을 위해, 설계 고안은 세 레벨의 레졸루션으로 나타날 수도 있다:

<150> (1) 통합자의 경험 필드(FOE),

<151> (2) 이러한 모델을 실행시키기 위한 실제 모델 및 도구를 사용하여 평가, 및

<152> (3) 수신되는 레벨 보다 더 높은 레벨의 설계 레졸루션으로 블록을 유지함으로 인한 딥(dip).

<153> 설계 레졸루션의 세 레벨이 연성, 경성 및 강성과 같은 오름순으로 배열된다. 모든 블록을 불필요하게 개량함이 없이 가능성 판단을 실행하고 표준 예상간의 작용하는 메카니즘을 제공함으로써 효과는 얻어진다.

<154> 도 20에는 본 발명에 따른 FEA 프로세스를 위한 흐름도가 도시된다.

<155> 도 20에서, FEA 프로세스는 전술한 세 레벨의 설계 개량을 반영하는 가능성 판단의 세 단계를 포함한다. 이러한 세 단계는 개략적인 판단(coarse-grained assessment), 기본적 판단(medium-grained assessment) 및 세밀한 판단(fine-grained assessment)이다.

<156> 개략적 판단은 유사한 설계를 가진 설계 통합자의 앞선 경험에 기초한 판단을 조절하는 경험의 필드이다. 개략적 판단은 특히 수십 개의 블록 및 시스템 설계 옵션에 특히 적합하며, 설계 에러 한계가 50 퍼센트 이상의 순서에 있는 상황에 적합하다. 개략적 분석은 고려된 블록의 피상적 평가에 사용될 수 있으며, 여기서 블록간의 작용에 대한 평가는 중요치 않다. 이런 단계에서, 고려되는 모든 블록이 최종 설계에서 사용되지는 않는 경향이다.

<157> 기본적 판단은 등가 또는 시뮬레이션을 통한 특성의 분석 형태에 의해 평가하기 위한 평가 조절 판단이다. 이는 두개 내지 10개의 시스템 설계 옵션에 적합하며, 수용가능한 설계 평가 에러 한계가 20%의 순서에 있는 상황에 적합하며, 통합자는 블록이 어떻게 상호 작용하는지에 대한 이해를 갖는다. 이는 설계의 작동 효율에 중요한 블록들 사이의 상호 작용을 검사하는데 사용될 수 있다. 이 단계에서, 고려되는 모든 블록은 최종 설계에서 사용되는 높은 가능성을 갖는다.

<158> 가장 개량된(세밀한) 판단은 블록 설계의 개량으로부터 측정하기 위한 설계-딥-조절된 판단이다. 디핑은 새로운 블록이 연성 블록, 사전 설계된 연성 블록에서 경성 블록으로 그리고 사전 개량된 경성 블록에서 강성 블록으로 전환하는 프로세스이다. 결과는 시뮬레이션, 에뮬레이션 또는 프로토타이핑이다. 세밀한 판단은 현재의 설계 개량이 충분한 중요한 현안의 최종 레졸루션동안 처럼 수용가능한 설계 평가 에러 한계가 5%미만인 단일 옵션 칩 설계의 전부 또는 부분에 적합하다. 이는 블록을 위해 존재하는 시뮬레이션 모델에 의해 제공되는 레졸루션이 충분한 것을 보장하거나 유효성을 보장하기 위해 상세하게 연구될 필요가 있는 칩 특성 또는 블록-상

호작용의 서브 세트를 검사하는데 사용된다. 이는 또한 설계 요구를 충족시키기 위해 블록의 실패를 검사하는데 사용되어, 최종 설계 가능성에 강하게 영향을 미칠 것이다. 이러한 단계에서, 고려되는 모든 블록은 디핑될 것이며, 대신에 실질적으로 FEA 결정 프로세스에 중요한 영향을 미치는 이러한 블록은 디핑된다.

<159> 도 20에서, 각각의 삼각형의 폭은 시스템 FEA 표준의 예상에서 에러를 나타낸다. 평가의 각각의 레벨에서, 주안점은 FEA가 신속하게 결정되도록 설계의 에러를 감소하는 동안 가능하면 적게 FEA 표준을 개량하는 것이다. FEA 프로세스의 각각의 단계에서, 기본적인 의도 및 전략은 동일하며, 이하와 같다:

<160> (1) 신중하게 블록에 대해 사용 가능한 정보를 수집;

<161> (2) 시스템 평가 에러에 영향을 가장 잘 미칠 이러한 블록을 식별 및 조절;

<162> (3) 설계가 FEA 억압을 받는지를 평가하며, 그러한 경우 FEA 프로세스를 중단, 그러하지 않은 경우,

<163> (4) 만일 FEA 억압이 없는 경우, 시스템에서 전체적으로 블록-평가를 조절.

<164> 도 20에 도시된 FEA 프로세스의 주요 부분은 시스템 표준의 예상에서 수용가능한 에러(또는 전체 에러)의 계산하는 방법 및 몇몇 블록이 전체적인 에러를 수용 가능한 한계 내로 하기 위해 평가 개량을 필요로하는 식별이다. 이러한 계산 프로세스는 세개의 파라미터를 필요로한다:

<165> (1) 결정을 하기 위한 수용 가능한 전체 에러의 평가;

<166> (2) 전류 시스템 분석으로부터 초래될 전체 에러의 평가; 및

<167> (3) 설계에서 특정 블록을 평가하는데 전체 에러 대 에러의 민감도(또한 블록 에러 효과로 언급됨).

<168> 제 1 파라미터는 시스템 통합자, 고객에 의해 공급된 억압 및 전체 에러의 우수한 예상의 위험-프로파일에 의해 한정되며, 이는 데이터의 전류 상태에 대한 시스템 예상에 기초하여 초래된다. 제 2 및 제 3 파라미터는 정밀한 에러 효과 곡선을 만듦으로써 모두 유도된다. 도 21은 본 발명에 따라, 주어진 에러 효과 곡선인 개량 프로세스의 구동을 도시한다.

<169> FEA 프로세스를 추가로 한정하기 위해, 본 발명은 네개의 기초 평가 기술을 사용한다:

<170> 1. FEA 결정 프로세스: 델타-인, 델타-아웃 및 델타-아웃에 기초한 결정 프로세스를 결정. (즉, 델타-아웃이 수용 가능한 위험의 평가에 어떻게 관련되었나?);

<171> 2. FEA 데이터 추출 프로세스: 델타-아웃의 발생으로 고려되는 추상적 레벨을 위해 델타-인의 완전한 세트로부터 이동;

<172> 3. FEA 블록-세분 식별: 시스템 설계내에서 추정-에러 및 블록 임계치가 주어질 때, 시스템 추정 충격을 설정하기 위한 공통 메커니즘을 한정한다(즉, 최고 포텐셜 충격 블록이 결정 프로세스에 대한 허용 표준(acceptance criteria)을 충족하지 못할 때 추가로 세분화된다; 및

<173> 4. FEA 평가(assessment)-축 매트릭스: FEA와 관련된 각각의 허용-축에 대해 사용되어질 실제 매트릭스를 한정한다(즉, 시스템내 블록의 임계치를 한정하는 방식을 한정한다).

<174> 본 발명의 방법 및 시스템에서, 추정 수정 곡선 세트가 FEA 프로세스의 타당성을 검사하는데 사용된다. 각각의 추정 수정 곡선은 FEA 축 상부에 위치하고, 이들은 시각적으로 FEA 프로세스의 타당성을 검사하기 위한 엘리먼트 및 표준을 제공한다. 추정 수정 곡선의 기능을 더 잘 설명하기 위해, 다음의 엘리먼트 및 표준이 한정된다. 집합적으로, 이러한 엘리먼트 및 표준이 허용의 FEA 축을 불린다. 이러한 한정은 블록 및 전체 시스템에 모두 적용된다.

<175> 파워 - 동작 모드당(예를 들면, mW)

<176> 성능 - 인트라-사이클 지연(예를 들면, ps/ns/us)

<177> - 대기(예를 들면, ns/us/ms)

<178> - 처리량(대상물/초-예를 들면, 50kB/sec)

<179> 영역 - 게이트, 라우팅(routing), 주변, 비사용 적색-공간(에를 들면, 일)

<180> 비용 - 비순환 엔지니어링 비용(예를 들면, U. S. $)

<181> - 유니트당 비용(예를 들면, U. S. $)

<182> 스케쥴 - 자원 할당(예를 들면, 인년(man-year))

<183> - 배달 가능 기한(시간)

<184> 위험성 - 에러 가능성(%)

<185> - 에러 충격(예를 들면, $ 및/또는 시간)

<186> FEA 프로세스를 수행하기 전에, 고객은 가능한 한 다음의 정보를 가지고 시스템 통합자를 제공한다:

<187> (1) 소프트, 펌(firm) 또는 하드 포멧중 하나인 회로 블록 세트;

<188> (2) 추정치에 대한 에러-허용오차를 따라 블록에 대한 시뮬레이터(추정기) 또는 이전-경험 추정치 세트;

<189> (3) 전체 칩 기능 및 성능 요구조건을 설명하는 내역 세트; 및

<190> (4) 수용 가능 스케쥴, 비용 미 사업 위험성에 관한 규정(stipulation) 세트.

<191> 고객은 또한 다음을 제공한다:

<192> (5) 칩에 통합될 임의의 새로운 블록에 관한 규정 세트; 및

<193> (6) 알려진 중요 문제에 대한 식별.

<194> FEA 프로세스를 수행하기 전에, 시스템 통합자는:

<195> (1) 설계 적합성이 평가되는 위험성 프로파일(profile)을 결정하는데, 이러한 결정은:

<196> a. 가드-밴드 - 각각의 FEA 축에 대한 통합자의 전체 설계 마진;

<197> b. 허용 위험성 - 고객 요구를 수용하기 전에 설계가 요구조건을 만족시키는 확실성. 이는 표준-편차 측정으로서 간단히 표현되고- Aσ 설계-허용 위험성; 및

<198> c. 거부 위험성 - 특정 설계가 사용 가능 블록으로 조립되고 제조될 수 없는 확실성. 거부가 시스템 통합자에 대한 실제적인 위험성 행동이라는 것을 주목한다: 위험성은 초기 허용가 의심스러운 것으로 표현되도록 할지라도 실제로 실행 가능했다. 이는 표준-편차-측정으로서 표현될 수 있다-Rσ 설계-거부 위험성.

<199> (2) 제출된 블록이 임의의 새로운 또는 제 3 블록과 조합하여 수용 가능한 한계의 위험성 내에서 거부 억압요인을 충족하기에 충분한 것을 입증한다.

<200> 도 22를 참조하면, 본 발명에 따른 예시적인 수정 곡선 추정치가 도시된다. 수평축은 FEA 축이고, 시스템에 대한 고객 억압요인 또는 전체 억압요인을 나타낼 수 있다. 설명을 용이하게 하기 위해, FEA 축은 파워를 나타내는 것으로 가정한다. 수직축은 추정치 수정을 나타낸다. 도 22에 다르면, 파워 억압요인의 가드밴드는 고객에 의해 초기에 설정된 억압요인과 FEA 프로세스테 의해 수정된 억압요인 사이이다. 주어진 예에서 설계는 파워 억압요인이 가드밴드에 의해 수정되고 거부 영역 내에 위치하기 때문에 거부된다. 이는 파워 억압요인이 거부 영역에 위치하지 않더라도 그러하다.

<201> 수정된 파워 억압요인이 Aσ와 Rσ사이에 있었다면, FEA 세분 프로세스는 계속되었을 것이다. 이러한 프로세스는 수용 또는 거부 결정이 세분화된 추정 수정 곡선에 기초하여 결정될 때까지 예상된 에러 변화(즉, 이러한 예에서 파워-에러 변화)를 감소시킨다.

<202> 도 23을 참조하면, 본 발명에 따른 FEA를 검증하기 위한 프로세스가 도시된다. 본 발명의 FEA 검증 프로세스는 4가지 상태를 포함한다:

<203> 0. 사전-FOE 상태(도시안됨):

<204> 허용의 각각의 FEA 축에 대한 고객 설계 억압요인을 얻는다. 요구된 가드밴드에 의해 이러한 억압요인 각각을 수정한다. 이러한 수정된 고객 억압요인이 FEA 프로세스의 입증에만 사용되고, 설계 억압요인으로서만 불린다.

<205> 1. FOE 우세 상태:

<206> 시스템 통합자는 요구된 억압요인이 충족된 것으로 보증되었는가(패스에 대한 Aσ 또는 실패에 대한 Rσ에 의해 한정된 것보다 확신이 더 클 때) 여부를 결정하도록 FOE 추정치와 추정치-에러 허용오차를 함께 조합함으로써

시작된다.

<207> (a). 만일, 제 3 블록을 고려함에도 불구하고 억압요인이 위반되었다면, 설계를 가능하지 않다. 시스템 통합자는 이러한 구성에 의해 충족된 선택사항 및 억압요인 세트를 가지고 고객에게 돌아가야 한다.

<208> (b). 만일 억압요인이 허용 가능한 위험성 내에서 충족된다면, FEA 프로세스트는 완결된다.

<209> (c). 만일 설계의 패스 또는 실패에 대한 예측 이하의 허용 가능한 확신이 존재한다면, 추정 상태는 시작되어야만 한다. 추정 상태에 들어가기 위해. "가장 잘 패스할 것 같은" 설계 구성 세트(즉, 최상) 가 선택되어야만 한다.

<210> 2. 추정 우세 상태:

<211> FOE 단계로부터 유도된 최상의 설계 세트에 대해, 임계치 식별이 수행되어야만 한다 즉, 각각의 블록에 대해 주어진 에러 허용오차가 관련되고, 이는 설계가 일정한 검증을 패스하는 것으로 검증될 것을 통계학적으로 나타낸다. 이는 블록에 대한 FOE 특정 예측의 편차의 크기와 문제시되는 설계 억압요인에 대한 블록의 충격의 곱이다.

<212> 추정은 가능한 한 중요하지 않은 설계를 정합시킴으로써 수행되고, 남겨진 것에 대한 설계 특정 추정치를 생성한다.

<213> (a) 위반: 상술된 1(a)와 유사

<214> (b) 만족:: 불확정성 레벨이 추정치의 정확도를 증가시킴으로서 추가로 감소되지 않을 때(에러-허용오차가 추정치에 이미 포함된 블록에 의해 지배되는 사실에 의해 통계학적 특정 방식으로 추정치를 개선시키지 않은 정합양을 감소시킴으로써), 또는 SOC 설계의 전체 추정치가 주어진 현존하는 블록 모델을 구성한다면, 최상의 설계는 딥핑 상태로 패스되어야 한다.

<215> 3. 설계-딥 우세 상태:

<216> 전체 에러가 가장 민감한 블록 추정치를 한정하면, 추정 상태당 진행된다. FEA가 검증 또는 부정될 때까지 이러한 프로세스를 반복한다. 통계학적 임계치에 대한 한정도 유사하다.

<217> 도 24를 참조하면, 본 발명의 FEA 설계-특성 세분 프로세스를 사용하여 세분된 추정치 수정 곡선이 도시된다. 상술된, FEA로부터 이동하는 세분 프로세스가 0 내지 3으로 이동되지만, 세분화된 추정 수정 곡선상의 예상되는 에러 편차는 도 22에 도시된 세분된 추정 수정 곡선과 비교하여 상당히 감소된다. 따라서, 수용 또는 거부에 대한 결정은 도 24에 도시된 바와 같은 세분화된 추정 수정 곡선에 기초하는 반면, 이러한 결정은 도 22에 도시된 추정 수정 곡선에 기초하여 결정될 수도 있고 결정되지 않을 수도 있다.

<218> FEA 결정이 검증의 하나의 상태에서 이용 가능한 정보 및 데이터에 기초하여 결정될 수 없다면, 본 발명은 예상된 에러 편차를 감소시키는 설계-특성 세분 프로세스를 수행한다. 세분화된 데이터 및 정보에 기초하여, 본 발명은 다음 단계에서 FEA 검증을 수행한다. 설계-특성 세분 프로세스는 이하의 3가지 특성을 가진다:

<219> (1) FEA 데이터-추출 프로세스;

<220> (2) FEA 블록-세분 식별; 및

<221> (3) FEA 평가-축 매트릭스.

<222> 도 25를 참조하면, 본 발명에 따른 FEA 데이터-추출 프로세스가 도시된다. 시스템 설계내 각각의 블록 블록과 관련된 예상 데이터에 대한 "시스템 충격에 대한 추정"을 수행하기 위한 표준화된 메커니즘 또는 프로세스가 있다. 이러한 메커니즘은 블록-세분화 식별로서 불리며, 임의의 특정 블록의 특성(FEA 설계 표준---예를 들면, 파워, 영역, 성능 등)에 대한 요구된 에러-경계가 FEA 시스템-설계 평가의 각각의 세분 상태에 대해 결정될 수 있도록 한다.

<223> L(β)가 임의의 요구된 설계 마진에 의해 수정된 바와 같이, FEA 표준 β를 만족시키는 설계에 대한 고객에 의해 한정된 한계라 하자. FEA 표준 β에 대해 측정된 설계의 예상치를 E(β)라 하자. FEA 표준 β에 대한 패스/실패로서 한정될 설계의 설계 결정 억압요인 또는 "최대 에러 허용치"가 $DDC(\beta)=|L(\beta)-E(\beta)|$로서 주어진다. 예상된 "패스"에 대해, E(β) 그 자체는 FEA 표준에 대한 허용 가능 영역내에 위치해야 하고, 예상된 "실패" 에 대해 E(β)는 거부 영역내에 위치해야 한다. 효과적으로, "패스"의 가장 처음의 경우에, 우리는 Aσ

$_{SYSTEM}$〈DDC를 요구하고, "실패"에 대한 두 번째 경우, Rσ_{SYSTEM}〈DDC를 요구한다. 만일 불균형(inequality)이 만족되지 않는다면, 시스템 분석은 결정-품질 결과를 생성하지 않는다.

<224> 일반적으로, 평균 추정치 E(β)는 시스템-평가의 이전 상태에 의해 생성된 바와 같이 시스템-표준 β의 최종 추정치이다 즉, 중간 그레인 평가 단계가 코오스(coarse) 평가 단계의 최종 추정치를 평균함에 따라 수행되고, 최종 그레인 평가 단계는 중간 그레인 평가 단계의 최종 추정치를 평균함으로써 수행된다. 프로세스를 시작하기 위해, 코오스 평가 단계는 각각의 FEA 표준에 대한 코오스-레벨 예상치 추정를 가장 먼저 수행함으로써 시작된다.

<225> 특정 FEA 표준 β에 대한 설계 결정 억압요인(DDC)에 대해 평가될 시스템에 대해, 블록 추정치와 관련된 에러 및 시스템에 대한 전체 추정치 에러 사이에 관계가 설립되어야 한다. 블록 추정치와 관련된 에러가 블록에 대한 β-표준을 추정하는 고유 에러만일 뿐만 아니라 통합(integration) 비용의 어려움에 따른 블록 및 블록-에러의 특정 영향이라는 것을 주목한다. 블록을 추정하는데 있어서의 에러는결과적으로 시스템-임계치 측정값 C에 의해 스케일링되고, 이는 FEA 표준 β에 대한 자신의 특성 또는 부족-또는 한정(에러)에 기초한 블록을 통합하는데 있어서의 어려움의 측정값이다. 시스템의 패스(실패)에 관한 결정은 $\{C_{block}.\sigma_{block} \mid block \in system\}$ 내지 σ_{SYSTEM}의 집합과 요구된 불균형 사이의 관계에 의해 설립된다: 각각의 FEA 표준에 대해 $A\sigma_{system} < DDC$ ($R\sigma_{system} < DDC$).

<226> 상술된 시스템 불균형과 관련된 임계치 측정값 C_{block}의 포함을 유지하기 위해(즉, 임계치 스케일링된 블록 에러를 조합하는 표현으로부터 σ_{SYSTEM}이 공식화된다: C_{block}, σ_{block}), 임계치 측정치는 표준화된다: $\Sigma_{blocks}(C_{block})^2 = 1$. 이를 평가하기 위한 프로세스는 평가될 시스템-특성의 등급(class)에 따라 약간 변한다. FEA에 대해, 각각 후술되는 시스템-특성의 3가지 등급이 존재한다:

<227> • 절대 (블록) 억압요인 (예를 들면, 인트라-사이클 지연, 처리량)

<228> • 상대 (블록) 억압요인 (예를 들면, 파워, 영역, 잠재시간, 비용, 스케쥴)

<229> • 혼합 (블록) 억압요인 (예를 들면, 품질).

<230> 간략함을 위해, FEA 표준 β는 블록 설계 억압요인으로서 BDC를 한정하고 여기서: 설계 수용에 대한 테스트의 경우 $BDC_{block}=A.C_{block}.\sigma_{block}$이고, 설계 거절에 대한 테스트의 경우 $BDC_{block}=R.C_{block}.\sigma_{block}$이다. 다음으로, 각각의 FEA 표준은:

<231> a. 절대 억압요인: 결정-품질 결과를 얻기 위해, 각각의 블록 또는 중간 환경에 놓인 각각의 블록(예를 들면, 라우팅 로드 등 포함)은 절대 억압요인에 대한 DDC를 패스해야만 한다. 수학적으로, 절대 억압요인에 대한 결정-품질 결과의 획득은 다음을 포함한다:

<232> 시스템내 모든 블록에 대해, BDC_{block}〈DDC.

<233> b. 상대 억압요인: 결정 품질 결과는 만일 시스템을 통해 블록-설계 억압요인의 제곱의 합이 DDC의 제곱보다 작을 때 얻어진다. 상대적이라는 용어는 이러한 억압요인에 대한 평가의 수용 가능한 에러가 블록중에 분산된 융통성을 가질 때 사용되고, 이는 전체 시스템을 형성한다. 상대 형태의 몇몇 평가 표준이 다수의 억압요인이라는 것을 주목한다. 이에 대한 한 예가 대기이고, 여러 표준 경로가 가능하며, 이는 완전한 시스템의 정당한 평가에 기여한다. 수학적으로, 임의-변수와는 무관하게 모든 블록-에러가 가우시안-분포라고 가정한다면, 상대 억압요인에 대한 결정-품질 결과는 $\sum_{blocks}(BDC_{block})^2 < DDC^2$ 을 의미한다.

<234> c. 혼합 억압요인: 혼합 억압요인은 억압요인의 상대 형태와 절대 형태모두를 포함한다. 예를 들면, 품질은 혼합 억압요인이다. 설계내 어떠한 블록도 품질에 대한 자신의 측정값에 기인한 특정 경계치를 초과할 수 없지만, 시스템 전체에 대한 모든 품질 평가에 대한 합은 특정 범위내에 들어가야만 한다. 이 경우, 블록에 대한 DDC_{block}뿐만 아니라 전체 시스템에 대한 DDC_{system}이 존재한다. 수학적으로, 혼합 억압요인 시스템 특성에 대해 두 개의 표준이 만족되어야 한다:

<235> (i) 모두에 대해: 블록 & 시스템, $BDC_{block} < DDC_{block}$

<236> (ii) $\sum_{blocks}(BDC_{block})^2 < (DDC_{system})^2$.

<237> 도 26을 참조하면, 본 발명에 따른 블록-추정 세분화의 필요성을 검증하는 프로세스가 도시된다. 도시된 바와 같이, FEA 블록-세분화 검증내 3 단계는:

<238> 1. 절대 또는 혼합 형태의 억압요인의 각각의 FEA 평가 표준에 대해, 절대 에러 허용 오차(CIC's)를 달성하는데 필요한 작업 레벨이 결정된다. 절대 억압요인의 필요성을 만족시키는 모델을 개선하는 것에 따른 부산물로서, 상대 억압요인과 관련된 몇몇 에러-경계가 감소된다.

<239> 2. 모델이 절대 억압요인을 만족시키도록 세분화된 이후 예상된 에러에 기초하여, 혼합 억압요인 형태의 절대값 부분, 시스템에 대한 나머지 시스템-에러 허용오차(CIC)가 분리 IP 블록중에서 결정되고 분할된다. 분할은 추정치를 생성하는데 필요한 작업을 최소화하는 방식으로 결정된다. 이러한 분할의 융통성은 조립된 시스템내의 각각의 블록에 대한 분포의 세분화된 임계치에 의해 변조된다. 이는 에러 충격의 노션을 결정한다. 이러한 문제점은 각각의 FEA 축을 따른 허용 가능한 에러-허용오차에 대해 필요한 작업을 동시에 최적화해야 한다.

<240> 3. 만일 제안된 CIC's를 사용하여 어떠한 단계 적합성도 결정될 수 없다면, 이들은 추가로 조밀해져야하고 프로세스는 반복되어야 한다:

<241> (a) 블록에 대해, 만일 특정 절대 억압요인이 불충분할 때, 또는

<242> (b) 시스템에 대해, 칩에 대한 상대 억압요인이 불충분할 때.

<243> 도 27을 참조하면, 본 발명에 따른 평가-축 임계치(AAC)의 개념을 한정하는 테이블을 포함하는 FEA 평가-축 매트릭스가 도시되고, 이러한 매트릭스는 적절하게는 예시적인 임계치 측정치를 포함한다. AAC는 이하의 관계식에 기초하여 예상 추정치(EEE)를 통해 예상 시스템-충격(ESI)에 관계된다: ESI=AAC * EEE.

<244> 도 27에 도시된 바와 같이, 테이블은 이하와 같은 5개의 컬럼을 포함한다.

<245> (1) 평가 축 FEA 이러한 표준에 기초하여 측정된다

<246> (2) 억압요인 형태 각각의 FEA 축은 축과 관련된 하나 또는 다수의 억압요인 형태를 가진다.

<247> (3) 억압요인 등급 상술된 등급

<248> (4) 라우팅 세분화 칩 라우팅의 충격이 특정 블록 및 시스템 억압요인과 같은 에러 정도를 가지도록 하기에 필요한 라우팅-세분화 형태

<249> (5) 임계치 측정 FEA 평가 축과 관련된 특성의 임계치를 측정하기는 표준화된 방법.

<250> 몇몇 테이블의 엘리먼트는 라우팅 임계치에 대한 참조를 형성한다. 라우팅 임계치는 핀 라우팅 임계치와 같이 블록의 출력 핀 또는 칩 입력 패드에 대해 한정된다. 블록 라우팅 임계치는 블록의 라우팅 임계치 및 출력의 합이다.

<251> 심볼: α는 유효 라우팅 영역 스칼라를 한정하고, α^*(라우팅 임계치)는 유니트와 라우팅 임계치의 스케일을 영역-적용 가능 수로 변환한다.

<252> 라우팅 결과로서 소비된 파워는 라인상의 활성의 추정치를 필요로 한다. 이는 분해능의 블록 또는 핀 레벨에서 수행될 수 있다. 블록에 제공될 때, 활성 추정치는 E_{block}으로 표시된 바와 같이 블록의 출력 라인상의 평균 활성으로부터 유도된다.

<253> 여러 팬아웃(fanout)이 공유된 버스의 사용에의해 연결되지 않는다면 어떠한 팬아웃 포인트로서 포인트 연결이 카운트된다. 공유 버스는 단일 불연속 블록으로서 카운트된다. 라우팅 임계치는 핀에 라우팅 연결에 있어서의 예상된 어려움의 측정치이고, 그러므로 FEA 불확실성의 측정치이다.

<254> 많은 평가 축이 분해능의 몇몇 레벨에서 혼합된 억압요인으로서 검증되는 것을 주목한다; 예를 들면, 영역은 초기 플로어(floor) 플랜이 한정된 이후 혼합된 것으로 한정되고 블록-레벨 억압요인으로 SOC 설계 칩-레벨 억압요인을 분할하는데 사용된다. 하지만, 빠른 FEA 주기 동안 사용된 우세 억압요인 형태가 리스트된다.

<255> 테이블내에서 사용된 에러라는 용어는 문제시되는 특성에 관한 에러상의 경계를 칭한다.

<256> 경험 데이터 필드 조직(organizing)

<257> 설계자 경험은 BBD 방법론의 시스템-결정 프로세스내 중요부이다. BBD 방법론은 단일 주요 설계자 또는 설계자와 관련된 경험의 개념을 "회사 설계 경험"의 개념으로 연장한다. 이러한 일반적인 경험의 "풀(pool)은 본 발명의 BBD 경험의 필드(FOE)로서 불린다.

<258> FOE의 설계와 사용에 대한 4가지 컨셉과 메커니즘을 제안하는 것이 BBD 방법의 목적이다. 이러한 컨셉은:

<259> a) 데이터 개더링(gathering) - FOE 데이터를 입수하고 시작하기 위한 논리적으로 정확한 프로세스의 정의.

<260> b) 데이터 분류 - 관련 분류를 전개하기 위한 정보 분류와 메커니즘. 이러한 분류는 개더링된 데이터가 축적된 설계-지식의 양이 증가됨에 따라 통계학적으로 분석, 외삽 및 전체적으로 세분화되는 것을 보장한다.

<261> c) 데이터 검증 -"경험 법칙(rule-of-thumb)" 수로 불리는 "트러스트"의 수정 보장을 설계하는 프로세스 정의. FOE 데이터 검증은 FOE 데이터베이스로부터 수립된 추정치가 통계학적으로 잘 경계지어지는 것을 보장한다.

<262> d) 데이터 적용 - FOE를 설계 프로세스에 적용하기 위한 메커니즘. 이는 BBD를 위한 프런트 엔드 허용의 일부이다.

<263> 경험적 정의 필드

<264> BBD에서, 경험 필드는 설계 스타일, 설계 목적 및 설계 특성의 임계 측정치에 따라 분류된 이전의 설계의 측정치로부터 정의된다. 중요 특성은 :영역, 처리량, 파워 및 대기를 포함한다. 경험-기초 추정치의 정의는 설계 정도는 설계 특성으로 인한 경험에 기초한 시스템 예측이다. FOE 추정치의 정의는 FOE 데이터를 사용하는 경험 기초 추정이다.

<265> 이것이 문제시되는 설계의 특정 분석을 내포하지 않는다는 점에서-- 또는 하드웨어 설계가 이전의 공지된 사실로부터 실질적으로 알려져있다면 --그러한 하드웨어의 요구된 새로운 행동의 특정 분석이 BBD 추정으로부터 분리된다는 것을 주목한다. 예를 들면, DSP 코어는 코어의 이전의 설치에서 회사 및 FIR-필터 삽입된 루틴 런내에서 개발된다. 이는 동일한 코어에서 고려되는 FFT 알고리즘의 사용을 필요로 한다. 그러한 제 1 경험 법칙은 설계에 대해 FIR 동작을 수행할 때 관찰된 이전의 알고리즘 효율만에 기초하지만, FFT 알고리즘에 대해 매우 특정화된 데이터로 들어가지 않는다면, 이는 FOE 추정치이다.

<266> 경험 필드는 이전 설계 프로젝트 세트 동안 유도된 정보에 따라 사용되어야 한다. FOE 데이터는 표준 데이터베이스를 통해 카탈로그, 저장 및 액세스될 수 있어야 한다.

<267> 설계에서 사용된 3가지 다른 등급의 경험-기초 데이터가 있고, 각각은 특정 데이터 프로파일과 관련된다:

<268> a) 프로젝트 데이터 - 프로젝트 시간에 설계자-요구된 추정치. 설계자는 FOE 데이터베이스내 로깅된 바와 같이 다른 사람의 경험에 기초하지 않고 자기 자신의 카탈로그되지 않은 설계 경험에 더욱 기초한다. 설계 추정에서의 에러는 설계자-에러 편차에 의해 주어지고, 이는 일반적인 설계에서 발견된다. 설계자-에러 편차는 결과를 정확하게 예상할 수 있는 설계자의 능력의 일반적인 히스토리를 측정함으로써 성립된다.

<269> b) 프로젝트된 데이터 - 소망하는 측정 프로젝트없이 설계 분류 내에서, 설계자는 현존하는 FOE 데이터를 확장시키기 위해 자신의 최상의 파라미터-관계에 요구되어 진다. 이 경우, 확장된 FOE 데이터는 단일 설계-포인트로만큼 적게 구성된다. 이에 대한 에러는 파라미터화 에러에서 설계자의 최상의 추측에 의해 특정화된 부분내에 있지만, 또한 결과를 정확하게 예상하는 설계자의 능력의 히스토리에 의해 수정된다. 통계학적으로 독립성을 가정할 때, 이러한 에러는 합산된다.

<270> c) 대조된 데이터 - 설계 경험 세트로부터 대조되고 분류되며 파라미터화된 데이터. 이러한 데이터와 직접 관련된 측정 에러의 가능성이 있지만, 이는 아주 적을 것이다. 주 에러는 측정된 결과와 데이터-파라미터의 편차에 의해 예상된 결과 사이의 차이로서 정의된다.

<271> 프로젝트 데이터는 추가의 설계에 대한 현재의 추정치를 확장시키는 어떠한 메커니즘도 제공하지 못하기 때문에 FOE 데이터의 한 형태가 아니라는 점을 주목한다. 게다가, 프로젝트 데이터가 프로젝트의 시작시 완전하게는 아니지만 개더링되기 때문에, 카탈로그된 설계 경험에 대해 입증할 수 없다. 이는 검증되지 않았음을 의미한다. 설계의 최종 측정으로부터 개더링된 데이터는 FOE 데이터베이스로 들어가고, 프로젝트 데이터 대 최종 측정치의 정확도가 회사에 대한 설계자 에러 편차를 세분화하는데 사용된다.

<272> 프로젝트된 데이터는 FOE 시드-데이터라 불린다. 프로젝트된 데이터는 유사 설계에 FOE 추정에 즉시 적용된다.

<273> 수용된 데이터 형태의 공통적인 분류는 FOE 데이터의 상기 소스 모두에 적용되어야 한다. 이러한 공통의 분류는 수용된 데이터의 빠른 검증 및 카탈로그를 허용한다. 초기 분류-특정사항은 FOE에 대한 플래닝 단계로서 간주되고, 데이터의 엔터링/개더링은 빌딩 단계이다. FOE 데이터내 정보량이 증가됨에 따라, 세분화 프로세스는 통계학적으로 관찰된 것 내에서 에러 허용 오차를 감소시키도록 제공된다. 이러한 모든 3 단계와 병행하여, FOE 입증 프로세스가 수행된다.

<274> 상기 리스팅된 파라미터는 현존하는 일반적인 FOE 데이터로부터 프로젝트-특정 FOE 추정치를 유도하여 외삽하는데 사용된다. 이러한 외삽된 추정치와 FOE 데이터 사이의 관계는 바람직하게는 각각의 분류에 대해 정의된다. 각각의 파라미터 FOE 관계는 설계자의 개인적인 경험에 의해 정의되고(상술된 프로젝트된 데이터 참조), 또는 충분한 정보가 입수 가능하다면 FOE-데이터를 곡선-정합시킴으로써 실험적으로 특정화된다. 파라미터는 파이프라인 깊이, 평행도, 비트-폭 및 클록킹-속도로서 이러한 기술적인 변수를 포함한다.

<275> FOE는 블록을 설계할 뿐만 아니라 블록 사이를 상호결합시키는데 사용된다는 것을 주목한다. 이 경우, FOE는 하나의 분류의 블록과 다른 하나의 블록 사이의 라우팅 비용으로서 특정된다. 블록의 응용과 같이, 상호결합에 대한 FOE 추정치는 파라미터화된다.

<276> 최대 정확성을 가진 추정

<277> FOE의 주요 특성은 주어진 데이터에 관한 최대 정확도 추정의 일반화이다. 이는 2단계 프로세스이다:

<278> a) 세분 - 상술된 바와 같이, 세분은 통계학적으로 관찰된 범위내로 에러-추정치를 감소시키는 프로세스이다. 즉, 특정 카타고리내 FOE 데이터의 양이 적을 때, 데이터에 대한 에러 허용오차는 크다. 이는 고유 에러때문이 아니라 오히려 알려지지 않은(또는 시험되지 않은) 다른 특정 설계에 대한 파라미터화된 데이터의 적용 가능성이다. 시험된 설계의 수가 증가됨에 따라, 데이터의 통계학적 분산이 파라미터화된 예상에 대해 직접 측정된다. 특정 분류의 설계에 대해 큰 경우의 수가 카탈로그될 때, 파라미터와 방법의 정확성은 잘 성립될 것이다. 많은 상호관련된 에러의 검증은 (데이터의 임의 분산에 반대됨) 파라미터 관계를 재고찰할 수 있도록 한다.

<279> b) 분류 붕괴 - 설계의 다른 분류는 상호 근접에 의해 관계를 가지게 된다. 예를 들면, 버터플라이 FFT 실행은 설계의 하나의 분류이지만, 모든 FFT 블록은 이러한 설계에 매우 근접한 것으로 간주된다. 만일 원하는 특정 분류와 관련된 데이터의 수가 너무 작아서 통계학적으로 의미가 없다면, 초근접 FOE 데이터는 함께 붕괴되어 전체 추정 에러를 감소시킨다. 분류의 붕괴는 함께 설계 형태에서의 약간의 차이로 인해 에러를 유도하지만, 고려된 설계가 수적 견지에서 통계학적 개선은 이러한 에러차를 압도할 수 있다. 도 28에 도시된 바와 같은 곡선을 계산하고 및 최고 에러의 구성을 표시하는 것이 바람직하다.

<280> 그러므로, FOE에 대한 프로세스/사용 모델은 다음과 같다:

<281> I. 평가되는 블록에 적용 가능한 블록 분류를 선택

<282> II. 분류에 대해 충분한 데이터가 존재하는가?(즉, 예상 에러가 충분한가?)

<283> 예 - 최고 FOE 추정치로 돌아가서 종료

<284> 아니오 - 계속 진행

<285> III. 추정 에러가 개선되는 것을 중지할 때까지 초근접 카탈로그를 붕괴

<286> IV. 예상된 에러가 FOE 추정에 대해 충분한가?

<287> 예 - 최고 FOE 추정치로 돌아가서 종료

<288> 아니오 - 계속 진행

<289> V. 설계자에게 설계에 대한 자신의 최고 추측을 생성하였는지 묻는다. (이는 BBD의 추정 상태가 된다)

<290> FOE 검증

<291> FOE 검증은 개더링된 FOE 정보가 신뢰성 있는 것으로 도시되는 프로세스이다. 이러한 검증 프로세스는 설립 및 세분 단계 동안 추정 에러를 설립할 것이다.

<292> 검증에는 두 가지 특성이 있다:

<293> a) 완벽함의 검증 - 모든 FEA 매트릭스는 제공된 파라미터화 설계를 통해 측정될 수 있어야 한다.

<294> b) 정확성의 검증 - 수집된 데이터의 정확성을 보장하기 위해 설계자에 대한 경험 측정 및 프로세스의 정의.

<295> 글루 로직

<296> 본 발명은 개선된 글루 로직 분포 및 감소 방법론에 관해 추가로 개시한다. 3개의 선택 가능한 글루 로직 분포 메커니즘의 조합은 본 발명의 바람직한 실시예를 포함한다. 가장 먼저, 미리 설계된 블록으로 통합되지 않는 글루 로직이 존재하는 블록에 대한 분포의 다수의 복사본으로 복사될 수 있다. 다음으로, 최상 레벨에서 블록에 대한 어떠한 친화력도 가지지 않는 로직은 작은 블록으로 남겨질 수 있고, 선택적으로 유효 모노폴라리제이션, 와이어링 밀집(congestion) 및 플로어플래닝 충격으로서 남겨질 수 있다. 세 번째, 블록의 수가 블록 위치와 루트(route) 제한을 초과할 때, 글루 로직은 블록 카운트가 허용 가능한 레벨로 감소될 때까지 글루 클러스터 블록으로 클러스터된다.

<297> 도 29를 참조하면, 글루 로직(2910)이 상호 연결된 블록 사이의 바람직하지 않게 존재하여, 실리콘 실제 평가의 상당한 영역의 사용을 비효율적이 되도록 하고 상당한 와이어링 밀집을 형성하는 회로 설계가 도시된다.

<298> 도 30을 참조하면, 더 큰 최상-레벨 블록에 대한 분포의 글루 로닉의 다수의 복사본을 형성하기 위한 본 발명의 방법의 설명으로부터 시작된다. 만일 엘리먼트(3010)가 다수의 로드를 유도하는 출력 네트를 가진다면, 엘리먼트는 다수의 엘리먼트(3012)로 분할되고, 각각의 엘리먼트는 출력에 단일 로드만을 갖는다. 다시, 복사된 엘리먼트를 유도하는 각각의 입력 "콘"(도시 안됨)은 모든 블록 출력이 도달될 때까지 잘 복사된다. 유사하게, 큰 입력 게이트가 트리의 최상부에서 원래 함수의 2-입력 게이트를 가지고, 비전환 2-입력 게이트의 트리로 감소된다. 이러한 방법으로, 실질적으로 더 많은 로직이 이전의 더 적은 글루 로직 함수로 집중된다. 하지만, 큰 블록 사이의 영역으로부터 글루 로직을 제거함으로써, 큰 블록은 더욱 효율적으로 위치할 수 있고, 전체 효율이 증가된다.

<299> 분포에 대해 효율적으로 복사될 수 없는 글루 로직 엘리먼트는 바람직하게는 위치된 엘리먼트에 가장 인접한 친화력을 가진 큰 블록으로 통합된다. 글루 로직 통합은 다수의 표준에 기초하여 수행되고, 다수의 표준중 가장 중요한 표준은 머저(merger) 상부-레벨 핀-아웃의 수를 감소시키는지의 여부이다. 따라서, 다중 복사가 생성될 때, 대부분의 결과적인 로직이 두개의 입력 게이트로 이루어지기 때문에 하나의 핀이 접속되는 블록들로 게이트를 머징하는 것은 핀의 총수를 2만큼 감소시킨다. 두개 이상의 블록이 머저에 대해 동일한 후보일때 가장 낮은 핀 밀도를 가진 블록이 바람직하게 선택된다. 최종적으로, 가장 낮은 우선순위는 바람직하게 타이밍에 대한 고려대상이 된다.

<300> 다음에, 도 31을 참조하면, 머징될 수 없는 게이트 및 작은 블록(3110)은 클러스터(3112)로 클러스터링된다. 머징될 수 없는 게이트는 그들의 입력 및 출력 네트상에 다중 로드를 가질 가능성이 가장 높다. 유사한 기능을 가진 입력과 게이트를 결합함으로써, 게이트의 총수는 감소될 수 있다.

<301> 본 발명은 미리 설계된 회로 블록을 표준화된 인터페이스를 가진 회로로 변환하는 방법을 더 기술한다.

<302> 도 1에서 블록 설계 단계(106)에서 실행되는 태스크는 선택된 회로 블록에 대한 임의의 손실 요약을 생성하는 단계와, (2) 칼라로써 공지된 각각의 표준화된 인터페이스에 회로 블록을 삽입하는 단계와, (3) 칼라회로 블록에 대한 완전한 요약 세트를 생성하는 단계를 포함한다.

<303> 도 32에는 본 발명에 따라 회로블록을 칼라에 삽입하는 칼라링 프로세스가 도시되어 있다.

<304> BBD 방법에서, 선택된 회로블록은 칩 레벨에서 주요 입력 구성요소이다. 칼라링 프로세스는 회로블록의 경계 둘레에 표준 인터페이스를 생성하기 위하여 각각의 회로 블록을 둘레에 칼라를 배치한다. 칼라 블록을 칩 레벨에 성공적으로 통합하기 위하여, 완전한 요약 세트는 칼라 블록을 위하여 생성되어야 한다. 칼라 블록에 대한 완전한 요약 세트를 생성하기전에, 본 발명의 시스템은 선택된 블록에 대한 임의의 손실 요약을 우선 형성한다. 예시적인 기판은 (1) 정적 타이밍 요약-TLF, (2) 레이아웃 블록파일-LEF, (3)검사에 대한 모델, 볼트-버스-블록 모델,및 (4) 시스템에 정해진 블록 레이아웃을 포함한다.

<305> 도 33에는 본 발명에 따라 회로 블록의 완전한 요약세트가 기술되어 있는 반면에, 도 34는 도 32 및 도 33에 기술된 특징의 결합을 기술하고 있다.

<306> 다음으로, 칼라링 프로세스에 대해 설명할 것이다. 표준 인터페이스는 설계시 사용될 블록의 각 형태에 대해

정의되었다는 것이 가정된다.

<307> 제 1 단계에서, 프로세스는 각각의 블록이 완성된 블록 요약을 가지는지의 여부를 검사한다. 만일 임의의 블록이 안전한 블록 요약을 가지지 않는다면, 프로세스는 블록에 대한 완전한 블록 요약을 형성한다.

<308> 다음에, 프로세스는 각각의 블록에 대한 블록형태를 식별한다. 특히, 블록은 메모리 형태, 프로세서 형태, 파워 형태 또는 아날로그/혼합 신호 형태일 수 있다. 그러나, 다른 소스로부터의 회로 블록의 형태는 다른 회로 블록을 접속하기 위하여 다른 설계를 요구하는 다른 인터페이스를 가질 수 있다. 예컨대, 다른 벤더에 의하여 설계된 프로세서는 다른 인터페이스 및 버스 구조를 가질 수있다.

<309> 다음에, 프로세스는 식별된 블록을 그것의 각각의 인터페이스 표준과 연관시킨다.

<310> 그 다음에, 프로세스는 식별된 블록의 특정 인터페이스에 접속가능한 구성요소를 포함하는 제 1 칼라 부분을 생성한다.

<311> 다음 단계에서, 프로세스는 식별된 회로블록과 연관된 표준 인터페이스를 컴플라이언스에서 제 2 칼라 부분을 생성한다.

<312> 그 다음에, 프로세스는 표준 인터페이스에 접속가능한 포맷으로 특정 인터페이스를 변환시키기 위한 구성요소를 포함하며 제 1 칼라 부분을 제 2 칼라부분에 접속하는 제 3 칼라부분을 생성한다.

<313> 블록 칼라는 다중 층으로 이루어질 수 있다. 현재, 두개의 칼라층(블록 표준 칼라 및 시스템 특정 칼라)은 BBD 및 SOC에 대해 정의되었다. 도 35에는 두개의 층을 포함하는 칼라가 도시되어 있으며, 하나의 칼라는 특정 블록에 대한 표준이며, 다른 칼라는 블록이 배치되는 특정 시스템에 특정된다. 블록 표준 칼라는 통합될 수 있는 특정 콘텍스트 또는 특정 시스템에 대한 지식없이 한정될 수 있는 인터페이스 구성요소를 포함한다. 예컨대, BBD의 콘텍스트에서, 특정 설계 그룹은 JTAG- 표준 테스트 인터페이스가 설계시 요구되는 것을 결정할 수 있다. 따라서, 설계된 임의의 시스템에 사용될 모든 블록에 대하여, JTAG 테스트 인터페이스는 표준이며 블록 표준 칼라에 속한다. 시스템 특정 칼라(또는 적응 칼라)는 블록에 속하는 인터페이스 구성요소를 포함하나 특정 시스템 또는 콘텍스트이다. 예컨대, 데이터 라인에 대한 표준 세트는 패리티 비트를 필요로하지 않으나 설계된 특정 시스템에 대해서는 모든 데이터 라인에 대하여 패리티 비트를 필요로한다. 패리티 비트를 발생시키는 로직은 칩 설계동안 블록과 연관되며 시스템 특정 칼라에 존재해야 한다.

<314> BBD에서 두개의 칼라층사이의 다른 구별은 블록 표준 칼라가 정면단부 수용전에 그리고 칩 플래닝(칩 설계는 최초 칼라가 요구된 칩 플래닝 기능을 보다 잘 수행하는 담금 프로세스의 일부분으로써 설계되는 것을 요구할 수 있다.)전에 입혀질 수 있으나 시스템 특정 칼라가 칩 플래닝이후에만 부가될 수 있다는 점이다.

<315> 두개의 칼라 형태간의 더 미세한 차이는 블록 표준 칼라에 대한 표준세트가 SOC에서의 표준 세트보다 훨씬 좁은 범위일 수 있다는 점이다. 예컨대, 임의의 파워 인터페이스는 BBC에 대한 표준 일 수 있으나 단지 특정 회사의 표준일 수 있으며, 다른 회사는 블록에 대한 표준 파워 인터페이스에 따를 필요가 없다. 결과적으로, 회사 이외의 블록은 표준 파워 인터페이스를 회사의 인터페이스로 변환하는 시스템 특정 칼라를 필요로한다. 이는 SOC와 대조적이며, 산업 전반의 파워 인터페이스 표준은 블록 표준 칼라에서 존재한다. SOC의 궁극적인 목표는 산업 전반의 표준인 표준 칼라를 생성하는 것이다. 이러한 칼라를 가지는 블록은 소켓형 블록이라 부른다. 미래에, 만일 칼라의 모든 특성이 산업 전반에 걸쳐 사용된다면, 시스템 특정 칼라의 추가 레이어링에 대판 필요성이 존재하지 않아서 블록이 플러그 및 플레이의 이상에 매우 근접하도록 한다.

<316> 시스템 특정 칼라에 대한 다른 중요성은 비록 그것이 칩 플래닝후에 설계될지라도 칩 설계시 시스템 특정 칼라를 만들어 칩 집적 프로세스의 속도를 높이는 것이다. 시스템 특정 칼라의 범위를 포착하기 위한 파라미터는 목표가 되어야 할 것이다. 이는 칩 플래닝후에 시스템 특정 칼라가 스크래치로부터 재설계되지 않는 반면에 단지 파라미터만이 변화하기 때문에 집적회로 프로세스의 속도를 높일 수 있다.

<317> 칼라 및 블록은 부드럽고 견고하고 강한 다양한 결합을 이룰 수 있다. 블록의 경도에 관한 단점이 존재하기 때문에, 칼라의 연화성, 견고성 및 경도의 결합에 대한 장점 및 단점이 존재한다. 예컨대, 블록 그자체가 부드럽기 때문에, 시스템 특정 칼라가 첨가될때 전체 블록이 최종 레이아웃의 변경시 합성, 배열 및 라운팅된 평면일 수 있도록 블록 표준 칼라를 남겨두는 것이 바람직하다. 블록이 강한 경우에, 시스템 특정 문제를 조절하기 위한 연한 시스템 특정 칼라가 그 기능을 변화시킬 가능성이 높기 때문에 미세한 표준 기능성 변화를 가진 우세한 물리적인 인터페이스 문제를 조절하기 위하여 강한 블록 표준 칼라를 사용하는 것이 바람직하다.

<318> 칼라는 블록 특정 인터페이스를 다음과 같은 단계를 포함하는 방법으로 표준 인터페이스로 변환한다.

<319> (1) 블록에 특정한 물리적인 구조를 핀층, 핀 위치 및 핀 분리를 갖는 표준 물리구조로 변환하는 단계, (2) 파워 로딩 및 파워 물리적 위치를 포함하는 전원으로 블록에 특정한 전원을 변환하는 단계, (3) 테스트 액세스 포트(TAP) 및 테스트 프로토콜을 포함하는 표준 테스트 프로세스로 블록에 특정한 테스트 프로세스를 변환시키는 단계, (4) 셋업 및 홀드 시간, 플립-플롭 또는 래치를 포함하는 표준 타이밍으로 블록에 특정한 타이밍을 변환하는 단계, (5) 각각의 클록 포트의 로딩을 포함하는 표준 클록 포트로 블록에 특정한 클록 포트를 변환시키는 단계, (6) 표준화 포지티브/네거티브 단정을 포함하는 표준 데이터/제어 신호로 블록에 특정한 데이터/제어 신호를 변환하는 단계, (7) 모든 사이클, 큰-엔디안 또는 작은-엔디안(큰-엔디안은 데이터 유니트의 좌측 단부상에서 0 비트를 가지며, 작은-엔디안은 우측에 있다)에서 유효 입력을 예측하는 블록에 레지스터를 부가함으로써 표준 버스 인터페이스로 블록에 특정한 버스 인터페이스를 변환하는 단계를 포함한다.

<320> 더욱이, 칼라는 칼라 블록에 대한 부가 기능을 수행하는 소자(전술한 글루 로직)를 포함할 수 있다. 글루는 3가지 레벨, 즉 (1) 칼라에 전개된 글루, (2) 칩 레벨에서 결합된 글루, 및 (3) 칩 레벨에서 하나 이상의 미니 블록에 전개된 글루로 존재할 수 있다. 더 상세히, 글루 로직은 단순한 기능을 가진 번역자(예컨대, 각각의 비트라인을 따라 배치된 NAND 게이트)로부터 더 복합한 기능을 가진 소자(예컨대, 레지스터, 누산기 등)까지 임의의 기능을 가진 소자를 포함할 수 있다. 비록 글루 로직이 임의의 크기를 가질지라도, 만일 글루 크기가 블록에 대해 중요하게 되면, 프런트 엔드 어셈블리 및 칩 플래닝동안 만들어진 추정은 글루의 크기가 고려되지 않기 때문에 부정확하게 된다. 글루의 상대 크기를 블록에 강제로 맞출 필요가 있다.

<321> 칼라링 프로세스에 다음과 같은 가설이 사용될 수 있다.

<322> (1) 글루를 추가해야 하는지에 관한 결정이 칩 플래닝에서 만들어진다.

<323> (2) 글루 로직의 3가지 형태(칼라에 주입된 글루, 칩 레벨에서 결합된 글루 및 칩 레벨에서 미니 블록에 주입된 글루)중에서, 칼라링 프로세스는 칼라에 주입된 글루만을 처리한다.

<324> (3) 에스펙트 비 문제가 합성동안(블록 칼라링이 아님) 조절된다.

<325> (4) BBD에 대하여 칼라링된 블록의 출력은 레이아웃이다.

<326> 도36을 참조하면, 본 발명에 따라 칼라(602)와 블록(604) 사이의 논리 관계가 도시되는데, 이는 전술한 칼라 기능의 몇몇 예를 나타낸다. 도 36에는 칼라(602) 및 블록(604)사이의 논리도가 도시되어 있으며, 이는 본 발명에 따라 앞서 기술된 칼라의 전형적인 기능을 기술한다. 제 1 부분은 블록(604)의 경계 주위에서 특정 인터페이스에 접속가능한 소자를 포함한다. 제 2 부분은 표준에 따라 입력 및 출력 소자를 포함하며, 제 3 부분은 블록(604)으로부터의 출력을 표준으로 변환하기 위한 소자를 포함한다.

<327> 특히, 칼라(602)에서, 버스 인터페이스(606)는 두개의 단방향 버스(608, 610)를 양방향 버스(612)로 통합한다. 테스트 액세스 포트(614)는 블록(604)으로부터의 정보를 수집하고 블록(604)에 대한 테스트를 수행하기 위하여 입력(616)에 접속된다. 게이트(618)는 게이트(619)에 의해 수신된 입력 신호를 블록(604)에 접합한 포맷으로 전환하며, 게이트(620-624)는 클록 버퍼링을 수행한다.

<328> 도 37에는 칼라(702) 및 블록(704)간의 물리적인 관계가 도시되어 있으며, 이는 본 발명에 따라 앞서 기술된 칼라의 전형적인 기능을 기술한다. 도 37에서, 칼라(702) 및 블록(704)은 둘다 다중 금속층을 포함한다. 파워 표준은 금속층(3)(M3)상에 Vdd 전압을 인가하고 금속층(4)(M4)상에 GND를 인가하기 위하여 사용된다. 만일 블록(704)이 파워 표준에 따르지 않는다면, 칼라(702)는 파워 표준에 따르도록 파워를 변환시킨다. 영역(706)은 핀 공간/층 표준을 세팅한다. 만일 블록(704)이 핀 공간/층 표준을 따르지 않는다면, 칼라(702)는 핀 공간/층 표준을 따르도록 파워를 변환시킨다. 칼라(702)는 글루(708)를 경화된 상태로 포함한다.

<329> 도 39에는 본 발명의 칼라링 프로세스를 사용하지 않는 시스템 설계(800)가 도시되어 있다. 도 38에 도시된 바와같이, 시스템 설계(800)는 네개의 회로 블록(A,B,C 및 D)으로 구성된다. 블록에 접속된 각각의 화살표 라인은 블록에 대한 인터페이스의 설계에 대한 제한성을 나타낸다. 따라서, 시스템이 n 회로 블록(본 실시예에서 n=4)으로 구성된다면, 임의의 특정 블록에 대한 인터페이스는 n-1 제한 세트까지 만족될 필요가 있다. 따라서, 모든 블록에 대하여 만족될 필요가 있는 전체 제한수는 O(n2)이다.

<330> 도 40에는 본 발명의 칼라링 프로세스를 사용하는 시스템 설계(900)가 도시되어 있다. 시스템 설계(900)는 4개의 회로블록(A, B, C, D)으로 구성된다. 블록에 접속된 각각의 화살표 라인은 상기 블록에 대한 인터페이스의 설계에 대한 제한성을 나타낸다. 본 발명의 칼라링 프로세스를 사용하면, 각각의 블록은 칼라링 인터페이스에

의해 한정된 한 제한세트만을 만족할 필요가 있다. 따라서, 시스템이 n 회로블록(본 실시예에서 n=4)으로 구성된다면, 모든 블록에 대해 만족될 필요가 있는 전체 제한 수는 O(n)이다.

<331> 도 38에는 여기에 논의된 칼라링 및 다른 BBD 프로세스에 대한 단계를 수행하기 위한 컴퓨터 시스템(1000)이 본 발명에 따라 도시되어 있다. 컴퓨터 시스템(1000)은 시스템 버스(1001), 처리 유니트(1002), 메모리 장치(1004), 디스크 드라이브 인터페이스(1006), 하드 디스크(1008), 디스플레이 인터페이스(1010), 디스플레이 모니터(1012), 직렬 버스 인터페이스(1014), 마우스(1016) 및 키보드(1018)를 포함한다.

<332> 하드 디스크(1008)는 디스크 드라이브 인터페이스(1006)에 접속된다. 모니터 디스플레이(1012)는 디스플레이 인터페이스(1010)에 접속되며, 마우스(1016) 및 키보드(1018)는 직렬 버스 인터페이스(1014)에 접속된다. 시스템 버스(1001)에는 처리 유니트(1002), 메모리 장치(1004), 디스크 드라이브 인터페이스(1006) 및 디스플레이 인터페이스(1010)가 접속된다.

<333> 메모리 장치(1004)는 데이터 및 프로그램을 저장한다. 디스크 드라이브 인터페이스와 함께 동작할때, 하드 디스크(1008)는 데이터 및 프로그램을 저장한다. 그러나, 메모리 장치(1004)는 하드 디스크(1008)보다 액세스 속도가 빠른 반면에, 하드 디스크(1008)는 보통 메모리 장치(1004) 보다 큰 용량을 가진다.

<334> 디스플레이 인터페이스(1010)와 함께 동작할때, 디스플레이 모니터(1012)는 실행된 프로그램 및 사용자사이의 시각적 인터페이스를 제공하며, 프로그램에 의해 발생된 출력을 디스플레이한다. 직렬 버스 인터페이스(1014)와 함께 동작할때, 마우스(1016) 및 키보드(1018)는 컴퓨터 시스템(1000)에 입력을 제공한다.

<335> 하나 이상의 프로세서를 포함할 수 있는 프로세싱 유니트(1002)는 메모리 장치(1004) 및 하드 디스크(1008)에 저장된 프로그램을 실행함으로써 컴퓨터 시스템(1000)의 동작을 제어한다. 프로세싱 유니트는 메모리 장치(1004) 및 하드 디스크(1008)사이의 데이터 및 프로그램의 전송을 제어한다.

<336> 본 발명에 있어서, 여기서 논의된 단계를 실행하기 위한 프로그램은, 당업자에 의해 이해되는 바와같이, 메모리 장치(1004) 또는 하드 디스크(1008)에 저장될 수 있으며 또한 프로세싱 유니트(1002)에 의해 실행된다.

<337> 버스 식별 및 플래닝

<338> 본 발명의 방법은 프런트 엔드 허용(앞서 기술됨) 동안 기술된 바와같이 엔드 사용자 또는 설계 팀이 원하는 시스템의 전체 설계에 대한 성능 요건을 충족시키기 위하여 제공된다. 성능은 본 발명의 설계 방법에 대한 첫번째 고려대상인 반면에, 두번째 고려대상은 버스의 형태를 선택하는 동안 게이트 카운트를 감소시키는 것이다. 왜냐하면, 버스의 크기는 크고 단순한 버스가 작고 복잡한 버스보다 더 많은 로직을 소비하도록 이용가능한 버스 형태에 따라 변화할 수 있기 때문이다.

<339> 먼저 도 41을 보면, 본 발명의 방법을 포함하는 일련의 단계가 도시된다. 단계(4110)에서는, 고객의 초기 스펙의 트-엔드 허용가 완성된다. 이 단계는 상기 상세하게 설명되어 있다. 다음, 단계(4112)에서, 이하 설명되는 것처럼 예정된 버스 요구조건이 분석된다. 단계(4114)에서, 대기, 대역폭, 방향을 포함하고, 블록 각각에 대한 인터페이스에 존재하는 변수가 분석되면서, 단계(4116)에서 버스 분류 표준 라이브러리를 정하면서 버스 클러스터링이 계획된다. 다음, 단계(4118)에서, 새로운 버스 스펙이 개발되고 단계(4120)에서, 새로운 스펙이 검증되어, 컴플라이언스 위치 및 버스 모델 검증 부단계를 포함한다. 단계(4118, 4120)는 프리스테이징 단계(4122)를 차단하도록 레퍼런스를 수행하며, 아비터 및 브리지를 커버링하는 새로운 블록 스펙이 생성되고 칼라(collar)를 포함하는 새로운 스펙이 수정되고, 글루 스펙이 한정되고 테스트벤치가 생성된다.

<340> 버스 플래닝에 대한 논의에서 시작하여, 트 엔드 스펙을 상위 레벨 버스 스펙으로의 전이를 포함한다. 유효한 기술로서, 시스템 설계자는 고레벨 기능 모델 또는 시스템의 스펙에서 시작하여 설계를 한다. 시스템의 전문지식 및 유사한 시스템의 지식을 사용하여, 설계자는 설계를 위한 버스 구조의 고레벨 다이아그램을 구성한다. 일반적으로 설계자는 각각의 버스에 대한 트래픽의 복잡한 생각을 갖고, 얼마나 많은 버스 및 얼마나 복잡한 것이 요구되는지를 추정할 수 있다. 버스는 인터페이스 로직 및 설계 노력을 감소시키면서 요구되는 시스템 성능을 갖도록 설계된다. 설계자는 이러한 아키텍처를 사용하여 스펙에서 정의된 것처럼 설계가 동작하는지를 검증하도록 버스 기능 모델을 생성한다. 이러한 종래의 프로세스는 다양한 전문적 지식 및 설계자의 오래된 경험을 필요로하기 때문에 제어가 어렵다. 본 명세서에서의 태스크(task)는 칩 설계에서 버스 구조를 한정하는 프로세스로 형식적 구조를 사용한다. 그러나, 이러한 태스크는 버스 및 최고의 결과를 달성하기 위해서 버스 및 시스템 개발 기술에 관련한 평균이상의 기술이 요구된다.

<341> 버스 프로토콜

<342> 버스는 설계에서 회로 블록간의 바람직한 통신 매체를 제공한다. 이러한 가장 간단한 형태의 버스는 극소의 로직을 요구하나 많은 배선이 요구되는 포인트-투-포인트 접속(point-to-point connection)의 연결일 수 있다. 간단한 버스는 각각의 클록 주기마다 블록들 사이에 데이터를 전송한다. 일부 블록은 이러한 정보 전송의 형태를 요구하나, 시스템에서의 대부분의 블록은 단지 가끔씩 다른 블록으로부터의 정보를 요구한다. 칩 핀은 대형 시스템 설계시에 고가이기 때문에, 버스는 요구되는 칩 핀의 수를 줄이고 성능의 최소 손실로 시스템에서 많은 상이한 블록 사이에 주기적 통신을 허용하도록 사용된다. 이를 위해, 설계자는, 블록은 버스 와이어를 사용하는지, 전송되는 데이터 블록이 무엇인지, 전송자는 데이터를 언제 전송하는지, 그리고 수신자가 데이터를 수신했는지 여부와 같은, 데이터 전송 스케줄의 트랙을 유지하기 위해 블록 각각에 로직을 추가해야 한다. 이러한 문제는 버스상에서의 신호를 제어함으로써 그리고 블록들 사이의 통신을 제어하기 위한 정치를 설정에 의해 처리된다(버스 프로토콜).

<343> 버스 프로토콜의 2가지 예로는 주변 버스 및 패킷 네트워크이다. 간단한 주변 버스 프로토콜에서, 하나의 장치는 버스를 제어한다. 모든 정보 및 데이터는 상기 장치를 흐르며, 데이터를 수신 또는 송신하는 시간을 결정한다. 주변 버스 프로세싱은 상대적으로 극소의 로직을 요구하는 반면, 효과적인 버스 와이어를 사용하지 않고, 플렉시블(flexible)하지 않다. 패킷 네트워크 프로토콜은 상대적으로 복잡하다. 데이터를 전송하는 블록 및 데이터를 수신하는 블록에 대한 모든 정보는 패킷의 데이터로 저장된다. 패킷 프로토콜은 임의의 시간에 임의의 다른 블록으로 데이터를 전송하게 한다. 이러한 프로토콜은 매우 플렉시블하며 버스 와이어를 효과적으로 사용하나, 각각의 버스는 패킷으로 전송되고 이를 수신하는 패킷을 해독해야할 때를 알도록 다수의 로직이 요구된다. 다른 버스 프로토콜은 플렉시빌리티(flexibility), 이용도, 및 대기(버스상에 있는 하나의 블록에서 또다른 블록으로 정보를 전송하는 초기 지연시간) 레벨이 상이하다. 상이한 버스 타입 및 이러한 프로토콜에 대한 분류를 도 59에 도시한다.

<344> 본 발명의 BBD 버스 설계 방법은 한정된 버스 타입을 사용한다. 설계자는 만약 이들이 권한이 부여된(authored) 블록의 일부가 아니라면 스크래치(scratch)로부터 버스를 개발하는 것을 기대하지 못했다. 또한, 설계자는 복잡한 버스를 생성하기 보다는 공지된 버스 타입에 존재하는 블록을 논리적으로 연결하는 것이 바람직하다. 본 발명의 BBD 방법은 블록들 간의 신호 연결로서 버스를 처리한다. 버스를 위한 로직은 설계시에 블록들 사이는, 버스가 글루 로직 섹션에서 상기 설명된 것처럼, 버스 외측과 통신되도록 하는 글루 로직인 것처럼, 구성된다.

<345> 모든 논리적인 상호연결은 간단한 또는 복잡한 버스로서 처리된다. 상호연결의 간단한 형태는 버스 연결 룰에 의해 정의되나, 복잡한 버스에 대한 특정 프로토콜은 바람직하게 정의되지 않는다. 본 발명의 BBD 방법은 바람직하게 버스를 지지한다, 즉 계층(hierarchy)을 갖고, 물리적 계층의 1개 레벨 내에 완전하게 포함되며, 블록과 외부에 와이어가 있고, 논리적 계층의 1개 레벨 내에 완전히 포함되며, 설명에 따른 VSI의 온-칩 버스(OCB; on-chip Bus)를 따르며, 트랜잭션(transaction) 벡터를 따라 변형된다. 또한, BBD에 대한 많은 아웃-오브-스코프(out-of-scope)의 조건은 본 발명에 따른 SOC 방법으로 바람직하게 지지된다.

<346> 버스는 블록내에 완전히 포함되거나 또는 상부 계층 레벨에서 상호연결됨으로써 한정되는 것이 바람직하다. 상부 레벨에서 한정된 버스는 버스 성분이 블록들 사이 및 블록내에 구성되도록 하는 레벨에서 생성된다.

<347> BBD 칩에 대한 버스를 정의하기 위해, 이하의 단계가 실행되며, 각각은 보다 상세히 다음과 같다;

<348> 버스 요구조건 추출

<349> 클러스터링에 기초한 버스 정의

<350> 버스 선택

<351> 버스 설계 설명

<352> 버스 분류 참조

<353> 버스 선택 검증

<354> 블록 설계 가정

<355> BBD 방법에서, 설계자는 버스 설계를 설명할 때, 블록 구조와 연결하여야 한다. 이러한 태스크는 만약 펌(firm) 또는 하드 블록이 특정 버스 인터페이스를 포함하는 경우, 칼라(collar)를 참조로 상기 설명된 것처럼

인터페이스는 소프트하다고 가정한다. 또한 모든 타입의 블록은 버스 인터페이스 로직 및 블록의 실제 기능 사이의 단순화된 인터페이스를 포함한다고 가정한다. 이는 주변 블록에 대해 무모한 것이 아니며, 많은 제 3자의 블록 제공자가 이들 자신의 간단한 인터페이스를 생성하여 사용자가 버스 인터페이스 로직을 부가할 수 있기 때문이다. 다수의 디자인으로 형성된 블록은 개별 내부 기능 및 버스 인터페이스 로직을 갖는다. 내부 인터페이스는 이러한 블록들이 상이한 버스로 재사용되게 한다. 하드 블록이 그의 내부 기능과 별개일 수 없는 특정 버스 인터페이스 로직을 갖는 경우, 보다 복잡한 버스 프로토콜 변이가 블록에 추가된다. 이러한 경우, 형성되는 버스 인터페이스 로직은 블록 설계 동안 생성된 소프트 칼라(collar)의 일부가 된다.

버스 요구조건 추출

<357> 프런트 엔드 허용 태스크는 버스망, 신호망, 및 블록 각각의 핀을 포함한다. 신호망의 4개의 카테고리는, 1) 프로세서와 같은 일정 블록에서 요구되는 PCI 또는 AMBA 버스와 같은 버스를 포함하는 망 및 블록 핀인 예정된 버스 신호; 2) 판독 및 기록 신호와 같은 버스인 망 및 블록 핀인 버스 신호; 3) 와이어 또는 버스일 수 있는 망 또는 블록 핀인 가능 버스 신호; 및 4) 버스에 의해 처리되지 않는 와이어 망인 신호이다.

<358> 설계자는 신호 타입이 결정되면, 프런트 엔드 허용 태스크로부터 수신된 데이터를 이들 4개 타입의 신호망에 따라 조직화한다. 타입 1 및 2 망(net)에 대해, 버스를 생성하는데 필요한 데이터는 고객 또는 다른 이용가능한 것에 의해 제공될 수 있다. 요구되는 데이터는 참조로 본 명세서에서 사용되는, VSI 온칩 버스(OCB) 속성(Attribute) 스펙 OCB1 1.0에 또한 정의된다.

<359> 추가로, 설계시에 설명 또는 사용될 수 있는 각각의 버스는; 버스를 생성하기에 충분한 완변한 사용자 가이드; 버스에 대한 물리적 요구사항을 한정하는 실시 가이드; 버스를 테스트하고 검증하는 완벽한 세트의 시뮬레이션 공구; 및 기술 속성 리스트 및 리스트와 버스의 비교 방법을 갖는다. 또한, VSI 온칩 버스 속성 설명서에 따라 버스를 형성하기 위해, 공급자는 이하 설명되는 문헌 및 모델을 제공해야 한다.

<360> 사용자 가이드 및 시뮬레이션 공구

<361> 사용자 가이드 및 스뮬레이션 공구는 버스 성분을 조립하고 테스트하는 버스 설계에 사용된다. 시뮬레이션 공구 세트는 버스 마스터; 버스 슬레이브(slave); 버스 지지 기능(아비터, 어드레스 디코더); 및 스탠다드 버스 브리지의 부재에 대한 행동 Verilog 및/또는 VHDL내에 기록된 모듈을 포함한다. 이들은 버스 검증과 관련된 섹션에 기술된 것처럼 버스를 검증하는데 사용된다.

<362> 실시 가이드

<363> 실시 가이드는 블록 설계, 칩 어셈블리 및 버스의 속성을 기술하기 위한 칩 설계시에 수반되는 태스크에 사용된다. 이하 정보는 블록 설명의 일부로서 블록 설계를 거친다; 특정 셀 요구; 셀의 물리적 특성; 버스 복합 또는 옵션 조정; 메모리 맵; 파워 분포; 및 타이밍 지침. 타이밍 및 최대 로딩(loading) 지침서는 칩 설계시에 차후 단계에서 사용될 수 있다. 타이밍 지침서, 최대 로딩, 및 버스 레이아웃 또는 와이어링의 제한은 버스 실시에서 사용을 위한 칩 어셈블리 태스크를 통과한다.

<364> 기술적 속성(attribute) 리스트

<365> 기술적 속성은 버스 분류 라이브러리에서 버스 속성으로서 유지될 수 있는 형태로 변환된다. 따라서 버스 분류 표준 및 버스 타입 테이블은 버스 타입을 선택하기 위해 설계자가 사용한다. 버스 신호를 예비 지정하기 위해서, 설계자는 요구되는 연결이 최대 로딩 및 타이밍 지침서에 부합하고, 버스 레이아웃 및 와이어링 제한이 칩 어셈블리 동안 부합할 수 있도록 검사한다. 만약 그렇지 않다면, 설계는 고객에 의해 변형되어 프런트 엔드 허용 태스크로 다시 보내진다.

<366> 클러스터링에 기초한 버스 정의

<367> 클러스터링에 기초한 버스를 정의하기 위해, 설계자는 상호연결 대역폭 및 프런트 엔드 허용에서 수신된 잠복시간을 사용한다. 이 단계에서, 각각의 클러스터 및 클러스터내의 블록에 대해, 잠복 시간, 대역폭, 현재의 버스 인터페이스 타입, 및 데이터 방향 흐름을 결정한다. 이러한 정보는 다음 단계로 통과되어 버스를 선택한다.

<368> 버스 계층은 최고 대역폭 및 최저 대기 버스 상호연결의 클러스터링에 의해 정의된다. 지점간 망일 수 있는 가능한 버스 신호는 여기서 제거될 수 있고 차후 버스가 분석되고 설계되며, 이는 이러한 신호가 절차를 위한 칩 어셈블리 태스크로 직접 제공되기 때문이다.

<369> 통신 매니저 행동 모델 생성

<370> 검증된 칩의 행동 모델은 블록들간의 상호연결의 행동 모델 및 요약 모델을 포함한다. 전형적으로, 이러한 상호연결은 테스트 벤치 및 블록 사이에 데이터를 전송하는 소프트웨어 메카니즘이다. 바람직하게, 가능한 스케쥴러의 통신 메니저의 형태는 모든 블록이 연결되도록 하는 것이다. 다른 방법(extreme)에서, 상호연결은 또한 행동 모델에서 지점간 인터페이스와 직접 연결될 수 있다.

<371> 이하 설명되는 것처럼 통신 매니저, 스케줄러는 시뮬레이션 모듈의 상위 레벨에서 사용된다. 이러한 스케줄러에 대한 의사코드(Pseudoode)는 다음과 같다:

<372> 큐(queue)가 빈 Do가 아닌 경우;

<373> 큐로부터 다음 트랜잭션 획득;

<374> 트랜잭션으로부터 타겟 블록 획득;

<375> 타겟 블록(트랜잭션) 호출;

<376> 엔드;

<377> 이러한 의사코드 예에서, 각각의 블록은 다음과 같다:

<378> 타겟 블록(트랜잭션);

<379> 블록의 기능 수행;

<380> 큐에 새로운 트랜잭션 추가;

<381> 엔드;

<382> 이러한 코드 레벨에서, 타이밍 또는 버스 사이즈 어느것도 정의되지 않는다. 모든 통신은 트랜잭션에서 수행되거나 또는 임의의 사이즈의 정보 패킷을 전송함으로써 행해진다. 트랜잭션은 가능 버스 신호 및 비-버스 와이어를 포함하여 블록 사이의 모든 통신은 스케줄러를 통하게 된다.

<383> 다른 방안으로, 설계자는 비동기식으로 비-버스 신호를 전송하고 판독하도록 블록 의사코드를 변조시킬 수 있다. 이러한 경우에, 각각의 블록은 다음과 같다:

<384> 타겟 블록(트랜잭션);

<385> 상위 레벨로부터 비-버스 신호값 획득;

<386> 블록의 기능 수행;

<387> 큐에 새로운 트랜잭션 추가;

<388> 상위 레벨로 새로운 비-버스 신호값 적용;

<389> 엔드

<390> 주목할 것은, 간략화를 위해, 이들 실시예는 비-버스 신호를 포함하지 않는다는 것이다. 그러나, 설계자는 실시예들이 비-버스 신호를 수반하도록 유사하게 조절할 수 있다.

<391> 패턴 세트는 한 개의 블록이 다른 블록과 통신되도록 하여 테스트 벤치(bench)에서의 벡터 집합이다. 테스트 벤치는 전체 칩의 기능이 수행되도록 충분한 패턴 세트를 포함해야 한다. 설계자는 코오스(coarse) 레벨에서의 패턴 세트 각각에 타겟 성능 레벨을 할당해야 한다. 예를 들어, 한 개 패턴 세트에서 MPEG 디코더를 위한 프레임 데이터가 있는 경우, 설계자는 얼마나 길게 타겟 하드웨어가 그 세트에서 프레임의 프로세스를 처리하는지를 결정할 수 있다. 만약 설계자가 초당 약 30 프레임인 출력 비율을 알고 있는 경우, 프로세싱 비율은 그 수를 초과하게된다. 이러한 성능의 타겟은 요구되는 버스 대역폭을 한정하기 위해 상기 프로세스의 차후 단계에 사용된다.

<392> 칩에 대해 선택된 블록은 사이클에 어느정도 근접한 성능 스펙을 갖는다. 행동 모듈이 이러한 설명을 갖지 않

지 않는 경우, 이들은 상기 단계에서 모듈로 통합된다.

<393> 도 42는 행동 모델의 상호연결부의 내부 구조를 도시하고 있다. 첫째, 테스트 벤치와 조건들이 수신된다. 다음, 예비 스케쥴러가 생성된다. 상호연결 관리자/스케쥴러(4210)가 그 설계의 블록들 사이에 정보를 전송하고 그들의 실행을 스케쥴한다. 다음, 상호연결(4210)이 수정되며, 수정된 상호연결 관리자(4212)는 통계 모임 및 모델이 사이클 근사 동작에 맞추어짐에 따라 추가된 대기 행렬을 포함한다. 마지막으로, 테스팅과 설계 반복을 위해 테스트 벤치가 다시 이용된다. 이들 수정예들의 세부사항은 이하의 섹션에서 설명된다.

<394> 대기 보상을 위한 모델 수정

<395> 어떤 설계들을 어떠한 특별한 대기 조건도 가지지 않는다. 허브나 스위치와 같은 다른 설계들은 데이터 대기(데이터의 첫 번째 단위가 송신자에서 수신자에게로 가는데 걸리는 시간)에 민감하다. 대부분의 네트워크 장치, 특히 비동기 전달 모드(ATM) 장치는 정보 전달을 위한 특별한 대기 조건을 가지며, 이것은 네트워크 내의 구성요소들과 버스에 대한 엄격한 대기 조건으로 변환된다. 설계자가 그 설계에 대한 대기 조건을 알기만 하면 상호연결 모델을 다음과 같이 조정하게 된다. 첫째, 각 패턴에 대해 두 개의 행렬이 생성되며, 각기 1) 블록들 사이에서 전달될 데이터의 양과 2) 실행되는 트랜잭션의 수를 나타낸다. 둘째, 각 패턴 세트에 대해 사이클 계수 근사를 특정하는 행렬이 생성된다. 이 두 단계는 대기 조건이 없는 설계에는 불필요하다.

<396> 데이터 전달 행렬

<397> 데이터 전달 행렬을 생성하기 위해, 설계자는 먼저 하나의 블록에서 다른 블록으로 전달되는 데이터의 양을 통신 관리자 모델에 추가한다. 다음, 스프레드시트 툴을 사용하여, 설계자가 각 패턴 세트에 대해 테이블에서 이 데이터를 누적한다.

<398> 예를 들어, 세 개의 블록과 하나의 테스트 벤치를 가진 칩에 대한 테이블은 테이블의 각 란에서 모든 데이터의 합이 전달된 4*4 테이블이다. 대각선은 모두 제로이다. 좀더 실제적인 모델은 버스들이 칩으로 들어가거나 칩으로부터 나온다는 것을 고려해야 하며, 따라서 테스트 벤치는 각 축에 적어도 하나의 란을 더 가지게 될 것이다.

<399> 데이터 전달 행렬의 예는 도 43의 테이블에 도시되어 있다. 이 행렬에 대한 설계는 세 개의 블록과 테스트 벤치를 위한 세 개의 포트를 가진다. 즉, 외부 메모리에 대한 인터페이스, PCI 인터페이스, 및 병렬 I/O 인터페이스이다. 테이블에 도시된 바와 같이, 블록 1에서 블록 2로 전달되는 데이터는 10,000바이트이며, 블록 2에서 블록 1로 전달되는 데이터는 8,000바이트이다.

<400> 따라서, 데이터 전달 행렬을 생성하는 첫 번째 단계는 도 44에 도시된 바와 같이 모든 트랜잭션의 계수가 예시적 패턴 세트 X에 대한 트랜잭션을 보여주는 테이블을 생성하는 것이다.

<401> 도 43과 도 44에 도시된 테이블들을 생성하기 위해, 설계자는 다음과 같이 스케쥴러 의사코드를 수정할 수 있다.

<402> 큐가 비어있는 동안 Do;

<403> 큐로부터 다음 트랜잭션을 획득;

<404> 트랜잭션으로부터 센더 블록을 획득;

<405> 트랜잭션으로부터 타겟 블록을 획득;

<406> 트랜잭션 바이트 계수를 획득;

<407> 트랜잭션 행렬(센더, 타겟)

<408> = 트랜잭션 행렬(센더, 타겟) + 1;

<409> 트랜잭션 행렬(센더, 타겟)

<410> = 트랜잭션 행렬(센더, 타겟) + 트랜잭션 바이트 계수;

<411> 타겟 블록(트랜잭션)을 호출;

<412> End;

<413> 비-버스(non-bus) 블록대블록(block-to-block) 전선은 얼마간의 지연(통상, 적어도 하나의 클록

사이클)을 가지며, 이들을 바람직하게는 타이밍 큐에서 버스 트랜잭션 외에 별개의 트랜잭션으로 추가된다.

<414> 대기 행렬

<415> 각 블록에 대한 클록 사이클 시간이 프런트-엔트 허용(front-end acceptance)에서 이미 정의되었기 때문에, 설계자는 원 성능(raw performance)을 사이클 계수로 다음과 같이 변환할 수 있다.

<416> 1. 설계서에 정의된 사이클 근사 동작을 반영하기 위해, 설계자는 각 블록에 대해 추정된 클록 사이클을 현존 행동 모델에 부가한다. 이 단계는 확인 후에 블록을 블록 설계 태스크로 보내기 전해 수행되는 것이 바람직하다.

<417> 2. 설계자는 블록들을 다시 칩 모델로 통합한다. 다음, 칩 모델은 상호연결에 어떤 시간도 정의되지 않은 사이클 근사 블록들을 가지게 된다.

<418> 3. 설계자는 스프레드시트를 사용하여 도 43과 도 44에 도시된 것과 유사한 테이블을 작성한다. 전달되는 바이트의 수 대신에, 설계자는 데이터가 이용가능한 시간부터 데이터가 다음 블록 또는 테스트 벤치에 도달할 때까지(대기동안) 각 전달이 취하는 사이클의 수를 정한다.

<419> 4. 설계자는, 새로운 테이블에 도시된 성능 값들을 사용하기 위해 상호연결 모델을 수정한다.

<420> 도 45는 예시적 대기 행렬을 도시하고 있다. 이러한 수정의 의사코드 예는 다음과 같다.

<421> While 큐가 비어있는 동안 Do;

<422> 큐로부터 다음 트랜잭션을 획득;

<423> 트랜잭션으로부터 시간을 획득;

<424> 트랜잭션으로부터 타겟 블록을 획득;

<425> 타겟 블록(트랜잭션, 시간)을 호출;

<426> End;

<427> 각 블록은 다음을 수행한다.

<428> 타겟 블록(트랜잭션, 시간);

<429> Do 블록의 함수(function);

<430> 트랜잭션 시간을 시간 + 지연 + 대기(상기 블록, 타겟)로 설정;

<431> 새로운 트랜잭션을 큐로 소트(sort);

<432> End;

<433> 도 44에서 "0"이란 란은 어떤 데이터도 전달되지 않으며 그 자체로는 대기 행렬에 적용될 수 없다는 것을 나타낸다.

<434> 5. 설계자는 설계 데이터 흐름에 대한 지식을 이용하여 추정된 상호연결 사이클 계수 지연을 가진 칩 대기 조건들을 포함하도록 테스트 벤치를 수정한다.

<435> 6. 설계자는 설계가 사이클 조건을 만족하는지 확인하기 위해 시뮬레이트한다.

<436> 7. 설계자는 대기 행렬을 수정하고, 칩의 사이클 조건이 만족될 때까지 확인 프로세스를 반복한다.

<437> 각 타입의 버스 전달에 이용할 수 있는 최대 사이클 계수를 가진 테이블을 생성하기 위해, 설계자는 먼저 큰 사이클 계수를 사용해야하고 세부사항이 만족될 때까지 계수를 감소시켜야 한다. 이는 엄격한 대기 조건일수록 더 게이트 집중적인(gate-intensive) 구성(scheme)으로 변환되기 때문이다.

<438> 클러스터 측정치 결정

<439> 다음, 데이터의 자연적 클러스터링을 반영하기 위해, 설계자는 센터 대각선에 가장 가까운 최대 계수를 이동시킴에 의해 데이터 전달 행렬을 재구성한다. 이 프로세스를 수행하는데에는 여러 가지 길이 있는데, 바람직한 방법은 피보팅(pivoting)이다. 피보팅의 목적은 블록들을 최고의 전달율로 클러스터하여 필요한 핀의 수를 최소화시키는 것이다. 설계자는 계산의 자동화를 위해 스프레드시트를 작성할 수도 있다.

<440> 클러스터링이 얼마나 효율적인지를 측정하기 위해, 데이터 전달 행렬의 각 사이트는 정확하게 가중되어야 한다. 이 예는 도 46에 도시된 거리 행렬을 사용하여 사이트들을 가중한다. 도 46의 테이블에서, 각 셀은 대각선과 셀간의 거리의 제곱을 포함한다. 데이터 전달 행렬을 가중하는 다른 방법도 이용될 수 있으나, 거리 제곱이 바람직하다. 이는 배치 알고리즘에서 알 수 있는 바와 같이, 고차 측정에서는 제한되는 시스템의 요소들의 이동성을 허용하면서도 신속하게 수렴되기 때문이다.

<441> 다음, 설계자는 데이터 전달 행렬의 각 셀을 거리 행렬의 대응 셀과 곱하고 모든 셀에 대한 모든 값들을 더한다. 그 결과치가 클러스터 측정치이다. 도 43의 테이블의 행렬의 클러스터 측정치는 428,200이다. 클러스터 측정치가 낮을수록 버스 클러스터링이 더 효율적이다.

<442> 피봇 블록들

<443> 더 낮은 클러스터 측정치를 얻기 위해, 설계자는 행들을 하나씩 교환(swapping)하고 각 교환 후에 클러스터 측정치를 재계산함에 의해 데이터 전달 행렬을 피봇하여 클러스터 측정치가 개선되는지를 확인해야 한다. 사이트들이 소트되는 목록의 요소들인 경우에는 다음의 의사코드에 도시된 바와 같이 소트를 수행함에 의해 행들을 교환할 수 있다.

<444> 행렬의 현재 클러스터 측정치를 획득;

<445> Do for 현재 사이트 = 행렬의 사이트 1 to n-1;

<446> Do for 다음 사이트 = 행렬의 현재 사이트 + 1 to n;

<447> 다음 사이트를 현재 사이트로 교환;

<448> 행렬의 다음 클러스터 측정치를 획득;

<449> If 다음 클러스터 측정치 > 현재 클러스터 측정치

<450> Then 원위치까지 다음 사이트를 현재 사이트로 교환.

<451> Else 현재 클러스터 측정치 = 다음 클러스터 측정치;

<452> End

<453> End;

<454> 비록 상호연결이 연결 대신 대역폭이지만, 이 소트는 이차(quadratic) 배치 알고리즘과 유사하다. 설계자는 이 방법 대신에 이것과 유사한 결과를 제공하는 다른 방법을 사용할 수도 있다.

<455> 상기 설명된 피보팅은 예를 들어 117,000의 개선된 클러스터 측정치를 갖는 도 47의 행렬을 생성한다. 이 이상적 예에서 컴포넌트들이 정보를 생성하지 않는다는 것에 유의해야 한다.

<456> 컴포넌트는 판독한 것을 기록하며, 컬럼과 로우 합계가 블록(3)와 PIO를 제외하고 매칭된다. 이것은 이 기술분야에 사용되기 위한 경우는 아닐 것이다.

<457> 다음, 설계자는 버스 클러스터를 정하기 위해 도47에 도시된 바와 같은 표를 이용할 수 있다. 이러한 실시예는 블록(1), 블록(2), PCI 및 메모리 사이의 높은 전송률 데이터 전송을 나타낸다. 따라서, 이들 컴포넌트는 고속 버스이어야 한다. 블록(3)과 PIO 사이에 낮은 데이터 전송률이 존재하기 때문에, 이들 설계 엘리먼트는 저속 버스에 기반할 수 있다.

<458> PIO는 출력만하는 것이나, 모든 다른 컴포넌트는 양방향이다. 클러스터 내부 및 외부 컴포넌트가 통신하여야 하기 때문에, 도48에 도시된 바와 같이 설계자는 두개의 버스 사이에 브리지를 형성하여야 한다.

<459> 클러스터링 기반 버스 정의

<460> 바람직하게는, 초기 클러스터링은 모든 소정의 버스 신호 네트(net)를 포함한다. 설계자는 내재적인 내부 서브클러스터를 표시하기 위해 클러스터 내에서 피봇(pivot)할 수 있으나, 하나 이상의 버스 타입이 이들 신호에 대해 정의되지 않는 경우에 이들은 다음 태스크(task)에서 하나의 클러스터로 취급되어야 한다.

<461> 프로세서의 시스템과 주변 버스들이 정의되는 경우에, 클러스터링 정보에 따라 클러스터가 시스템 버스와 주변

버스 또는 버스들로 분할된다. 예를들어 도47의 표의 버스 매트릭스(matrix)가 소정의 버스 신호 네트로 이루어지는 경우에, 초기 클러스터링은 전체 매트릭스를 포함한다. 하나 이상의 버스가 정의되는 경우에, 고속 버스 상에 실릴 필요가 있는 블록들은 하나의 버스를 형성하고 그 나머지는 다른 버스를 형성한다. 다음, 이들 분할(partition)이 다음 태스크로 통과된다.

<462> 소정의 버스 접속이 존재하지 않는 경우에, 버스들은 클러스터 정보에 따른 방식으로 정의된다. 피봇된 매트릭스는 다른 인접 블록들과 비교하여 그들 간의 대개 상대적으로 높은 통신 레벨을 갖는 인접 블록의 그룹을 갖는다. 도49의 표는 이전에 피봇된 매트릭스와 유사한, 이러한 종류의 클러스터링을 나타낸다. 도49는 클러스터링 프로세스를 보다 명확히 하기 위해서 이전에 도시된 것들과 다른 예들에 기초한 것이다. "##"가 매우 큰 수를 나타낸다는 점에 주의하여야 한다.

<463> 이 실시예에서, 블록 A, B 및 C는 하나의 독립적인 버스 클러스터를 형성하며, 이는 세개의 블록들 사이에 고속의 통신이 존재하고 이들 블록들과 블록 D 내지 H 간에 통신이 존재하지 않기 때문이다. 블록 D, E 및 F는 다른 클러스터를 형성하며, 이는 이들 세 개 모두 사이에 고속 통신이 존재하기 때문이다. 또한, 블록 D, E, 및 F는, D 및 E 사이의 점에 대한 버스와 E 및 F 사이의 또다른 점에 대한 버스의 두개의 분리된 버스를 형성할 수 있다. 블록 G 및 H는 세번째 클러스터를 형성한다. EF쌍과 GH쌍 사이에 저대역폭 접속이 존재한다. 데이터 전송량에 따라, E, F, G, 및 H는 낮은 레벨의 통신을 위해 이들간에 양방향 브리지를 갖는 하나의 버스 또는 두개의 분리된 EF 및 GH 버스 상에 위치할 수 있다.

<464> 서로 다른 클러스터링 선택들의 갯수를 선택하기 위해서는, 다음의 가이드라인을 따른다.

<465> 1. 가능한 클러스터를 결정하기 위해서, 비교적 낮은 통신 영역으로부터의 높은 통신 영역으로의 절단 점인 블록들간의 절단 점을 식별한다. 도49의 매트릭스의 C 및 D 사이의 절단은 도50에 도시된 다이어그램을 만들어낸다. ABC 및 DEFGH 그룹 사이의 통신의 양을 결정하기 위해서, 좌하 및 우상 그룹의 셀들이 합산된다. 이 실시예에서와 같이 이 합이 0이 되는 경우에, 두 그룹은 이들 간에 통신을 갖지 않는다. 이들 그룹은 완전히 분리된 버스를 형성한다. 절단을 통과하는 통신이 0이 되는 경우에 피봇된 매트릭스를 절단한다.

<466> 2. 각 식별된 그룹 내에서, 특별한 절단을 찾는다. 그룹들 간의 통신이 각 그룹 내에서 더 작아야 한다. 도50에서, 하나의 절단이 D-H 그룹에 존재하고 도51에서는 A-C 그룹 사이에는 절단이 존재하지 않는다. GH 그룹간의 데이터 전송률이 22이나, 다른 그룹들 내의 데이터 전송률은 매우 큰 수(##)를 갖는다. 이들 클러스터는 이들 간에 브리지를 갖는 두개의 버스를 형성할 수 있다.

<467> 3. 클러스터들 사이 또는 클러스터들 내의 통신이 모든 블록을 포함하지 않는 경우에, 클러스터링을 최적화하는 것이 요구된다. 대기 매트릭스가 소정의 블록들 사이의 통신에 대해 매우 상이한 요구를 가질 경우 최적화는 매우 중요하다. 예를 들어, 도51은 GH 클러스터가 DE와 통신하지 않음을 도시한다. DE와 EF는 통신하지만, D와 F는 통신하지 않는다. DE에 대한 대기 매트릭스 요구가 매우 타이트(tight)한 경우, 설계자는 DE 통신을 나머지 버스로부터 분리하여야 한다. 도 52로부터, 결과 매트릭스를 찾을 수 있다. 이 예에서는 E가 E'로 분리되어 두개의 서로 다른 블록이 된 것으로 보이며, 이는 분리된 인터페이스가 두개의 버스에 대한 E 상에 형성될 것이기 때문이다. 블록이 두개 이상의 버스 인터페이스를 갖는 경우에, 이 기술은 분리된 인터페이스를 효과적을 이용하는데 사용될 수 있다.

<468> 이 기술이 도 43의 원 실시예에 이용되는 경우에, 클러스터들 사이에 브리지를 갖는 두개의 버스로 이루어진 도53에 도시된 클러스터들이 형성된다. 하나의 버스는 다른 전송이 매우 작은 상태에서 상당한 양의 데이터를 전송한다. 블록(3)과 PIO 상의 다른 절단은 클러스터들 사이의 보다 낮은 통신을 야기한다. 그러나, 클러스터에 단지 하나의 블록을 남기기 때문에 이것은 중요한 것이 아니며, 따라서 이것은 가능하지 않다.

<469> 4. 모든 절단이 형성되는 경우에, 결과 클러스터 정보가 다음 태스크 상으로 전송된다.

<470> 이 클러스터링 기술은 칩에 대한 버스 구조를 형성하기 위해서 시스템 지식을 요구한다. 설계자는 데이터 타이밍과 현재 블록 버스 인터페이스, 추가 프로세서 요구, 및 버스 상의 마스터의 갯수와 같은 구현 상세 사항을 고려하여야 한다. 이들 인자들은 이러한 클러스터링 방법을 사용하여 얻어지는 구조로부터 벗어나는 것이 보다 향상된 성능 또는 순수하게 절차를 따름에 의해 얻어지는 것보다 더 낮은 게이트 카운트를 갖는 버스 구조를 형성한다. 이러한 경우에, 설계자는 클러스터링 결과를 수정하기 위해서 이 태스크를 반복하기를 원할 것이다.

<471> 버스 선택

<472> 일단 설계자가 클러스터링 방법을 이용하여 버스를 정의하면, 버스 타입과 성능 계층구조(hierachy)가 선택되어

야 한다. 버스 계층구조는 최고 성능 버스로부터 최저 성능 버스로 접속된 버스들의 순서이다. 예를들어, 설계가 고속 시스템 버스와 두개의 저속 주변 버스를 포함하는 경우에, 계층구조는 시스템 버스로부터 두 개의 주변 버스가 된다.

<473> 바람직하게는 버스 분류(taxonomy) 참조 라이브러리로부터 버스 속성(attribute) 및 크기가 각 버스에 대한 버스 타입을 정의하는데 이용된다. 라이브러리는 각 이용가능한 버스 타입에 대한 버스 속성 세트의 리스트를 갖는다. 적절한 버스를 선택하기 위해서, 설계자는 현재 버스 인터페이스에 대한 클러스터의 각 블록을 분석한다. 거의 또는 완전히 존재하지 않는 경우에, 가장 유사한 속성을 갖는 버스 분류 참조 내의 버스 타입이 선택된다. 이러한 선택 프로세스의 결과는 버스 설계를 구체화하는, 정의된 버스 세트와 다음 태스크에 사용되는 계층구조이다.

<474> 버스는 버스 분류 참조 라이브러리의 파라미터와 설계의 블록 인터페이스를 검사하여 다음과 같이 선택되어야 한다.

<475> 1. 클러스터의 대역폭과 지연 요구를 만족시키지 않는 버스를 제거한다.

<476> 2. 버스가 이미 정의된 경우는, 그 버스를 이용하며, 그렇지않은 경우는 이용하지 않는다.

<477> 3. 프로세서가 존재하는 경우는, 이미 접속된 시스템 버스를 이용하며, 그렇지 않은 경우는 이용하지 않는다.

<478> 4. 대부분의 블록이 이미 접속되어 있는 버스를 선택한다.

<479> 5. 접속된 대부분의 블록의 엔디안 형식(endian-ness)을 다룰수 있는 버스를 이용한다. (큰 끝 형식(big-endian)은 데이터 유닛의 왼편 끝에 0 비트를 가지며, 작은 끝 형식(little-endian)은 오른편 끝에 0 비트를 갖는다.)

<480> 6. 버스 상에 로딩(loading)하는 것이 과도한 경우는, 다중 버스를 이용한다.

<481> 7. 주변 버스 또는 버스들 상에서 낮은 대역폭 디바이스를 분리한다.

<482> 8. 선택된 시스템 버스에 현재 브리지를 갖는 주변 버스를 이용한다.

<483> 9. 선택 프로세스가 완결된 후에 하나 이상의 선택이 남아 있는 경우에는, 이것이 최적 툴과 모델 지원(support)을 갖는 것이므로, OCB 속성 리스트를 가장 잘 만족시키는 버스 타입을 선택한다.

<484> 버스 크기 계산

<485> 버스 대기시간 테이블은 이 단계에서 시작 포인트로 사용된다. 일단 특정 버스 구성이 클러스터링을 사용하여 확인되면, 정보는 버스의 크기를 결정하는데 적합한 형태로 변환되어야 한다. 이전 작업의 매트릭스에서, 첫번째의 네개 엔트리는 제 1 그룹으로 클러스터링되며, 마지막 두개는 제 2 그룹으로 클러스터링된다.

<486> 버스 크기를 계산하는 것은 전송된 데이터량에 요구되는 밴드폭을 결정하며 밴드폭을 계산하고 서로 다른 버스 폭값을 타겟 밴드폭이 가능한 가깝게 접근할 때까지 교환하는 것을 필요로 한다.

<487> 타겟 밴드폭 결정

<488> 패턴 세트에서 버스에 필요한 타겟 밴드폭을 결정하는 것은 다음 단계를 필요로 한다.

<489> 1. 피봇된 데이터 전송 매트릭스에 각각의 클러스터에서 발생한 모든 트랜잭션을 추가하는 단계. 동일한 예에서, 큰 클러스터에는 62,600, 작은 클러스터에는 100, 클러스터간에는 1,200이 존재한다. 도 55의 매트릭스는 따라서 도 54의 각각의 4개 그룹에 엔트리를 추가함으로써 생성된다.

<490> 2. 이러한 패턴 세트에 걸리는 시간을 결정하는 단계. 프런트-엔드 억셉턴스 작업은 이 정보를 제공한다. 패턴 세트는 일 밀리초안에 전송되어야 하며, 즉 빠른 클러스터가 1ms에 63,800 바이트의 데이터-브리지에 대해 1,200 바이트 및 버스의 내부에서 62,600 바이트-를 전송하여야 한다. 밴드폭은 일 초에 전송될 수 있는 비트에서의 데이터량으로 정의된다. 상기 예에서, 1ms에 510K비트를 전송할 수 있으며, 밴드폭은 대략 510MHz이다.

<491> 버스폭 계산

<492> 버스(버스폭)에서 다수의 와이어로 구성된 밴드폭은 데이터가 전송되는 클록 주파수의 시간을 조절한다.

<493> 계산은 다음과 같다.

<494> (util/clock_cycle)×bus_width = bandwidth

<495> util은 선택된 버스 타입에 대한 최소의 버스 이용 퍼센트(도 59참조);

<496> clock_cycle은 설계를 위한 클록 사이클이며;

<497> bus_width는 버스의 와이어 수이고, 이 값은 2이어야 한다.

<498> 계산을 위해, bus_width의 경우 2^1에서 시작하고, 최종 밴드폭값이 타겟 밴드폭보다 클때까지 높은값(2^2,2^3,...)을 계속적으로 교환한다. 예를 들어, 만일 clock cycle가 20ns라면 버스 이용도는 25%이며, 거의 2에서 근소화된 와이어수는 64비트이다.

<499> (25% / 20ns) * 26 = 800MHz 〉 510MHz.

<500> 이 예에서, 만일 도 59의 표에서 타입 4 또는 5 버스를 선택하였다면, 고속 클러스터의 버스에서 적어도 64비트가 필요할 것이다. 유사하게, 20ns 사이클 시간은 저속 클러스터에서 단지 8비트만을 필요로 할 것이다.

<501> 대기시간 정보는 증가된 버스의 이용이 대기시간을 증가시키기 때문에 이용의 부분적인 기능이 된다. 이 예를 단순하게 하기 위해, 복잡도가 포함되지 않으며; 이는 이용 수(utilization number)에서 부분적으로 설명된다. 그러나 일반적으로 만일 밴드폭 계산에 최소 버스 이용 수를 사용한다면, 대기시간은 최소값으로 향하기 쉽다. 이 효과를 설명하기 위해, 설계자는 클러스터로부터 최악의 경우(최소)의 대기시간 요구 조건을 선택해야 한다.

<502> 설계자는 시뮬레이션에서 사용된 대기시간 매트릭스로부터 전체 트랜잭션의 대기시간을 도출할 수 있으나, 도 59의 표는 분리된 수로서 버스 대기시간 데이터 및 전송값을 나타낸다. 도 59는 타입 4버스에 대한 10의 최대 전송 대기시간을 나타낸다. 최소 데이터 대기시간은 데이터에 필요한 사이클 수에 근접해진다. 설계자는 따라서 이하에 개시된 바와 같이 대기시간 매트릭스의 수로부터 데이터 전송 시간을 제거함으로써 네트 전송 대기시간이 얼마인지 계산할 필요가 있다.

<503> data_transfer_time = min_cycles / num_words * avg_trans

<504> min_cycles는 이 버스 타입에서 최소 데이터 대기시간 사이클

<505> num_words는 버스의 워드수; 및

<506> avg_trans는 평균 트랜잭션 크기이다: 즉, 트랜잭션 매트릭스(도 44)의 트랜잭션수에 의해 나누어진 데이터 전송 매트릭스(도 43)로부터 데이터의 바이트 수.

<507> 표로부터 대기시간을 비교하기 위해, 설계자는 시뮬레이션 매트릭스의 대기값에서 트랜잭션의 데이터 대기시간을 뺀 것을 사용하는 새로운 대기시간 매트릭스를 생성하여야 한다. 상술한 예에서, 이 표는 도 56에 도시되어 있다. 이 매트릭스의 각 엘레멘트는 다음과 같이 계산된다. [최종 대기시간(x,y) - Min 버스 대기시간 데이터(타입)] * (데이터 전송(x,y) / [트랜잭션(x,y) * 버스 크기])

<508> 시스템 버스 클러스터의 최소수는 25이다. 이 값은 밴드폭으로 인해 필요한 버스 타입의 큰 전송 대기시간보다 커야 한다. 그 수는 버스 타입 4의 전송 대기시간에 대해 도 59의 표에서 10이며, 따라서 설계자는 고속 클러스터에 대해 버스 타입 4이상을 선택할 수 있다.

<509> 버스 계층 구조 생성

<510> 일단 설계자가 버스 및 그 로드를 확인하면, 버스 성능 계층 구조가 확인되어야 하며, 어떤 것이 고속 버스인지 저속 버스인지, 어떤 브리지 및 아비터가 필요한지를 결정한다. 만일 두개의 버스가 감소된 버스 매트릭스(셀로부터/셀에 이르는 값은 제로값이 아님)에서 연결되며, 이들 사이에서 브리지를 생성한다. 도 54의 예를 사용하여, 피봇된 데이터 매트릭스 및 감소된 버스 매트릭스로부터 다음의 버스 모델을 생성한다.

<511> 64비트의 버스 시스템(4 또는 5 타입)은 다음에 접속된다:

<512> Block 1(RNV)

<513> Block 2(RNV)

<514> Memory(RNV)

<515> PCI(RNV)

<516> 8비트의 주변 버스(타입 3이상)에 대한 브리지(RNV)는 다음에 접속된다:

<517> Block 3(R/W)

<518> PIO (Write only)

<519> 주의 : PIO는 이로부터 오는 데이터가 없기 때문에 기록만 가능하다. 브리지는 버스 1과 2사이의 양대각선이 제로값이 아니기 때문에 판독/기록이 가능하다.

<520> 이 맵은 버스 설계를 정의하면서 다음 작업으로 진행된다.

<521> 버스 설계 지정

<522> 버스 설계를 지정하기 위해, 설계자는 원본 블록에 대한 인터페이스 스펙 세트, 브리지 및 아비터와 같은 새로운 블록 세트 및 글루 로직 세트로 생성된 버스를 확장해야 한다. 원본 및 새로운 블록 스펙은 블록 설계 작업으로 통과한다. 미니-블록과 같은 글루 로직은 블록 설계를 통하여 칩 어셈블리 작업으로 전달된다. 만일 버스가 OCB 특성 스펙에 합당하다면, 아비터 및 브리지와 같은 다른 버스 오브젝트외에 마스터 및 슬레이브 장치에 대한 모델을 가진다. 맵 정의된 선택 버스를 사용하여, 설계자는 자세한 버스 구조를 생성한다.

<523> 상세한 버스 구조

<524> 상세한 버스 구조를 생성하기 위해, 설계자는 다음을 행해야 한다.

<525> 1. 단일 로그 및 브리지로 모든 버스를 제거함으로써 버스를 최적화한다. 로드는 단지 하나의 로드에 대해 주변 버스 및 시스템 버스의 프로토콜 사이에서 전송하기 위해 게이트 시간에서 더 느리며 더욱 비싸기 때문에 브리지의 다른 면에 위치할 수 있다. 설계자가 브리지 로직을 전반적으로 제거할 수 없는 경우에 트리상태 인터페이스는 버스가 포인트-투-포인트 통신으로 축소되기 때문에 제거될 수 있다. 또한 8비트는 두개의 단부가 서로 함께 위치할 수 있기 때문에 많은 페널티없이 16으로 변할 수 있다.

<526> 2. 버스 마스터 및 슬레이브를 여러 로드에 할당한다. 설계자는 브리지에서 시작하여야 한다. 느린 면에는 마스터가 빠른 면에는 슬레이브가 존재한다. 주변 버스의 모든 장치는 슬레이브 장치이다. 시스템 버스에서, 마스터 및 슬레이브는 버스를 제어하는데 필요한 장치에 의해 정의된다. 설계의 인식은 상기의 결정에 도움이 된다. 만일 프로세서가 버스에 접속된다면, 그 인터페이스는 마스터이다. 그렇지 않고 만일 어떠한 분명한 마스터도 존재하지 않는다면, PCI와 같은 외부 인터페이스는 마스터이다. 메모리 인터페이스는 언제나 슬레이브 인터페이스이다. 어떤 블록이 마스터 인터페이스를 요구하는지를 결정하기 위해, 설계자는 버스에 대한 상호 접속 요구조건을 참조하여야 한다.

<527> 3. 만일 프로세서 또는 다른 블록이 메모리 인터페이스를 가진 버스에 접속된다면, 그리고 블록이 특별하게 이를 요청한다면, 설계자는 버스상에 하나 이상의 직접 메모리 액세스(DMA) 장치를 포함시켜야 한다. 상기 장치는 버스 마스터로서 작동하여야 한다.

<528> 4. 최종적으로 만일 버스에서 두개 이상의 장치가 버스 마스터라면, 아비터를 추가한다.

<529> 상세한 버스 설계

<530> 버스 구조가 정의될 때, 블록 버스 인터페이스가 체킹된다. 만일 블록이 이미 버스 인터페이스를 가졌다면, 인터페이스는 버스에 대한 주문을 위해 소프트, 펌, 또는 파라미터화된 폼일 것이다. 만일 이 경우라면, 현존 버스 인터페이스 로직이 사용되어야 하며, 그렇지 않다면, 버스에 제공된 모델이 용인될 수 있다. 만일 블록상에 서로 다른 버스 인터페이스가 존재한다면, 가능할 경우에 제거되어야 한다.

<531> 버스 로직은 다음과 같이 버스에 대한 인터페이스로 변경되어야 한다.

<532> 1. 각각의 인터페이스에 어드레스 스페이스를 할당한다. 어드레스 스페이스는 일반적으로 사익 블록이 어드레

싱되는지를 결정하기 위해 트랜잭션 어드레스의 상위 비트에 매칭하도록 설계된다. 또한, 각각의 블록이 블록에서 사용된 동작 코드 또는 내부 저장에 충분한 어드레스 공간을 가지는 것을 보장해야 한다.

<533> 2. 만일 단지 하나의 기능만이 사용된 경우 판독 또는 기록 버퍼를 제거한다. 대부분의 기존 버스 인터페이스는 판독 및 기록이 모두 가능하도록 설계된다. 설계자는 상기 기능들 중 하나만이 필요하다면 로직을 상당히 줄일 수 있다. 예를 들어 만일 버스가 하나 이상의 클록 사이클을 요구한다면, 판독 및 기록 데이터는 통상적으로 분리되어 버퍼링된다. 만일 단지 하나의 기능만이 필요하다면, 설계자는 레지스터 비트를 반으로 줄일 수 있다.

<534> 3. 정의된 버스 크기에 합당하게 설계를 확장 또는 수축시켜야 한다. 대부분의 버스 인터페이스는 표준 32 또는 64 비트 버스에 대하여 설계되지만, 다른 대안이 가능하다. 만일 설계자가 비표준 버스 인터페이스를 필요로 한다면, 레지스트 및 신호 라인을 제거 또는 추가하기 위해 로직을 변경해야 한다. 유사하게, 어드레스는 통상적으로 데이터와 동일 크기이지만, 이것이 그 경우는 아니다. 동일 버스 신호로 어드레스 및 데이터를 인터리빙하는 버스의 경우 데이터 및 어드레스 크기의 미스매치는 신호가 아닌 상위 어드레스 디코드 또는 데이터 레지스터 로직만을 제거하여야 한다.

<535> 4. 만일 필요하다면 브리지에 버퍼를 추가한다. 상기의 변경은 단계 3의 브리지의 양쪽에서 이루어져야 한다.

<536> 5 버스간의 브리지 크기 맵핑을 수정한다. 판독/기록 인터페이스의 경우, 브리지는 각 기능에 대해 적어도 하나의 레지스터를 필요로 하며, 양쪽에서 버스의 큰쪽과 동일하다. 각 기능에 대한 데이터 버퍼에 추가하여, 데이터 버스트는 데이터가 도 57에 도시된 브리지를 사용하여 다음 버스에 전송되기 전에 브리지에 의해 받아들여지는 경우에 보다 효율적으로 전송될 수 있다. 이는 버스트를 저장하며 도 58의 브리지에 도시된 바와 같이 다음 버스에 이를 포워딩하기 위해 각각의 기능에 FIFO를 필요로 한다.

<537> 6. 버스 마스터 및 중재 타입의 우선권을 정의한다. 만일 버스상에 하나 이상의 마스터가 있다면, 마스터간의 어떤 종류의 중재가 존재하여야 한다. 라운드-로빈 중재에 대한 엄정한 우선권으로 정해진 여러 타입의 중재가 존재한다. 만일 마스터가 유사한 수의 트랜잭션으로 동일 데이터량 및 요구되는 대기 시간을 모두 취급한다면, 동일한 우선권을 가져야 한다. 한편, 만일 마스터의 중요도에 관한 분명한 랭킹이 존재한다면, 중재는 대부분의 중요한 마스터를 먼저 실시하면서 차례로 나열하여야 한다.

<538> 7. 단계 5에서의 정의를 기초로 아비터를 생성 및 연결한다. 중재 계획은 버스를 기초로 분배 또는 집중화될 수 있다. 중재 로직은 이를 글루 로직과 함께 블록으로 분배할 수 있도록 가능한 분배되어야 한다.

<539> 8. 장치의 엔디안-니스에 의해 요구되는 인터페이스 로직에 버스를 맵핑한다. 대부분의 버스는 리틀-엔디안(little-endian)이지만, 어떤 장치는 빅-엔디안(big-endian)이다. 엔드 타입사이에 미스매치가 존재할 때, 설계자는 버스로부터의 데이터의 바이트를 스왑하는 방법을 결정하여야 한다. 이러한 결정은 일반적으로 문맥-의존이다. 만일 버스로의 모든 트랜잭션 및 버스로부터의 모든 트랜잭션은 동일 타입의 데이터이며, 설계자는 고정된 바이트-스와핑을 사용할 수 있으며, 그렇지 않다면 버스 마스터는 스와핑을 실시하여야 한다.

<540> 9. DMA 장치를 버스에 맞게 제작한다. 다이렉트 메모리 액세스 장치는 일 블록으로부터 다른 블록으로 데이터를 전송하는 제어기이다. 이는 임의의 다른 장치와 마찬가지로 어드레스 버스의 크기에 맞추어 변경되어야 한다.

<541> 10. 만일 필요하다면 시험 용이성 부분과 인터페이스를 추가한다. 최저 레벨의 테스트는 버스 자체를 테스트하는 능력이다. 표준 칩 테스트 로직은 또한 버스를 사용할 수 있다. 상기의 테스트 특성은 표준 동작 모드로부터 테스트를 차별화하기 위해 추가 신호를 필요로 할 것이다.

<542> 11. 만일 필요하다면, 초기 파라미터를 추가한다. PCI와 같은 버스는 구성 레지스터를 가진다. 이 레지스터는 변경되지 않은 구성에 대해 하드코딩될 것이다.

<543> 12. 버스에서 장치에 의해 요구된다면 추가의 버스 용량이 추가된다. 어떤 버스는 쓰레드, 스플릿 트랜잭션 및 에러 재시도와 같은 개량된 성능을 가지며, 이는 버스에 접속된 장치가 이들을 필요로 하지 않는다면 수행될 필요가 없다. 어떤 추가의 성능은 예를 들면 DMA 장치들, 불연속 버스트 전송 및 에러 복구 제어는 표준 버스에서 정의된 것보다 많은 신호를 필요로 한다. 상기의 신호는 만일 필요하다면 버스에 추가되어야 한다.

<544> 이 변경이 완료되면, 버스 인터페이스 로직은 블록의 최종 인터페이스에 접속된다.

<545> 버스 분류 표준

<546> 버스 분류 표준은 셀 라이브러리에서 이용될 수 있는 버스에 대한 데이터의 방향 및 밴드폭, 대기시간과의 관계와 버스 특성을 리스트한 라이브러리이다. 분류 표준은 정보의 상대적으로 고정된 집합이다. 이 라이브러리를 담당하는 사람은 새로운 버스가 입수될 때 버스 특성을 업데이트할 필요가 있다.

<547> 버스 타입 표준

<548> 버스 타입은 대기시간 및 밴드폭 이용에 의해 분류될 수 있다. 순수 밴드폭은 데이터가 전송될 때의 클록 주파수 곱하기 버스의 와이어의 수에 대한 함수이지만, 밴드폭 이용은 구조의 함수이다.

<549> 도 59는 최저 밴드폭 이용으로부터의 특정 버스 특성 및 최저 밴드폭 이용에 대한 최장 대기 시간 및 최단 대기 시간의 리스트를 도시한다. 전형적으로 로직 및 와이어의 비용은 처음이 가장 작고 나중이 가장 크다. 라이브러리의 각 버스는 이 표에 할당된 버스 타입을 가져야 한다. 각각의 버스 타입은 이용 퍼센트에서 버스 밴드폭 및 사이클된 대기시간 범위를 가질 수 있다. 각 버스는 서로 다른 클록 사이클 시간 및 크기를 가질 수 있으며, 이용 퍼센트는 버스의 크기 곱하기 사이클 제품에 대한 효과적인 처리량이다. 버스 사용효율 값이 100%라 함은 각 사이클이 충분히 이용됨을 의미한다. 데이터 대기 칼럼(data latency colume)는 버스가 데이터 워드를 전송하는 사이클의 수를 의미한다. 전송 대기 칼럼은 버스 트랜젝션(transaction)을 시작하는 평균 사이클의 수를 의미한다. 도 59의 테이블은 버스 사용효율과 지연 값의 대략적인 값을 보여준다. 설계자 그룹은 경험과 이 설계 타입에 기초한 값을 지정할 수 있다.

<550> **버스 분류 참조(Bus Taxonomy Reference)**

<551> 수많은 프로젝트에서, 설계자 그룹은 버스의 라이브러리를 수집한다. 각 버스는 도 41에 나타난 참조 라이브러리로부터 버스의 타입을 포함하는 일련의 정보를 포함하며, 버스의 리스트는 VSI OCG 특성 스팩에 나타나 있고, 버스 분류 참조문헌["Block-Based Design Methodology Documentation" version 1.2, May 21, 1999. at section B. 2 pages B.2 to B-10]에 나타나 있다. 이 정보는 기술된 바와 같이 사용될 버스를 결정하는 데 사용되어야 한다.

<552> **테스트용 설계(Design for Test)**

<553> 상기 종래기술에서 설명한 바와 같이, 각 테스트는 SOC 설계에 가장 중요한 요소중의 하나이다. 따라서, 테스트용 설계("DET")는 표준이 되었다. 소정의 고객 스펙에 대해서, 본 발명의 시스템과 방법을 이용하여 유도된 DET 지식은 질문 & 답변(Q&A)의 형태로 고객에게 제공된다. 이 디바이스를 통해서, 데스트 객체들은 교섭될 수 있으며 프런트 엔드 허용 동안에 교섭된 SOW(Statement Of Work)에서 테스트 이슈가 해결될 수 있다.

<554> 테스트 설계 후에 테스트 버짓팅(budgeting), 테스트 스케줄, 테스트 처리가 따르고, 일련의 스팩 및 테스트 설계를 만들어, 테스트 개발을 분리되고 서로 독립적인 일련의 공지된 수단과 절차를 갖는 한정된 목적의 하부태스크로 침투하게 한다.

<555> 각 테스트 블록은 가장 효율적인 기술을 가진 테스트 가능한 상기에서 기술된 방법과 관련하여 동시에 발전한다.

<556> 일단 테스트 블록이 테스트 통합할 준비가 되어 있다면, 이들은 제한받지 않는(unconstrained) SOC 주변으로 매핑될 수 있고, I/O 제한이 적용되지 않아, 따라서 각 층은 설계 블록으로 전환되는 제한받지 않는 SOC용 "테스트-준비" 텐플레이트(template)가 된다. 제한받지 않는 SOC는 이 때 추가적인 I/O 레벨 테스트를 갖는 특정 I/O 패킹에 한정된다. 이는 테스트 스케줄 프로세서가 일어나게 하고, SOC 레벨 테스트 객체를 채우게 한다.

<557> **DFT 테스트 설계하기(Making a DET Test Plan)**

<558> FEA 동안 고객의 설계를 안 후에, 본 발명의 테스트 설계 개발 계획은 이것이 통합가능한가(이 테스트가 다수의 블록에서 동시에 실행되는 지)를 보여주는 각 블록의 평가로 시작된다. 다음에 설계자는 합병될수 없는 각 블록는 얼마나 테스트 가능한지를 결정한다. 세 번째로, JTAG 주변 스캔, DC 테스트 및 PLL 테스트와 같은 테스트 타입을 포함하는 칩-레벨 테스트 스팩이 개발된다. 최종적으로 테스트의 디폴트 커버리지(default coverage)는 전체 칩 레벨에서의 테스트-통합 블록용, 블록 레벨에서의 비통합 블록용, 내부접촉용으로 지정된다. 이 4단계의 초기 분석의 결과는 본 발명의 전체 시스템의 설계를 위한 DET 객체를 제공한다.

<559> **DET 규칙의 이용(Using DET Rules)**

<560> 특정되고, 테스트 관련한 제한인 DET 기술 규칙은 일관된 테스트 개발 플로 및 부착성 테스트 데이터 통제를 유지하도록 사용된다. 이러한 규칙은 상위 레벨에서 실제적인 소켓에 위치한 비통합 블록에 테스트 특성들의 응용을 이끌어 내고, 가장 간단하면서도 가장 적합한 테스트 전략을 얻기 위해서 트래이드-오퍼(trade-off)의 실행을 어끌어 내고, 설계용 상위 레벨의 테스트 스팩을 만들어 내고, 테스트 설계를 유도하여 테스트 실행 프로세서을 세분화한다.

<561> **DET 용어설명(DET Glossary)**

<562> 발명의 상세한 설명 및 청구항에서 사용된 DET 용어는 하기와 같이 정의된다.

<563> 인가(Authorization) : 미리 설계 된 블록을 통합하는 것이 가능한 전환 프로세서

<564> BIST : 고유 자기 테스트(Built-in self test)

<565> BSR : 주변 스캔 레지스터

<566> CAP : 칩 액세스 포트

<567> CTAP : 코어 테스트 액세스 포트

<568> DAP : 설계 액세스 포트

<569> DFT : 테스트용 설계

<570> 폴트 커버리지(fault coverage) : 테스트의 고정된 폴트 커버리지

<571> ICTAP : 집적회로 테스트 액세스 포트

<572> IP : 지적 재산

<573> JTAG : 조인트 테스트 액션 그룹(Joint Test Action Group: iEEE-i 149.1)

<574> 레그시 블록(Legacy block) : 알려지지 않은 결과의 위험성 없이 재사용을 위해서

<575> 변경되거나 역으로 설계되지 않은 미리 설계된 게이트-레벨 블록

<576> 통합(Mergeable) : 하나이상의 다른 구성요소들과 결합되어, 유니트로서 테스트받

<577> 을 수 있으며 시간과 비용을 절약할 수 있는 통합 구성요소용 테스트 요구

<578> MISIR : 다중 입력 신호 제널레이트(Multiple input signature generator)

<579> Mux : 멀티플렉서

<580> 비통합(non-mergeable) : 평행 테스트를 위해 다른 블록과 통합되지 않음

<581> PRPG : 의사-랜덤 패턴 제널레이트(Pseudo-random pattern generator)

<582> SAP : 소켓 액세스 포트

<583> 소켓화(socketization) : 설계한대로 테스트하기 위한 미리 설계된 블록에

<584> 테스트 칼라를 가하고 열거하는 적응 프로세서

<585> TAP : 테스트 액세스 포트(test access port)

<586> TBA : 테스트 버스 구조(test bus architecture)

<587> 테스트 칼라(test collar) : 테스트 액세스와 컨트롤을 제공하는 미리-설계된

<588> 블록을 둘러싸는 로직 및 테스트 포트의 집합

<589> 타임셋(Timeset) : 주기적인 테스터기의 시간 포맷 : RZ(return to zero), NRZ(nonreturn to zero), RTO(return to one), DNRZ(delayed nonreturn to zero)

<590> UDL : 유저-한정 로직(user-defined logic)

<591> VC : 가상 컴포넌트(virtual component)

<592> 가상 소켓(virtual socket) : 테스트 인터페이스를 포함하는 미리-설계된 블록용 플레이스홀더

<593> VSIA : 가상 소켓 인터페이스 연합

테스트 설계 하기(Making a test plan)

<595> 전체 DFT 테스트 설계를 하는 공정은 테스트 설계자가 FEA-제널레이트화된 입력으로부터, 각 블록의 테스트 기술, 예상되는 테스트 벡터 스팩, 생산용 테스트 요구시간 및 I/O에 의해 공급된 특정 파라미터나 아날로그 테스트 및 아날로그/혼합-신호("AMS") 요구 모듈(xref)을 얻음으로서 시작된다. 완전한 DET 설계는 효과적인 조직과 이들 데이터의 이용을 포함한다.

<596> **비통합 블록용 테스트 요구(Test Requirements for Non-Mergeables Blocks)**

<597> 칩-레벨 테스트 요구는 비통합 블록 테스트 요구를 포함하는 데, 이 통합 블록 테스트 요구는 차례로 4가지 구성요소 즉 테스트 모듈, 전용 테스트 포트와 테스트 모듈과 간은 테스트 컨트롤 로직, 세이프아웃(safe-out)과 같은 테스트 격리 로직, 및 테스트 벤치와 테스트 벡터와 같은 테스트 유효 구성요소를 포함한다. 비통합 블록이 고객에게 전달될 때, 테스트 액세스 및 컨트롤 데이터(예를 들어, 테스트 모듈, 활성화 및 비활성화), 테스트 프로토콜, 테스트 데이터, 테스터기 포맷 및 테스트 응용/셋업 시간을 지정한다.

<598> **통합 블록용 테스트 요구(Test Requirements for Mergeable Blocks)**

<599> 칩-레벨 테스트 요구는 차례로, 테스트 방법, 테스트 컨트롤 로직, 내부접촉 실행 메카니즘 및 테스트 유효 구성요소를 포함하는 모든 테스트-통합 블록용 테스트 정보를 포함한다.

<600> **칩-레벨 테스트 요구(Chip-Level Test Requirements)**

<601> 칩-레벨 테스트 요구는 또한 DC 테스트 요구, AC 테스트 요구, 전력분배와 같은 Iddq 테스트 요구 및 아날로그 테스트 요구를 포함한다.

<602> **칩-레벨 테스트 컨트롤러(Chip-level test controller)**

<603> 칩 레벨에서의 테스트 컨트롤러는 테스트 인터페이스, JTAG, PRPG 및 MISR 일 수 있다.

<604> **구성요소 특성 매트릭스(Component Attributes Matrix)**

<605> 설계자는 BBD 설계에서 구성요소용 테스트 개발 환경을 설계하는 매트릭스를 사용할 것이다. 이 매트릭스 기록은 가능한 결과를 산출하고, 추천하고, 평가하고, 추가적인 정보가 필요한 곳을 알아낸다. 이 매트릭스는 테스트 설계에서 발생하는 어려움과 불일치가 있는 영역을 확인시켜 준다.

<606> **DFT 규칙의 이용(Using DFT Rules)**

<607> 일단 설계자가 매트릭스를 사용하여 칩-레벨 테스트 요구를 여과하고 분류하고 난 후에, 일련의 DFT 기술 규칙의 이러한 요구를 프로세서한다. 공통 액세스, 테스트 컨트롤, 테스트 클럭 및 비동기 특성 및 사용가능한 DFT 기술에 기초한 트레이트-아웃의 설정이 설계된 칩용 특정 하이브리드 DFT 기술을 만들 수 있게 한다.

<608> 적합성은 본 발명의 BBD DEF 전략의 두드러진 특징이다. 적절한 테스트 통합을 위해서, 설계자는 가상 소켓을 강제된 각 비통합 블록과 프런트 엔드 허용에서 받은 테스트 정보를 정렬한다. DFT 기술은 이러한 가상 소켓을 칩-레벨 테스트 요구의 레스트(rest)에 이러한 가상 소켓을 통합함으로써 스팩를 완성한다. 각 가상 소켓은 테스트 데이터의 변환을 위해서 칩 액세스 포트(CAP)에 맵팅된 소켓 액세스 포트(SAP)를 갖는다.

<609> 설계자가 테스트 설계를 하고, 테스트용 설계를 준비하기 시작하기 전에, 일관성과 결합성을 위해서 그룹의 DFT 기술을 체크하여야 한다.

<610> **일관성(Consistency)**

<611> 일관성은 4가지 동작 모드: 노말(normal), 테스트(test), 격리(isolation) 및 주변(boundary:co-test)에서 각

구성요소의 테스트 개발 커버리지가 완전한지의 정도를 나타낸다. 설계자는 각 구성요소용 채크리스트를 사용하여, 이것의 모델, 콘트롤러 설계, 격리 및 테스트 유효 값이 각 블록 및 칩-레벨 디스크립션(description) 사이에서 일정하도록 한다.

<612> 예를 들어, 3개의 비통합 블록 A, B 및 C를 갖는 설계에서, B와 C가 격리되면, 테스트 컨터롤러 설계는 단지 블록 A만을 테스트한다. B와 C가 격리되었을 때, 테스트 컨트롤러 스팩은 블록 A가 액세스를 테스트할 수 있어야 한다. 만약 블록 B 및 블록 C가 동시에 테스트되면, 테스트 컨트롤러 스펙은 하나의 시뮬레이션 환경에서 이 테스트 데이터를 동기화시키는 테스트 유효 스킴(scheme)으로 두 블록에 대한 액세스를 테스트하여야 한다.

<613> 상기 예에서, 도 60의 테이블은 예시적인 블록 A 일관성 체크를 도시하고 있다.

<614> **부착성(Cohesion)**

<615> 부착성은 플로(flow)에서 테스트 방법의 다른 것과의 관련 정도를 나타낸다. 4가지의 밀접하게 관련된 방법 파라미터가 있는 데, 이들 각각은 다른 것을 수정한다. 예를 들어, 테스트 액세스 방법은 테스트 프로토콜의 활성 조건을 한정하고, 테스트 프로토콜은 어떻게 테스트 데이터를 정렬할 것인지를 한정하고, 데스터 데이터는 특정 테스터기 타임셋을 갖는 일련의 패턴으로 파괴된다. 외부 블록으로의 테스트 액세서는 칩 I/O 제한 및 콘트롤러 설계에 매우 민감하기 때문에, 이러한 파라미터의 부착성은 테스트 데이터의 완전성을 유지하도록 특정의 검증 스타일을 요구한다. 5가지의 테스트 방법 파라미터는 테스트 액세스, 테스트 프로토콜, 테스트 데이터, 데스터기의 타임셋 및 테스트 타임이다.

<616> **구조 규칙(Architecture Rules)**

<617> 도 61는 DFT 관점에서부터 칩의 상위-레벨의 계층을 도시한다. 설계자가 DFT 프로세서를 시작하기 전에, 설계자는 도 61에 도시된 바와 같이 칩을 오히려 기능성 블록의 집합으로서 시각화한다. 도 62는 비통합 블록이 소켓화된 SAPs 및 DAP을 갖는 기능성 블록으로 만들어 진 설계를 도시한다.

<618> 실제로, 설계에서 기능성 블록은 행동, RTL, 게이트 또는 혼합-레벨 HDL에서 기술될 수 있다. HDL 파일은 디렉토리(directory) 구조에서 조직화된다. 테스트 파일을 조직화하는 바람직한 방법은 하기의 구조 규칙에서 도시된 바와 같이 직접적인 계층을 만드는 것이다. 이 경우에 있어서, 칩은 HDL를 사용하는 다른 구성으로 설정된다.

<619> 칩-레벨 DFT 구조는 단지 하나의 레벨을 지니기 때문에, 모든 특성은 상위 레벨에 있다. 따라서, 설계자는 상위 레벨 설계 파일에 추출가능 코멘트 폼으로 특성을 집어넣기 위해서 본 발명의 방법과 관련된 하기의 구조 규칙을 사용한다.

<620> 1. 계층적으로 DFT 구조를 기술한다.

<621> 2. 가장 높은 레벨의 계층에 하나의 칩 액세스 포트(CAP)를 만든다. CAP 스팩은 바람직하게

<622> a. 설계과 테스트 데이터를 일정하게 유지하도록, 모든 데이터 컨트롤 및 테스트 데이터 핀을 팻키지-레벨 핀에 맵핑하여야 한다.

<623> b. 테스트 데이터 핀으로부터 데스터 컨트롤 핀을 분리하여야 한다.

<624> c. 전용화되거나 선택적인 특성을 갖는 테스트 컨트롤 핀을 설정하여야 한다. i) 노말 모드에서 예외적으로 비활성화될 수 있다면 전용화되었고 하는 데, 전용 핀은 기능성 핀과 공유할 수 없다.

<625> ii) 테스트 동안에 테스트 상수-논리값-에 설정된다면 선택적이라고 하는 데, 선택적 핀은 기능성 핀과 공유할 수 있다.

<626> d. 테스트 데이터 핀은 다음의 특성을 가진다.

<627> 테스트 클럭 : 테스트 동안에 클럭으로 사용된다면, 테스트 클럭이라고 하는 데, 테스트 클럭은 단지 외부 기능성 클럭 핀과 공유될 수 있다.

<628> 테스트 비동기 : 리셋을 위한 테스트 동안에 비동기적으로 사용된다면, 테스트 비동기라고 하는 데, 테스트 비동기 핀은 다른 테스트, 테스트 모드 또는 격리 모드와 어떤 충돌을 야기하지 않는다면 전용화되고 공유될 수 있다.

<629> 테스트 그룹 : 테스트 동안에 테스트 그룹 핀이 동기화되는 테스트 클럭이면 테스트 그룹이다.

<630> e. 각 테스트 모드에 대해서 하기에서 기술된다.

<631> i) 엑세싱 시퀀스를 요구한다면 테스트 하에서 디바이스로의 액세스를 얻는 데 필요한 테스트 셋업. JTAG 명령어, 테스트 클럭 또는 테스트 리셋과 같은 프로토콜을 기술한다.

<632> ii) 실질적인 테스트를 실행하는 데 필요한 테스트 실행. 스택 레벨, 반복 카운트, 사이클 시간, 테스트 길이 및 테스트 결과에 내려진 테스트 시퀀스를 페이저 형태로 기술한다.

<633> iii) 디폴트 조건(노말 모드)에서 테스트를 끝내고 칩을 되돌리기 위해 필요한 테스트 포스트프로세싱.

<634> 3. 테스트 셋업을 설명하고 각 테스트 코드용 프로세싱 시퀀스를 설명하는 CAP 컨트롤러 스팩을 만든다. 스팩은 바람직하게는 실행(합성)될 수 있고(테스트 벤치와 테스트 시퀀스를 통해), 검증될 수 있다.

<635> 4. 설계자는 선택적으로 일련의 스테이징 레치(staging latch)를 연결하여 내부 테스트 데이터 버스를 사용가능한 테스트 데이터 핀에 겹친다. 이 스테이징은 바람직하게 이후의 테스트 결과를 변화시키지 않는다. 스테이징은 바람직하게는

<636> a. 상태가 변화될 수 있고 시간에 민감한 신호로부터 자유롭다. 테스트 동기화 신호를 사용하거나, 이것을 해결하도록 테스트 클럭과 관련되어 일어나는 불변적인 차수에 따른다.

<637> b. 상태가 변화될 수 있고, 시간에 민감한 신호로부터 자유로우면, 이는 여분의 테스트 핀을 가져야 한다. 이 규칙은 테스트 팻키징(packaging) 문제를 피하기 위해서 바람직하게 사용되어져야 한다.

<638> 5. 설계자는 선택적으로 물리적인 I/O 제한을 줄이는 테스트 데이터를 압축하기 위해서 MISR과 같은 테스트 데이터 서명 분석 능력을 지정한다. 서명 분석은 각각의 주기 동작에 대해 바람직하게 결정되며, 바람직하게,

<639> a. MISR 입력에서 이것을 피함으로써 X-값 전파로부터 자유롭다.

<640> b. 단계 a가 실패하면, MISR 사이클을 억압한다. 이 규칙을 폴트 커버리지(fault coverage)의 손실을 피하기 위해서 주의깊게 따라야 한다.

<641> 6. 설계자는 선택적으로 하기의 특정 테스트(주변 스캔 테스트와 같은 DC 및 AC 파라미터 테스트, PLL 테스트와 같은 주파수 테스트, ADO 및 DAC 테스트와 같은 혼합-신호 테스트)를 형성하도록 칩 주변에서 다른 일련의 테스트 메카니즘을 만든다. 이러한 테스트용 컨트롤 핀은 바람직하게는 모든 테스트 컨트롤 핀 테이블내에 포함되어야 한다. 설계자는 내부접촉을 피하기 위해서 CAP 컨트롤러 스팩에 이들을 포함하길 원한다.

<642> 7. 다음 레벨의 계층에서 신호 디바이스 액세스 포트(DAP)을 지정한다. I/O 또는 I/O와 관련된 셀이 없는 레벨은 물리적 I/O에 제한되지 않는다.

<643> 8. DAP는 바람직하게는 TAP에 기초한 포트와 같은 멀티플렉싱 일반 포트를 연결하고, 통합하고, 사이즈를 바꿈으로써 형성될 수 있는 하이브리드 테스트 포트여야 한다.

<644> 9. 설계자는 바람직하게는 CAP 컨트롤러로부터 DAP를 직접적으로 형성할수 있어야 한다. 각 구성을 테스트 컨트롤, 테스트 데이터 또는 테스트 격리 포트로 분배한다. 각 구성에서

<645> a. 하기의 특성을 갖는 테스트 컨트롤 포트를 설정한다..

<646> 목표된 구성 k를 설정하는 데 바람직하게 사용된다면, 테스트 콘(test con) f(k),

<647> 테스트 상수로 설정된다면 테스트 선택.

<648> b. 하기의 특성을 갖는 테스트 데이터 포트를 설정한다.

<649> 테스트 동안에 클럭으로서 사용된다면, 테스트 클럭,

<650> 테스트 동안에 비동기적으로 사용된다면, 테스트 비동기,

<651> 포트가 동기화된 테스트 클럭을 가리키는 테스트 그룹(i),

<652> 테스트 데이터 방향을 가리키는 데 사용된다면, 단지 1 또는 0의 값의 테스트 방향.

<653> c. 테스트 동안에 0, 1 또는 Z의 안정 상태의 논리값으로 바람직하게 격리될 수 있다면 안정 상태 및 비플로팅(non-floating) 논리값 0 또는 1로 설정될 수 있다면 돈트 캐어(dont care)에 특성을 갖는 테스트 격리 포

트를 설정한다.

<654> 10. 상위레벨 DFT 구조에서 CAP 및 CAP 컨트롤러, 스테이징 래치, MISR, DAP 및 다른 테스트 매카니즘의 내부 접촉을 지정한다.

<655> 11. 전용 섹션에서 CAP 컨트롤러, 스테이징 래치, MISR, 설계 바디 및 다른 테스트 매카니즘을 지정한다.

<656> 12. 설계 바디 구조용 DAP, 소켓, UDL, 및 테스트 내부 접촉에 상세히 지정한다.

<657> 13. 설계 바디 구조(design body architecture)는 바람직하게 계층적으로 기술되어야 한다.

<658> 14. 바람직하게는 소켓 레벨의 다음 계층 레벨에 다수 SAP이 있어야 한다.

<659> 15. 각 SAP는 바람직하게는 DAP에 유용한 하나 또는 수많은 응용가능 구조를 지닌 DAP의 순환적인 이미지(image)이여야 한다. SAP의 모든 구조는 바람직하게는 DAR에 의해 지지되어야 한다.

<660> **소켓화 규칙(Socketization Rules)**

<661> 일단 비합병 블록 또는 VC이 설계되면, 이것의 I/O 포트는 더 이상 칩 I/O로부터 액세스할 수 없다. I/O포트에서 만들어진 테스트 데이터는 더 이상 사용되지 않는다.

<662> 일반적으로 칩 레벨에서 테스트 데이터를 다시 만든다는 것은 어렵고, 예측하기 어렵다. 왜냐하면, 설계 블록 테스트 값은 다른 논리 블록을 통해서 전파되어야 하기 때문이다. 따라서, 바람직한 접근은 설계 블록용 가상 소켓을 만들므로써 접근능력(accessibility)을 설계 블록에 첨가하는 것이다. 가상 소켓은 테스트 액세스, 격리 및 칩 I/O으로부터 접근가능한 주변 테스트 기능을 포함한다.

<663> 설계자는 설계에서 설계 블록용 플레이스홀더로서 가상 소켓을 사용할 수 있으며, 또한 설계 블록 자체에 테스트 제한을 가하는 소켓을 사용할 수 있다. I/O 맵핑 및 제한을 사용하는 설계에서 제한이 설계 블록에 맵핑될 때, 설계 블록은 소켓화된다. 이 제한은 설계에 센서티브하고 조건적이나, 설계 통합시에 설계 블록의 트랙을 유지하는 동안에 설계자가 각 설계 블록 소켓화 태스크를 나누게 한다. 기능적인 인터페이스를 유지하는 동안에, 소켓화된 설계 블록은 칩-레벨 테스트 제한에 매치되는 여분의 I/O 포트와 논리 또는 테스트 칼라를 필요로 한다. 인터페이스 시간은 약간 변화될 수 있기 때문에, RTL 코드에 테스트 칼라를 쓰고, 각 소켓화된 설계 블록용 합성에서 특성화되거나 재편성되는 것이 최상이다. 전체 설계를 합성한 후에 게이트 레벨에서 테스트 칼라를 첨가하는 것은 타이밍 문제를 야기시킨다.

<664> 설계 블록 소켓화 규칙은 하기와 같다.

<665> 1. 소켓은 계층적으로 기술될 수 있으나, 상위 레벨은 바람직하게는 모든 테스트 특성을 포함하여야 한다.

<666> 2. 소켓당 단지 하나의 SAP가 필요할 수 있다.

<667> 3. SAP는 소켓내의 각 엘리먼트를어떻게 격리할 것인지, 어떻게 테스트 할 것인지, 어떻게 진단할 것인지, 어떻게 디버깅할 것인지에 대한 테스트 정보용 참조내용이다.

<668> 4. 각 SAP는 바람직하게는 높은 레벨의 스팩과 관련되어 만들어지고 합성된다.

<669> 5. 설계자는 바람직하게는 높은 레벨의 구조 및 컨텍스트(context)에서 각 SAP이 노말, 테스트, 격리 및 주변 모드를 활성화하고 비활성할 수 있도록 변화시킬 수 있어야 한다. 이는 설계자가 소켓의 외부 테스트 정보 구조를 변화시켜야 한다는 것을 의미한다.

<670> a. 외부 테스트 정보 구조는 바람직하게는 VSIA 컨플라이언스 규칙에 특정된 표준화된 표현 언어에 따라야 한다.

<671> b. 표준화된 표현 언어가 사용가능하지 않다면, 테스트 정보 구조는 가상 소켓에 특성을 갖는 칩-레벨 설계 테스트에 따라야 한다.

<672> 6. 각 SAP는 바람직하게는 테스트 셋업, 테스트 실행 및 테스트 포스트프로세싱 시퀀스를 적절히 실행하는 것을 보증하도록 재포맷된 테스트 데이터를 갖는 소켓 레벨에 유효하여야 한다. 이것은 설계자가 소켓의 내부 테스트 정보 구조를 변화시켜야 한다는 것을 의미한다.

<673> a. 내부 테스트 정보 구조는 바람직하게는 모든 설계 블록 테스트 모델, 모든 기능성 블록 및 소켓에 의해

둘러싸인 모든 다른 로직을 포함하여야 한다.

<674> b. 내부 테스트 정보 구조는 바람직하게는 칩-레벨 시뮬레이션 환경에서 함께 시뮬레이트되고 내부적으로 작동되어야 한다.

<675> 7. 노말 모드에서, SAP와 관련된 모든 테스트 로직은 바람직하게는 비활성인 동시에 직접적이여야 하며, SAP으로부터 연속적이지 않아야 한다. 노말 모드는 하나의 테스트 컨트롤 포트에 의해 활성화되어야 한다.

<676> 8. 격리(정지) 모드에서, SAP와 관련된 모든 테스트 로직은 비활성화되어야 하며, 중간 충돌 없이 세이프-스데이터(safe-state) 값에 정렬되어야 한다. 어떠한 기능성 상태도 격리 시퀀스에서 나타나지 않을 것이다.

<677> 9. 테스트 모드에서, SAP과 관련된 모든 테스트 로직은 테스트 시퀀스를 시작하기 전에 바람직하게는 하나의 활성 시퀀스에 의해서 그 후에 선택적으로 시퀀스를 배열함으로써 인에이블(enable)되어야 한다. 테스트 시간을 최소화하기 위해서, 동일구성의 연속적인 테스트 시퀀스는 묶여져야 한다.

<678> 10. 소켓의 주변 로직 모두는 주변(코-테스트) 모드에서 테스트 가능하여야 하며, 상기 로직은 SAP과 관련된 테스트 로직을 포함한다.

<679> **상위 레벨 테스트 로직 스팩을 설계하기(Designing a Top-Level Test Logic Specification)**

<680> 설계자가 커버리지 및 시간 요구를 만족하도록 상위 레벨의 테스트 로직 스팩을 설계할 때, 테스트 로직의 평형 네이처(nature)를 증가시키는 트레이드-오퍼(trade-off)를 만드는 것이 필요할 것이다. 주요하게 결정할 내용은 각 블록 테스트를 하는데 어떻게 직렬 또는 어떻게 평형하게 할 것인지를 결정하는 것이다.

<681> 테스트 제한은 테스트 칼라를 만드는 데 필요한 테스트 요구를 설정하는 소켓화 규칙으로 각 가상 소켓을 위해 사용된다. 테스트 액세스 관점에서 보면, SAP는 완벽하며, 테스트 통합 목적에 적합하다. 설계 및 테스트 제한을 야기시킬 수 있는 설계 변화를 피하기 위해서, SAP는 블록의 기능성 엘리먼트를 공유하거나 사용하지 않아야 한다. 다른 타입의 블록, 소프트, 펌(firm) 또는 하드 블록이 사용될 때, 이 분리는 매우 바람직하여 테스트 통합동안에 비예측성을 피하는 것이 가능하다.

<682> 일반적으로, 각 구조는 특정한 일련의 해결이나 특정한 일련의 툴을 겨냥하여 특정 커버리지의 테스트 응용에 이용된다. 많은 구조는 설계에서 거의 모든 역할을 하는 특정 설계 환경에서 시작된다. 따라서, 개발 플로(development flow)는 하기와 같다.

<683> 1. 설계과 관련해서 테스트 문제점들을 특성화하고 분류한다.

<684> 2. 각 구조에 트레이드-오퍼(trade-off)를 부여한다.

<685> 3. 각 목표된 설계를 위해 추가적인 변경을 제공한다.

<686> 4. 본 발명 전까지, BBD 테스트 문제는 하기의 영역에서 명백하였다.

<687> 테스트 데이터 재사용성

<688> 테스트 소켓 설계 및 소켓 정보

<689> UDL 및 칩-레벨 내부접촉 테스트

<690> 테스트 팻키징

<691> 테스트 유효화

<692> 테스트 프로토콜

<693> 진단 및 디버깅

<694> 이러한 문제는 BBD 설계 설계 동안에 만들어진 가정과 관련된다. 그러나, 설계 설계는 재사용가능한 테스트 데이터를 가지고 설계 블록을 팻키징하는 많은 특정 프로세서를 요구한다. 예를 들어, 테스트용 BBD 설계를 제작하고, 설계 블록 테스트 인터페이스를 주문받아 만들고, 테스트 액세스 및 컨트롤 메카니즘을 설계하고 유효하게 하고, 테스트 버짓(budget)내에 칩 I/O로 테스트를 패키징한다.

<695> **DFT 분류(DFT Taxonomy)**

<696> DFT 구조는 그 테스트 방법, 그 테스트 인터페이스 및 사용될 수 있는 블록의 형태에 의해서 분류된다. 4가지의 다른 일반적 DFT 구조가 있고, 좀처럼 동일한 테스트 인터페이스를 가지지 않는다. 예를 들어, 칩의 레스트(rest)가 스캔 방법을 사용하는 동안에, 대부분의 칩은 메모리 BIST 인터페이스를 사용하는 인베드된 RAM(embedded RAM)를 가진다. 도 63에 도시된 테이블은 설계 시나리오에서 대표적인 선택사항의 리스트이다.

<697> **상위 레벨의 DFT 구조의 제조 절차(Procedure for creating a Top-level DFT Architecture)**

<698> 도 64의 플로차트는 상위 레벨의 구조 스팩을 만들고 칩-레벨 테스트 구조를 지정하는 데 사용되는 절차를 도시한다. DFT 설계는 바람직하게는 칩 위에 각 블록용 칩-레벨 테스트 로직을 지정하여야 한다. 테스트 로직을 가진 블록은 상위-레벨로 인터페이스를 받아야 한다. 테스트 로직이 없는 블록은 테스트 로직 요구를 받아야 한다. 이러한 설계 요구들을 블록 설계 태스크로 전달하고, 바람직하게는 상위-레벨 테스트 로직과 액세스 매카니즘을 만든다.

<699> 도 65에 도시된 플로차트는 블록 테스트 로직 스팩을 만드는 데 사용되는 소켓화 절차를 도시한다. 설계에서의 각 소켓에서, 각 설계 블록이 도시된 DFT 구조를 따르게 하는 테스트 칼라를 지정한다.

<700> **테스트 제널레이션 매카니즘 만들기(Creating a Test Generation Mechanism)**

<701> 테스트 제널레이션용 BBD 전략은 맨뉴얼(manual) 벡터, ATPG 또는 혼합된 것을 포함할 수 있다. 번역(translation) 및 접합(concatenation) 매카니즘은 상위 레벨 테스트 로직과 각 블록의 테스트 매카니즘에 매치되도록 한정되어야 한다. BBD에서, 테스트 개발은 하기의 두 독립적인 프로세서를 포함한다.

<702> 1. 각 가상 소켓용 블록-레벨 테스트 개발. 대부분의 경우에서, 이 프로세서는 하기의 태스크로 이루어진다.

<703> a. SAP 선언(declaration) : SAP를 behavioral 모델 인터페이서에 첨가하고 가상 소켓으로 블록을 다시 시작한다.

<704> i. 테스트 로직 삽입 : 테스트 액세스, 격리, 내부접촉 테스트 및 테스트 컨트롤 로직을 첨가하여 목표한 블록 주변에 테스트 칼라를 형성한다. 최상의 결과를 위해 합성가능한 TRL 포맷으로 테스트 칼라를 기술한다.

<705> ii. 테스트 데이터 변환 : 테스트 데이터를 SAP 포트에 연장하고 맵핑한다. 새로운 테스트 데이터 포맷을 수용하도록 블록-레벨 테스트 벤치를 지정하여야 한다. 테스트 플로를 유선형화하기 위해, 소켓당 테스트 셋업 시간을 최소화함과 동시에 다수 블록 테스트를 작동하도록 몇몇 블록에서 테스터기의 타이밍을 변경해도 된다.

<706> iii. 테스트 검증 : 테스트 로직을 검증하기 위해서 블록-레벨 테스트 벤치를 수정한다. 상기 단계의 전후에 테스트 데이터 통합을 확실히 하기 위해서 완벽한 블록-레벨 테스트 벡터 셋의 하부셋을 갖는 목표된 블록을 수정할 것이다.

<707> 2. DC 테스트 및 아날로그 테스트와 같은 칩-레벨 테스트 및 모든 테스트-합병 블록을 위한 칩-레벨 테스트 개발.

<708> 이 프로세서는 하기의 태스크를 포함한다.

<709> a. 테스트 로직 삽입 : 테스트 컨트롤러, 전용 테스트 핀, DC 테스트 로직, 아날로그 테스트 로직을 첨가하고, 모든 테스트용 테스트 클럭 및 클럭 멀티플렉서가 필요하다. 이 태스크는 가능하다면 UDL 및 테스트 합병 블록용 스캔 삽입을 포함한다.

<710> b. 테스트 제널레이션 : 테스트-합병 블록 및 UDS용 테스트 데이터를 만들고 주기적인 기능성 테스트 데이터를 잡아내기 위한 ATPG 툴을 사용한다. 목표된 제조 테스트 데이터로 폴트 커버리지 객체를 맞추는 것이 중요하다.

<711> c. 테스트 검증 : 테스트 컨트롤러를 검증하기 위해서 칩-레벨 테스트 벤치를 수정하고, DC 테스트, 아날로그 테스트, 설계에서의 모든 가상 소켓의 테스트 및 UDL 테스트를 실시한다. 이러한 테스트는 JTAG 요구와 같은 전- 또는 후-테스트 시퀀스를 필요로 할 것이다.

<712> d. 테스트 데이터 포맷 : 시뮬레이션 결과를 취하여 WGL과 같은 테스트 데이터 표현 언어로 이것을 표현한다.

<713> BBD DET 방법론적 관점에서 블록 레벨에서의 DET의 응용을 보면, 지적재산의 핵심 또는 설계 블록의 최종적인 생산물은 표준화되거나 일반적인 테스트 인터페이스 및 칩-레벨에서 재사용될 수 있는 테스트 데이터 셋을 가진 테스트-준비가 된 블록이다. 설계 블록 소켓화 설계는 각 블록을 설계하는 동안에 생산된 장치 및 대부분의 테스트 절차를 재사용하는 동안에, 칩 레벨 테스트의 적분 부분으로 설계 블록을 변화시키는 데 적용된다. BBD DET 혼합-및-매치 전략은 가장 일반적인 스캔에 기초한 테스트 방법론와 대비하여 비-합병 블록을 가려냄으로써 다른 테스트 방법 및 테스트 인터페이스를 가지는 다양한 미리-설계된 블록을 통합하도록 하는 탄력적인 접근을 제공한다. 테스트 통합 선택의 베이스에 설계 방법을 주사(scan)하는 이유는 간단히 자동 목적의 용이함 때문이다.

<714> 재사용가능한 테스트 데이터로 설계 블록을 패키지화하기 위한 많은 특정 프로세서에 관련된 블록 설계 설계는 I/O 블록의 접근가능성에 관한 어떤 가정을 고려하여 표준화되고 주문생산된 설계 블록 테스트 인터페이스에 기초한다. 그러나, 일단 끼워넣게 된다면, I/O 블록은 다른 관점에 놓이고, 보다 접근하기 어렵게 된다. 통합의 용이성을 보장하기 위해서, 테스트 인터페이스는 칩 설계 관점에서부터 직교성을 제공하기 위해서 기능성 인터페이스로부터 분리되어야 한다. BBD에서, 도 68에 도시된 바와 같은 칩 레벨에 이들을 단일화시키고, 설계 블록 인터페이스에 매칭시키고 혼합시키려 한다. 따라서, 테스트 인터페이스의 탄력성(flexibility)과 수정성(modifiability)은 테스트 액세스를 유효화하고, 매카니즘을 제어하고, 블록 레벨 테스트 버짓(budget)내에서 및 칩 I/O으로 테스트를 패키화하도록 제공되어야 한다. 당업자에 의해서 이해되는 바와 같이, 테스트 버스의 일부로서 온 칩 버스(on chip bus : OCB)의 사용이 가능하며, 이 또한 본 발명의 범위에 속한다.

<715> **비통합 블록(Non Mergeable Blocks)**

<716> DET 로직 및 테스트 벡터 검증 기능은 설계자가 보다 짧게, 생산 주기 동안 생산준비된 테스트를 보다 빨리 수행하게 한다. DFT 스캔 경로(path)는 사용불가능한 칩 및 시스템 상태로의 접근을 가능하게 한다. 메모리 BIST는 다른 끼워넣은 메모리 폴트 클레스(memory fault classes)를 포함하는 십진(algorithmic) 테스트 벡터를 사용한다. 로직 BIST는 테스트 액세스 및 테스트 데이터 병목현상을 줄이도록 설계된 스캔의 무작위 테스트 가능 구조를 이용한다. 그러나, 각각의 미리-설계된 블록은 수많은 이유로 통합되지 않을 것이다. 일반적으로,

<717> 비통합 블록은 내부 테스트 접근가능성(예를 들어, 게이트화된 클럭, 렌치-베이스된 데이터 경로)의 부족 또는 폴트 커버리지(예를 들어, 비동기)의 부족 때문에 일반적인 테스트 방법과 양립되지 않는 합성가능한 RTL 소프트 블록이다.

<718> 비통합 블록은 스캔 방법론(즉, 동기화), 스캔 스타일(즉, 먹스-스캔(mux-scan), 클럭 스캔, LSSD)과 같은 일반적 테스트 방법과 양립되지 않는 게이트-레벨 소프트 블록이다.

<719> 비통합 블록은 일반적으로 배열되어 있는 컴파일된 블록이다. 예를 들어, 끼워넣은 RAMs, ROMs, DRAM, FLASH 등은 조합 로직으로서 동일한 폴트 모델을 갖지 않는다. 이러한 블록은 큰 십진(algorithmic) 테스트 패턴을 요구한다.

<720> 비통합 블록은 특정한 테스트 방법으로 만들어지나, 테스트 통합용 하부조직(infrastructure)을 지니지 않는 하드 블록이다. 일반적으로, 이러한 블록은 바람직하게는 특정 테스트 인터페이스를 갖거나 또는 갖지 않는 특정 블록 레벨 테스트 데이터로 전달된다.

<721> 비통합 블록은 특정 테스트 방법으로나 그렇지 않은 방법으로 제조되지만, 통합을 위한 내부구조를 지니지 않는 레그시 블록(legacy block)이다. 일반적으로, 이러한 블록은 알려지지 않은 시퀀스를 피하도록 변경되지 않을 것이다.

<722> **테스트 칼라(test collars)**

<723> 소켓화된 설계 블록은 SAP 스팩을 갖는 소켓을 기술하는 새로운 모듈을 제조함으로써, 최초의 설계 블록을 기술함으로써, 도 66의 플로차트에 도시된 바와 같이 이들 사이에 테스트 로직을 삽입함으로써 모델화될 수 있다. 소켓화된 설계 블록은 처음에 설계 블록 기능 인터페이스를 저장하고, 테스트 액세스, 테스트 격리를 가하고, 주변 테스트 구조는 칩의 설계 동안에 기본적인 테스트 인터페이스(예를 들어, TAR 스캔, BSR 또는 직접적인

먹스)을 제공한다.

<724> 그 결과는 각 관련 테스트 I/O 포트용 코멘트로서 가해진 테스트 특성을 갖는 SAP이다. 각 비-통합 블록은 테스트 칼라에 의해서 감싸여져, 블록 베이스에 의해 블록에 테스트 셋업, 테스트 실행 및 테스트 포스트 프로세서를 실행시키는 테스트 액세스, 격리, 및 내부접촉 테스트를 가한다. 출력은 다음을 포함하는 소캔화된 설계 블록이다.

<725> 1. 테스트 액세스 및 컨트롤(예를 들어, 테스트 모드, 활성화 및 비활성화)

<726> 2. 테스트 프로토콜(예를 들어, 기능성, 먹스-스캔, BIST, 진단성)

<727> 3. 테스트 데이터(예를 들어, 테스트 언어, 벡터 크기, 폴트 커버리지)

<728> 4. 테스트 포맷(예를 들어, 테스터기 스팩, 타임셋, 타임스피드)

<729> 5. 테스트 응응시간(예를 들어, 비 테스트 셋업 타입).

<730> **테스트가능성의 부가(Adding Testability)**

<731> 재사용가능한 테스트 데이터에 부수되지 않는 각 비통합 블록에서, 설계 페이즈(phase)는 테스트 구조를 삽입하고 전체 면적 및 타이밍 코스트(timing cost)를 평가함으로써 테스트 인터페이스, 테스트 방법, 테스트 데이터 포맷, 예견되는 폴트 커버리지 및 테스트 버짓을 지정할 수 있다. 이 평가는 각 블록에 대한 테스트가능성을 부가하는 제한이 된다.

합성가능한 RTL 소프트 블록(Synthesizable RTL Soft Blocks)

<733> 미리-설계된 블록은 스캔에 기초한 테스트에 적용되지 않는 합성가능한 소프트 블록이다. 따라서 폴트 커버리지가 문제된다. 예를 들어, 스캔 설계 규칙의 체크는 스캔 잘못을 가려내기 위해서 RTL 또는 게이트 레벨에서 이루어진다. 스캔 체인 또는 테스트 포인트는 쉽게 모델로 삽입되지 않기 때문에, 도 67의 플로차트에 도시된 바와 같이, 연속적인 ATPG는 기능적인 테스트 벡터와 공유하도록 사용될 수 있다. 이러한 타입의 설계에서의 폴트 커버리지는 예측하기 어렵고, 폴트 시뮬레이션은 바람직하게는 설계 페이즈 동안에 이러한 블록의 재사용 가능성 표준을 설정하는 데 사용되어야 한다. TBA 기초 베이스 칼라는 최상의 테스트 인터페이스이지만, BSR 기초 베이스 칼라는 블록용 테스트 버짓이 허용될 때 고려될 수 있다.

<734> **검증(Verification)**

<735> DFT에서 설계 검증으로 옮겨 보면, 본 발명의 시스템 및 검증방법의 주로 목적은 완전한 설계(최종적인 탭 아웃에서)이 프런트-앤드 허용(front-end acceptance)의 일부로서 제공되고 기능성 스팩 및 칩 테스트 벤치에 지정된 고객의 기능성 요구와 일치하도록 보장하는 것이다. 두 번째 목적은 가능한 짧은 시간동안에 상기 첫 번째 목적을 달성하는 것이다.

<736> 설계 테스트 설계에서 고객에게 제공된 칩 테스트 벤치가 고객이 요구한 기능성의 완전한 테스트를 형성하는 것이 본 발명의 적절한 기능에 특히 필수적이다. 이러한 가정은 바람직하게는 프런트 엔드 허용 동안에 강조된다. 따라서, BBD 설계 플로(flow)는 기능성 스팩 모델에서 작동할 때 칩 테스트 벤치의 등급과 합쳐져서, 칩 테스트 벤치의 측정을 제공한다.

<737> 통합적인 방법으로 기능성 스팩 및 칩 테스트 벤치를 이용하는 발명적 접근은 이 두가지가 확실히 일치되도록 하는 것이다. 이 후에, 칩 설계, 칩 어셈블리 및 블록 설계를 통해서 세밀하게 한정되고 부가되기 때문에, 설계는 기능성이 최초의 기능성 스팩과 일치시키되도록 하는 것을 보장하기 위해 칩 테스트 벤치를 통해서 재검증된다. 하기에서 기술되는 바와 같이, 칩 테스트 벤치로부터 추출된 블록 테스트 벤치를 갖는 각 블록 레벨에서 또는 완전한 칩 레벨에서 보다 세밀한 검증이 형성될 것이다.

<738> 설계 프로세서에서 빠르게 계속적으로 전달되지 않는다면, 반복적으로 다시 설계하여야 하는, 동일한 버스를 따라 연결된 다양한 블록의 내부접촉 및 버스 로직이 해결될 충분한 시간을 가질 수 있음을 경험으로부터 알 수 있다. 이러한 이유에서, 특정한 주의가 설계 주기 동안에 버스 기능성을 유효하게 된다. 버스 및 관련 로직은

설계의 레스트과 관련없이 상기에서 기술된 바와 같이 버스 컴파일러 테스트 벤치를 사용하여 초기 단계에서 동일시되고, 검증된다. 그러나, 본 발명의 바람직한 검증 플로는 빠르게 전환되는 다양한 설계를 다룰 수 있을 정도로 매우 유동적이다. 예를 들어, 설계가 간단한 버스를 갖거나 설계자가 버스에 부착된 블록에 대해서 중요한 경험을 가지고 있다면, 버스 컴플라이언스 테스트의 전부 또는 일부는 연기될 것이다. 이와 유사하게 블록 전체 또는 일부가 간단하거나 이전의 설계로부터 재사용된다면, 각 블록 검증의 일부는 빼고, 칩 레벨 검증 단계에 도달할 때까지 검증이 이루어진다.

<739> 특정 설계에 따른 세밀한 플로는 FEA 프로세서의 일부로서 설정되어야 한다. 도 12-15는 본 발명에 따른 기능성 검증 동안에 수행될 태스크의 일반적인 플로를 제공한다. 도 69-73의 칩 테스트 벤치에 대한 크로스 문헌에서 이러한 도면은 세세하게 기술될 것이다. 도 12-15에서 큰 화살표는 태스크 플로를, 보다 작은 화살표는 태스크 입력을, 대시 형상의 화살표는 선택적인 바이패스 경로를 의미한다.

<740> 도 12에서, FEA를 완성한 후에, 상기에서 기술된 바와 같이, 본 발명의 방법은 칩 테스트 벤치 검증 단계(8210)로 계속된다. 여기서, 칩-레벨 기능 모델은 도 69의 칩 테스트 벤치(8310)로 실행된다. 모델 및 테스트 벤치는 고객에게 제공되며, 검증의 목적은 테스트 벤치 및 기능 모델이 일치하도록 보증하는 것이다. 모델은 다른 적합한 언어도 가능하지만 바람직하게는 베릴로그(Verilog), VHDL 및 실행 C 코드일 것이다. 칩 테스트 벤치(8310)는 모델에 호환성을 가진 파일내에 있을 것이다. 모델 및 테스트 벤치사이의 잘못된 매치는 고객에게 다시 되돌려질 것이고, 모델 또는 테스트 벤치 중 어느 하나는 내부적 일관성을 얻도록 수정될 것이다.

<741> 다음으로, 칩 테스트 벤치는 기능 모델이 작동하는 동안 변경된다. 이러한 변경은 하나 이상의 하기의 특성을 측정하므로써 테스트 벤치의 우수함과 커버리지 미터를 제공한다. 그 특성은 상태 커버리지, 토글(toggle) 커버리지, FSM arc 커버리지, 방문상태(visited state coverage) 커버리지, pair arc 커버리지, 경로/가지 커버리지 및/또는 표현 커버리지이다. 이 커버리지 미터는 고객에게로 되돌려진다. 설계가 부적적하게 테스트되거나 설계가 여분의 기능성을 포함할 때, 커버리지 미터는 불완전하게 테스트되는 설계의 가장 높은 영역일 것이다. 어떤 경우에서, 고객은 커버리지 미터를 향상시키기 위해서 테스트 벤치 또는 모델을 수정할지 여부를 선택할 것이다. 따라서 여기서 기술된 BBD 설계 방법용 프로젝트 시작 시간을 재설정할 것이다.

<742> 일단 칩 테스트 벤치가 기능 모델과 일치되는 것이 보장된다면, 도 69에서의 칩의 새로운 모습(8312, 도69)는 이러한 블록 사이에서 한정된 글루 로직을 갖는 각 블록용 블록 기능 모델을 혼합함으로써 단계 8212에서 제조된다. 블록 기능 모델(8312)는 상기에서 기술된 바와 같이 FEA 동안에 디핑(dipping) 프로세서를 통해서 고객에게 제공되거나 제조된다. 상기에서 기술된 바와 같이, 칩 설계 동안에 글루 로직 모델은 또한 지정된다.

<743> 다시 도 12로 돌아와서, 칩 레벨 구조의 검증 단계(8214)는 칩 테스트 벤치로 칩의 블록 기능 모델을 모의 실험하는 단계를 포함한다. 어떤 불일치는 하나이상의 블록 기능 모델(832)이나 글루 로직을 수정함으로써, 또는 모의 실험을 다시 함으로써 해결된다. 이 단계는 블록 기능 모델이 칩 기능 모델과 일치할 것을 보장한다.

<744> 도 13 및 도 14로 다시 돌아와서, 버스 검증 플로의 목적은 칩내의 버스 로직이 올바르게 작동하고 다른 버스 엘리먼트와의 내부접촉이 버스 프로토콜 에러를 야기시키지 않도록 보장하는 것이다. 따라서, 컴플라이언스 벡터는 버스 설계를 위해서 제조된다. 이러한 벡터는 고객 또는 블록 설계 공급기에 의해 제공된 컴플라이언스 테스트 슈트(suit)에 기초한다. 벡터는 설계의 특정 버스 형태에 대응하도록 조정되어야 할 것이다. 컴플라이언스 벡터가 제공되지 않을 때, 이것들은 설계 팀에 의해서 기록되어야 할 것이며, 바람직하게는 이러한 방법으로 이들은 버스에 부착된 다양한 블록의 내부접촉을 실행하고, 모든 주변 조건을 실행하고, 버스 에러가 완벽하게 조절되는 지를 검증한다.

<745> 도 13의 8218 단계는 버스 기능성에 대한 검증을 제공한다. 버스 컴플라이언스 벡터는 상기에서 기술된 칩 설계 단계로부터 제공된 버스의 정확한 주기적 모델에 반하여 모의실험된다. 컴플라언스 벡터 셋(도시되지 않음)을 수정함으로써, 또는 도 70에 도시된 하나 이상의 버스 로직 엘리먼트(8512)를 수정함으로써 에러는 해결되어야 한다. 이 단계는 컴플라이언스 테스트 도구(suite)가 버스 로직 모델에서 성공적으로 수행될 때까지 반복된다.

<746> 도 14에서, 버스 로직 모델 및 테스트 벤치 제조단계(8610 내지 8614)가 도시되어 있다. 버스 블록 모델 검증 단계(8614) 뿐만아니라 버스 블록 모델 제조단계(8610) 및 테스트 벤치 제널레이션 추출단계(8612)의 목적은 높은 레벨의 행동 모델 및 설계 내의 각 블록용 관련 테스트 벤치를 제조하는 것이다. 이는 블록 설계자에게 전달되어 각 블록을 위해 목표된 기능성을 한정한다.

<747> 도 70에서 각 블록용 버스 블록 모델(8510)을 제조하는 단계는 블록에서 정확한 버스 로직 모델로, 기능적으로

정확하게 주기으로 접근하는 블록 기능 모델(8312)을 결합하는 단계를 포함한다. 이 버스 로직은 상기의 칩 설계 및 검증으로부터 제공된 버스 글루 로직 모델로부터 추출된다. 버스 기능 모델에 대한 일부의 수정은 인터페이서가 정렬되도록 하는 데 요구될 것이다.

<748> 버스 블록 모델은 모든 버스 블록 모델을 결합하는 칩 모델을 모음으로써 검증된다. 칩 모델은 칩 테스트 벤치로 이것을 모의실험함으로서 검증된다. 칩 테스트 벤치가 이전에 주기적 접근 모델에서 검증되는 동안에, 칩의 이 동작 블록 모델은 정확히 주기적인 동작을 가지고, 칩 테스트 벤치의 일부 개선사항은 블록 모델이 패스하는 것을 얻도록 하는 데 요구될 것이다. 몇몇의 경우에 있어서, 블록 기능 모델 및 버스 로직에서의 잘못된 매치 때문에 에러가 발생할 것인데 이 때에 모델은 에러를 바로잡도록 수정될 것이다. 일단 칩 테스트 벤치가 이 칩 모델에서 성공적으로 작동한다면, 각 버스 블록 모델은 정밀한 실행을 위해 블록 설계자에게 전달될 것이다.

<749> 도 14의 8612 단계에서, 블록 테스트 벤치가 추출된다. 일단 칩 테스트 벤치가 칩 레벨 버스 블록 모델(8710)에서 성공적으로 실행된다면, 도 71에 도시된 바와 같이, 프로브(probe)는 각 블록의 인터페이스에 설정될 수 있고, 모델에서 실행되는 동안에 블록 테스트 벤치는 칩 테스트 벤치로부터 추출될 수 있다. 이러한 블록 테스트 벤치는 실행을 통해서 수행되는 동안에 블록의 유효화를 위해 블록 설계자에게 전달된다.

<750> 도 15에서 도시된 로직 검증 플로에서, 로직 검증 태스크의 목적은 각 블록이 설계의 실행 페이지(RTL에서 프리- 또는 포스트-레이아웃 넷리스트)를 통해서 수행되는 동안에 기능화할 것을 보장하는 것이다. 또한 어셈블리 칩이 계속적으로 요구되는 기능성을 제공하는지가 테스트된다.

<751> 검증은 등가 체크를 수행하는 기능 시뮬레이션을 동적으로 이용하여 또는 공식적 검증 툴을 동적으로 이용하여 수행될 수 있다. 동적인 검증은 본 발명의 BBD 방법론 흐름에서 다른 곳에 요구되고 설명되는 시뮬레이션 툴을 요구한다. 동적인 검증은 다른 곳에서 이용되는 벡터 세트를 이용하며 대략적인 사이클에서 본질적으로 정확한 사이클로의 테스트 슈우트의 이동에 도움을 준다. 정적인 검증은 새로운 툴을 요구한다. 그러나, 정적인 검증은 일반적으로 시뮬레이션보다 빠르게 동작하며 테스트 브랜치가 설계 기능을 수행하는 정도의 등가성만을 제공하는 시뮬레이션에 비하여 "완전한" 등가 체크를 제공한다.

<752> 다음, 개별 RTL 블록 모델이 단계(8710)에서 검증되고, 여기서 블록 설계자에 의하여 생성된 RTL 시뮬레이션 모델은 칩 테스트 브랜치에 대하여 검증된다. 이는 칩 레벨 행동 모델의 대응하는 행동 모델로 블록 RTL 모델을 스웨핑하고 완전 칩 테스트 브랜치를 이용하여 칩의 혼합 모드 시뮬레이션을 수행함으로써 이루어진다. 선택적으로, 개별 블록 RTL 모델은 추출된 블록 테스트 브랜치로 시뮬레이션될 수 있다. 두 경우, 근접 사이클 모델에서 정확한 사이클 모델로의 전이 때문에 미스매칭이 발생될 수 있다. 이들 미스매칭은 테스트 선택 플래그를 변형함으로써 해결될 것이다. 미스매칭이 미싱 기능 또는 부정확한 기능에 의하여 발생될 경우, RTL 모델은 에러를 보정하기 위하여 변형되어야 한다.

<753> 단계(8712)에서, RTL 블록 모델은 칩 레벨에서 검증된다. 각각의 블록에 대한 RTL 시뮬레이션 모델은 칩 레벨 RTL 모델을 형성하기 위하여 결합된다. 이 모델은 칩 테스트 브랜치로 시뮬레이션함으로써 검증된다. 또한, 근접 사이클 모델에서 정확한 사이클 모델로의 전이 때문에 일부 에러가 발생될 수 있다. 이들 에러는 칩 테스트 브랜치를 변형함으로써 해결될 것이다. 다른 기능 에러는 하나 이상의 블록 레벨 RTL 모델을 변형함으로써 해결되어야 한다.

<754> 단계(8714)에서, 개별 프리-레이아웃 블록 네트리스트가 검증된다. 각각의 블록에 대한 포스트 통합 네트리스트 시뮬레이션 모델은 상기 블록에 대한 RTL 모델에 대한 것이다.

<755> 단계(8716)에서, 다이나믹 및 정적인 칩 레벨 프리-레이아웃 블록 네트리스트가 검증된다. 다이나믹 검증은 칩 레벨 행동 모델의 대응하는 행동 모델로 블록 레벨 포스트 통합 네트리스트를 스웨핑하고 완전 칩 테스트 브랜치를 이용하여 칩의 혼합 모드 시뮬레이션을 수행함으로써 이루어진다. 선택적으로, 개별 블록 레벨 포스트 통합 네트리스트는 블록 테스트 브랜치로 시뮬레이션될 수 있다. 두 경우, 근접 사이클 모델에서 인트라 사이클 타이밍을 가진 모델로의 전이 때문에 미스매칭이 발생될 수 있다. 이들 미스매칭은 테스트 브랜치내의 타이밍 스트로브를 변형함으로써 해결될 것이다. 정적 검증은 포스트 통합 네트리스트 및 각각의 블록에 대한 RTL 모델에 대한 등가 체킹 툴을 동작시킴으로써 수행된다. RTL 모델을 매칭시키기 위하여 포스트 통합 네트리스트를 변형함으로써 미스매칭이 해결될 것이다.

<756> 각각의 블록에 대한 포스트 통합 네트리스트는 칩 포스트 통합 네트리스트를 형성하도록 결합된다. 이들 칩 레벨 네트리스트는 시뮬레이션 또는 정적인 정상 등가 체킹 툴을 통하여 검증된다. 다이나믹 검증은 칩 테스트

브랜치로 칩 포스트 통합 네트리스트를 시뮬레이션함으로써 이루어진다. 정적인 칩 레벨 프리-레이아웃 검증은 칩 포스트 통합 네트리스트 및 각각의 칩에 대한 칩 RTL 모델에 대한 등가 체킹 툴을 수행함으로써 이루어진다. 미스매칭은 RTL 모델을 매칭시키기 위하여 포스트 통합 네트리스트을 변형함으로써 해결될 것이다.

<757> 단계(8718)에서, 개별 포스트-레이아웃 블록 네트리스트이 검증된다. 이 단계는 단계(8714)의 반복이지만, 포스트-레이아웃 네트리스트는 프리-레이아웃 네트리스트에 대체된다. 네트리스트 레벨에서 이들 두 개의 모델사이의 차이점은 단지 레이아웃 설계의 타이밍 목적을 달성하기 위하여 버퍼 및 드라이브 강도의 변형일 뿐이다. 발생되는 에러는 버퍼의 부정확한 추가 또는 삭제로 한정되어야 한다. 블록 테스트 브랜치의 타이밍은 만약 포스트-레이아웃 타이밍 변경이 타이밍 스트로브에 대한 신호를 이동시킬 경우 변형될 필요가 있다.

<758> 이러한 검증은 정적으로 또는 다이내믹하게 수행될 수 있다. 다이나믹 검증은 칩 레벨 RTL 모델의 대응하는 블록 RTL 모델로 블록 레벨 포스트 레이아웃 네트리스트를 스웨핑하고 완전 칩 테스트 브랜치를 이용하여 칩의 혼합 모드 시뮬레이션을 수행함으로써 이루어진다. 선택적으로, 개별 블록 레벨 포스트 레이아웃 네트리스트는 블록 테스트 브랜치로 시뮬레이션될 수 있다. 정적인 검증은 포스트 레이아웃 네트리스트 및 각각의 블록에 대한 RTL 모델에 대한 등가 체킹 툴을 동작시킴으로써 수행된다. RTL 모델을 매칭시키기 위하여 포스트 통합 네트리스트를 변형함으로써 미스매칭이 해결될 것이다.

<759> 칩 레벨 포스트-레이아웃 네트리스트의 검증은 단계(8720)에서 달성되는데, 이는 단계(8716)의 반복이지만, 포스트-레이아웃 칩 레벨 네트리스트는 프리-레이아웃 네트리스트에 대하여 대체된다. 네트리스트 레벨에서 이들 두 개의 모델사이의 차이점은 단지 레이아웃 설계의 타이밍 목적을 달성하기 위하여 버퍼 및 드라이브 강도의 변형일 뿐이다. 발생되는 에러는 버퍼의 부정확한 추가 또는 삭제로 한정되어야 한다. 다이나믹 검증은 칩 테스트 브랜치로 칩 포스트-레이아웃 네트리스트를 시뮬레이션함으로써 수행된다. 정적 검증은 칩 포스트-레이아웃 네트리스트 및 칩 RTL 모델에 대한 등가 체킹 툴을 수행함으로서 이루어진다. RTL 모델을 매칭시키기 위하여 포스트-레이아웃 네트리스트를 변형함으로써 미스매칭이 해결될 것이다.

<760> 마지막으로, 물리적 검증은 도 72 및 73에 도시된 것처럼 수행되며, 여기서 블록 및 칩 테이프는 본 발명 분야의 당업자에 의하여 이해되는 방식으로 검증된다. 물리적 검증 태스크의 목적은 블록 설계 및 칩 어셈블리 설계 단계를 통하여 발생된 GDSII가 기능적으로 정확한지 목적 기술에 대한 설계 룰을 위반하고 있지 않은지를 검증하는 것이다.

<761> 블록 설계 프로세스에 의하여 형성된 각각의 블록에 대한 GDSII는 목적 기술에 대한 DRC를 수행함으로써 검증된다. 에러 및 경고는 해결을 위하여 다시 블록 설계자에게 피드백된다. LVS가 또한 블록 GDSII 파일 및 상기 블록에 대한 포스트-레이아웃 네트리스트사이에서 수행된다. 에러와 경고는 해결을 위하여 블록 설계자에게 피드백된다.

<762> 칩 어셈블리 프로세스에 의하여 형성된 완전한 칩에 대한 GDSII는 목적 기술에 대한 DRC를 수행함으로써 검증된다. 에러 및 경고는 해결을 위하여 다시 칩 어셈블리 설계자에게 피드백된다. LVS가 또한 칩 GDSII 파일 및 상기 칩에 대한 포스트-레이아웃 네트리스트사이에서 수행된다. 에러와 경고는 해결을 위하여 칩 어셈블리 설계자에게 피드백된다.

<763> 본 발명이 도면 및 상기 설명에서 도시되고 설명되었지만, 본 발명은 본 발명의 사상내의 선택적인 실시예를 통하여 수행될 수 있다. 따라서, 본 발명의 범위는 실시예의 설명에 한정되는 것이 아니라 첨부된 청구범위에 따른다.

도면의 간단한 설명

<44> 도 1은 본 발명에 따라 블록기반 설계 방법에 기초한 설계 프로세스를 도시하는 흐름도이다.

<45> 도 2는 본 발명에 따라 프런트 엔드 액세스 단계들을 도시하는 흐름도이다.

<46> 도 3은 본 발명에 따른 클록 플래닝 모듈을 도시한다.

<47> 도 4는 본 발명에 따른 버스 식별 및 플래닝 모듈을 도시한다.

<48> 도 5는 본 발명에 따른 파워 플래닝 모듈을 도시한다.

<49> 도 6은 본 발명에 따른 I/O 및 아날로그/혼합 신호 요건들을 도시한다.

<50> 도 7은 본 발명에 따른 테스트 플래닝 모듈을 도시한다.

<51> 도 8은 본 발명에 따른 타이밍 및 플로어 플래닝 모듈을 도시한다.

<52> 도 9는 본 발명에 따른 블록 설계의 메타 흐름을 도시한다.

<53> 도 10은 본 발명에 따른 칩 어셈블리의 데이터 흐름을 도시한다.

<54> 도 11은 본 발명에 따른 칩 어셈블리의 태스크 흐름을 도시한다.

<55> 도 12, 13, 14 및 15는 본 발명에 따른 기능적 검증 흐름을 도시한다.

<56> 도 16은 본 발명에 따라 다수의 미리 설계된 회로 블록들을 사용하여 회로 설계의 실행가능성을 평가하는 방법을 도시한다.

<57> 도 17은 본 발명에 따라 도 2에 도시된 방법을 사용한 실행가능성 평가 결과를 도시한다.

<58> 도 18은 본 발명에 따라 다수의 미리 설계된 회로 블록들을 사용하여 회로 설계의 실행가능성을 평가하는 방법을 도시한다.

<59> 도 19는 본 발명에 따라 도 18에 도시된 방법을 사용한 실행가능성 평가 결과를 도시한다.

<60> 도 20은 본 발명에 따른 프런트 엔드 허용("FEA") 프로세스를 도시한다.

<61> 도 21은 본 발명에 따른 세분화로세스를 도시한다.

<62> 도 22는 본 발명에 따른 전형적인 평가 정정 곡선을 도시한다.

<63> 도 23은 본 발명에 따라 FEA를 유효화하는 프로세스를 도시한다.

<64> 도 24는 본 발명에 따라 FEA 설계 특성 세분화로세스를 사용한 정제된 평가 정정 곡선을 도시한다.

<65> 도 25는 본 발명에 따른 FEA 데이터 추출 프로세스를 도시한다.

<66> 도 26는 본 발명에 따른 블록 평가 세분화의 필요성을 확인하는 프로세스를 도시한다.

<67> 도 27은 본 발명에 따른 FEA 평가 축 미터를 도시한다.

<68> 도 28은 본 발명에 따른 분류 컬랩스(collapse) 곡선을 도시한다.

<69> 도 29는 글루 로직이 최적 설계 블록 배치를 방해하는 회로 설계의 다수의 설계 블록들을 도시한다.

<70> 도 30은 본 발명에 따른 글루 로직 분배의 제 1 형태를 도시한다.

<71> 도 31은 본 발명에 따른 글루 로직 분배의 제 2 및 제 3 형태를 도시한다.

<72> 도 32는 본 발명에 따라 회로 블록을 칼라(collar)로 구현하는 칼라링 프로세스를 도시한다.

<73> 도 33은 본 발명에 따른 설계에 사용되는 하나의 블록에 대한 개괄의 완전한 세트를 형성하는 것을 도시한다.

<74> 도 34는 본 발명에 따른 칼라링 프로세스를 도시하는 흐름도이다.

<75> 도 35는 본 발명에 따라 두개 층들을 갖는 칼라를 도시한다.

<76> 도 36은 본 발명에 따라 칼라와 회로 블록간의 논리도를 도시한다.

<77> 도 37은 본 발명에 따라 칼라와 회로 블록간의 물리도를 도시한다.

<78> 도 38은 본 발명의 칼라링 프로세스를 사용하지 않은 시스템 설계를 도시한다.

<79> 도 39는 본 발명의 칼라링 프로세스를 사용한 시스템 설계를 도시한다.

<80> 도 40은 본 발명에 따라 도 34의 칼라링 프로세스의 단계들을 수행하기 위한 컴퓨터 시스템을 도시한다.

<81> 도 41은 본 발명의 버스 식별 및 플래닝 설계를 포함하는 일련의 단계들을 도시한다.

<82> 도 42는 본 발명의 방법에 따라 형성된 행동 모델의 상호접속 섹션의 내부 구조를 도시한다.

<83> 도 43-47 및 도 49-56은 본 발명의 시스템 및 방법을 사용하여 형성된 버스 수정안을 통해 개선된 지연 시간들을 나타내는 표들이다.

<84> 도 48은 본 발명의 방법 및 시스템에 사용되는 버스 브리지를 도시한다.

<85> 도 57은 본 발명의 방법 및 시스템에 사용된 버스 브리지를 도시한다.

<86> 도 58은 본 발명의 방법 및 시스템에 사용된 FIFO를 포함하는 버스 브리지를 도시한다.

<87> 도 59는 다양한 버스 형태에 대한 버스 활용 및 대기시간 특성들을 도시하는 표이다.

<88> 도 60은 전형적인 지속 체크 진리표를 도시한다.

<89> 도 61은 본 발명의 방법을 사용하여 DFT 투시도로부터 칩의 상위 레벨 계층을 도시한다.

<90> 도 62는 기능적 블록들 및 소켓 액세스 포트("SAPs")들로 구성된 설계를 도시한다.

<91> 도 63은 다양한 설계 구조들에 대한 적합한 테스트 방법들을 도시하는 표이다.

<92> 도 64는 본 발명의 방법 및 시스템에 대한 상위 레벨 구조 명세 절차를 도시하는 흐름도이다.

<93> 도 65는 본 발명의 방법 및 시스템의 소켓화(socketization) 절차를 도시한다.

<94> 도 66은 본 발명의 방법 및 시스템의 블록 레벨 테스트 개발 절차를 도시한다.

<95> 도 67은 본 발명의 방법 및 시스템의 칩 레벨 테스트 개발 절차를 도시한다.

<96> 도 68은 본 발명의 방법 및 시스템에 따른 칩 어셈블리를 플래닝하는 테스트 흐름이다.

<97> 도 69는 본 발명의 프런트엔드 허용 검증 툴의 설계자측 도면이다.

<98> 도 70은 블록 설계에 대한 칩 플래닝으로부터 이동하는 설계자측 도면을 도시한다.

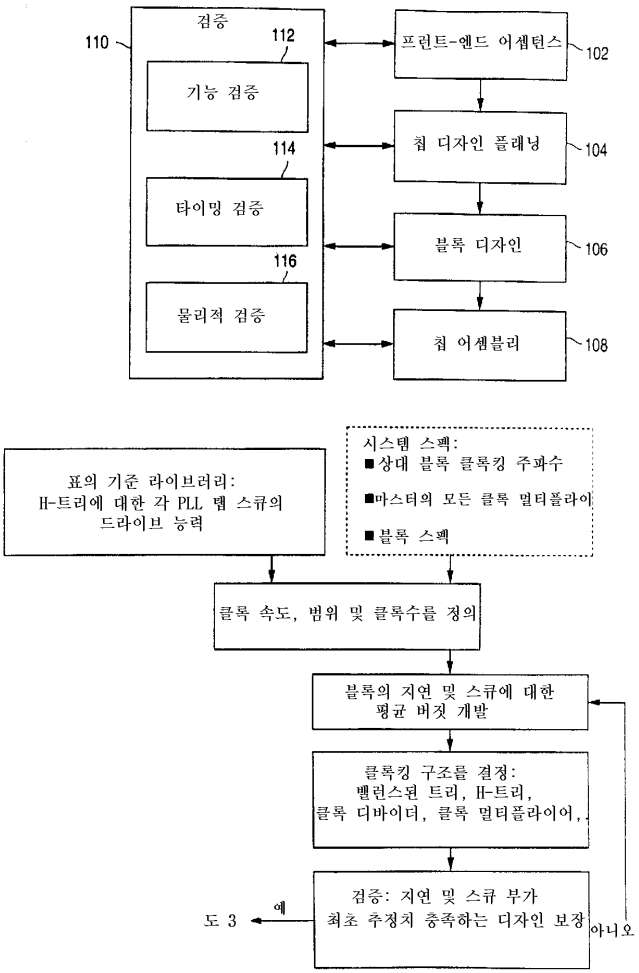
<99> 도 71은 본 발명의 방법 및 시스템의 버스 블록 모델 및 테스트 벤치 생성을 전개하는 설계자측 도면을 도시한다.

<100> 도 72는 블록 테스트 벤치 및 칩 테스트 벤치의 설계자측 도면을 도시한다.

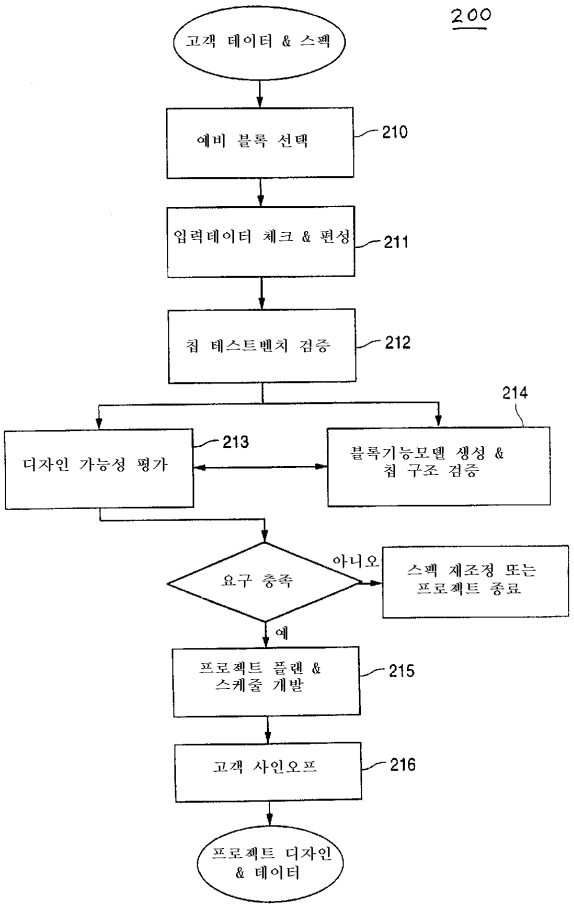
<101> 도 73은 블록 및 칩 로직 검증 모델들의 설계자측 도면이다.

도면

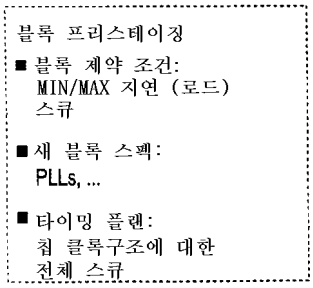
도면1


검증
110
112
기능 검증
114
타이밍 검증
116
물리적 검증
프린트-엔드 어셈턴스
102
칩 디자인 플래닝
104
블록 디자인
106
칩 어셈블리
108
표의 기준 라이브러리:
H-트리에 대한 각 PLL 탭 스큐의
드라이브 능력
시스템 스펙:
■상대 블록 클록킹 주파수
■마스터의 모든 클록 멀티플라이
■블록 스펙
클록 속도, 범위 및 클록수를 정의
블록의 지연 및 스큐에 대한
평균 버짓 개발
클록킹 구조를 결정:
밸런스된 트리, H-트리,
클록 디바이더, 클록 멀티플라이어,
검증: 지연 및 스큐 부가
최초 추정치 충족하는 디자인 보장
예
도 3
아니오

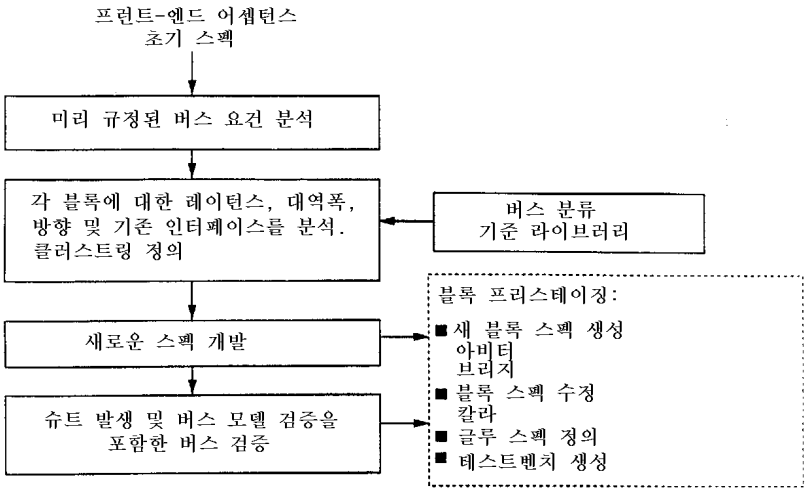
도면2


200
고객 데이터 & 스펙
예비 블록 선택
210
입력데이터 체크 & 편성
211
칩 테스트벤치 검증
212
214
디자인 가능성 평가
213
블록기능모델 생성 & 칩 구조 검증
요구 충족
아니오
스펙 재조정 또는 프로젝트 종료
예
프로젝트 플랜 & 스케줄 개발
215
고객 사인오프
216
프로젝트 디자인 & 데이터

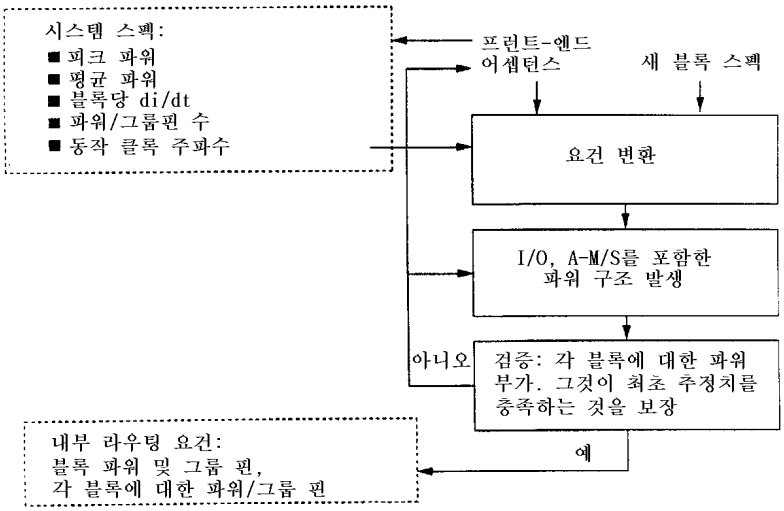
도면3


블록 프리스테이징
■ 블록 제약 조건:
MIN/MAX 지연 (로드)
스큐
■ 새 블록 스펙:
PLLs, ...
■ 타이밍 플랜:
칩 클록구조에 대한
전체 스큐

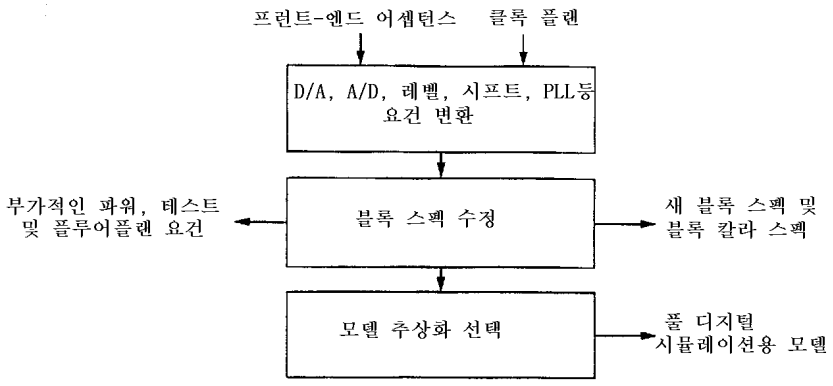
도면4


프런트-엔드 어셈턴스
초기 스펙
미리 규정된 버스 요건 분석
각 블록에 대한 레이턴스, 대역폭,
방향 및 기존 인터페이스를 분석.
클러스트링 정의
버스 분류
기준 라이브러리
새로운 스펙 개발
슈트 발생 및 버스 모델 검증을
포함한 버스 검증
블록 프리스테이징:
■ 새 블록 스펙 생성
아비터
브리지
■ 블록 스펙 수정
칼라
■ 글루 스펙 정의
■ 테스트벤치 생성

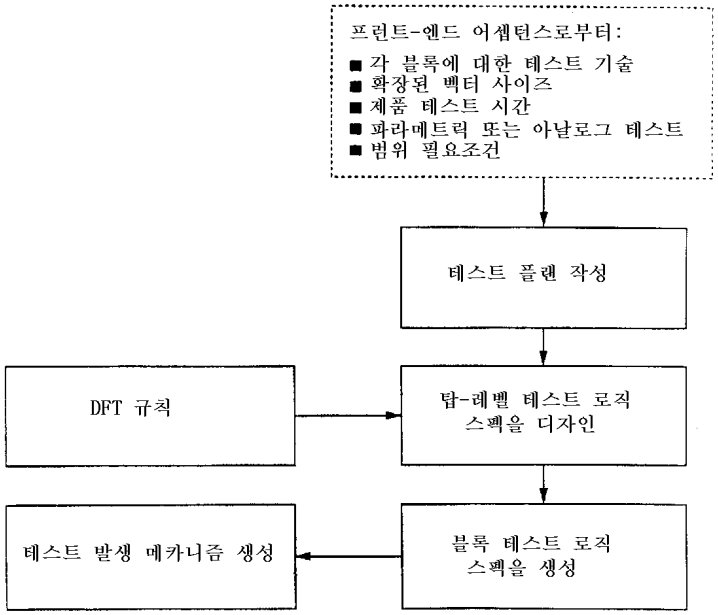
도면5


시스템 스펙:
■ 피크 파워
■ 평균 파워
■ 블록당 di/dt
■ 파워/그룹핀 수
■ 동작 클록 주파수
프런트-엔드
어셈턴스
새 블록 스펙
요건 변환
I/O, A-M/S를 포함한
파워 구조 발생
아니오
검증: 각 블록에 대한 파워
부가. 그것이 최초 추정치를
충족하는 것을 보장
예
내부 라우팅 요건:
블록 파워 및 그룹 핀,
각 블록에 대한 파워/그룹 핀

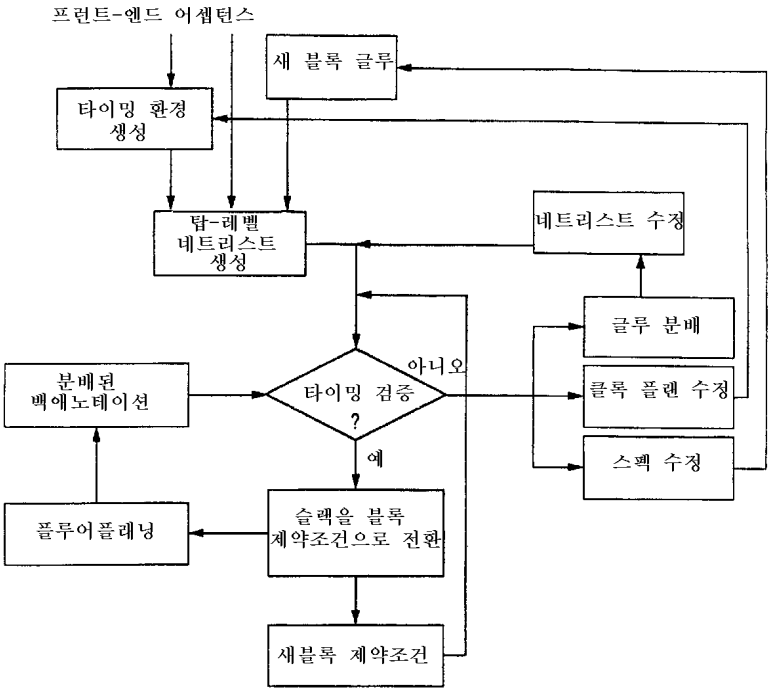
도면6


프런트-엔드 어셈턴스
클록 플랜
D/A, A/D, 레벨, 시프트, PLL등
요건 변환
부가적인 파워, 테스트
및 플루어플랜 요건
블록 스펙 수정
새 블록 스펙 및
블록 칼라 스펙
모델 추상화 선택
풀 디지털
시뮬레이션용 모델

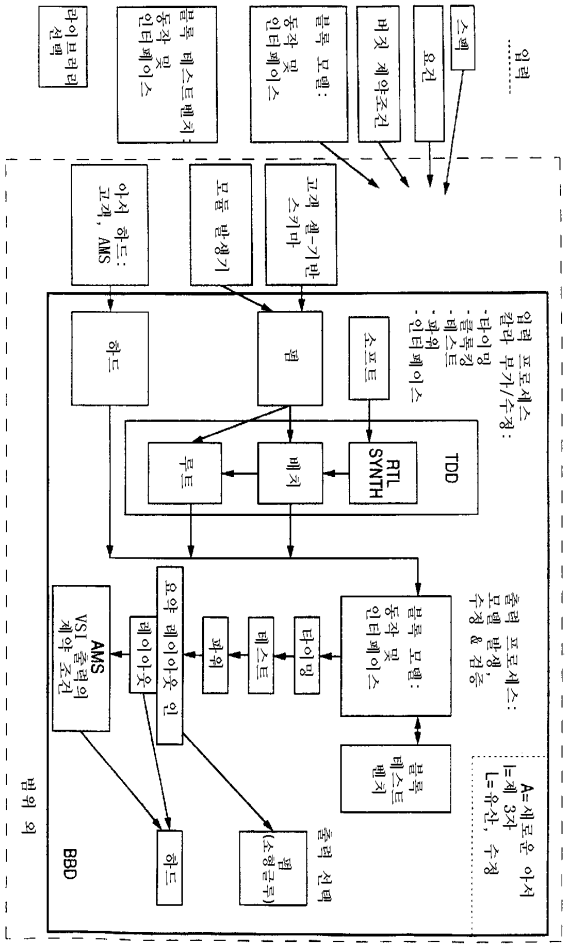
도면7


프런트-엔드 어셈턴스로부터:
■ 각 블록에 대한 테스트 기술
■ 확장된 벡터 사이즈
■ 제품 테스트 시간
■ 파라메트릭 또는 아날로그 테스트
■ 범위 필요조건
테스트 플랜 작성
DFT 규칙
탑-레벨 테스트 로직 스펙을 디자인
테스트 발생 메카니즘 생성
블록 테스트 로직 스펙을 생성

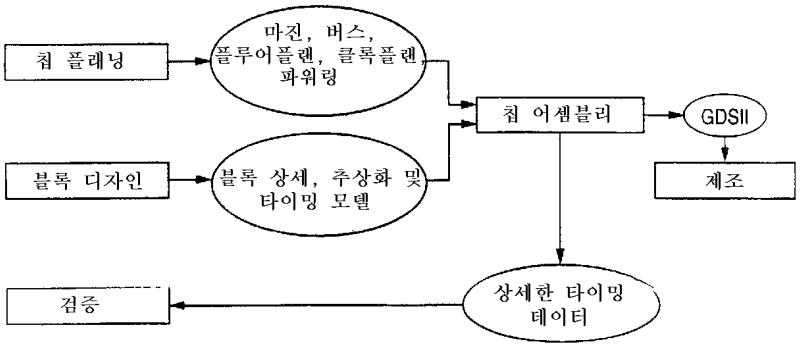
도면8


프런트-엔드 어셈턴스
새 블록 글루
타이밍 환경 생성
탑-레벨 네트리스트 생성
네트리스트 수정
글루 분배
분배된 백에노테이션
타이밍 검증 ?
아니오
클록 플랜 수정
스펙 수정
예
플루어플래닝
슬랙을 블록 제약조건으로 전환
새블록 제약조건

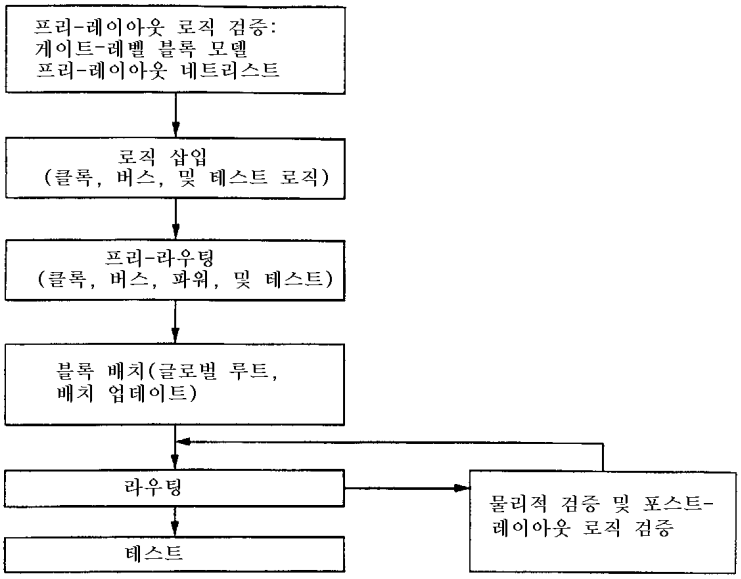
도면9


입력
스펙
요건
버짓 제약조건
블록 모델: 동작 및 인터페이스
블록 테스트벤치: 동작 및 인터페이스
라이브러리 선택
아서 하드: 고객, AMS
모듈 발생기
고객 셀-기반 스키마
입력 프로세스: 컬러 부가/수정: -타이밍 -플로링 -테스트 -파워 -인터페이스
하드
폼
소프트
TDD
RTL SYNTH
베치
루트
출력 프로세스: 모델 발생, 수정 & 검증
블록 모델: 동작 및 인터페이스
블록 테스트벤치
A=세로운 아서 L=계 3자 L=유산, 수정
요약 레이아웃 인
테이아웃
타이밍
테스트
파워
AMS VSI 출력의 제약 조건
하드
폼 (소형글루)
출력 산태
BBD
범위 외

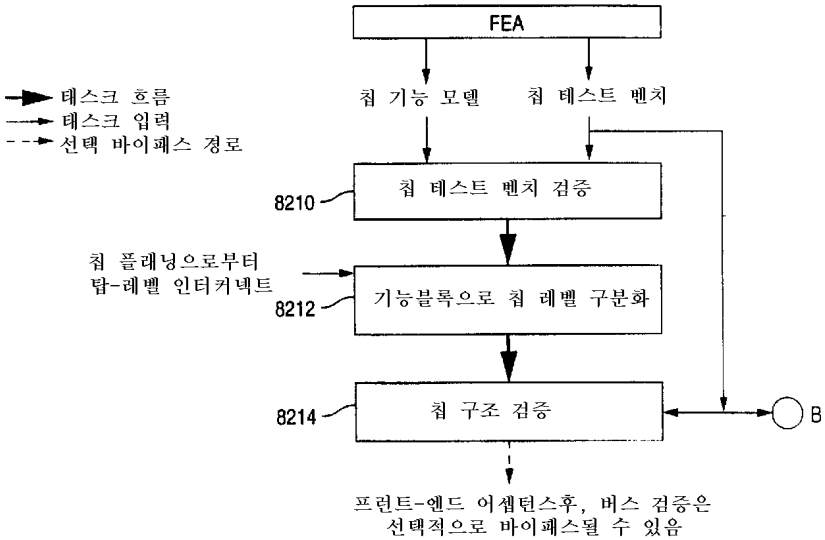
도면10


칩 플래닝
마진, 버스, 플루어플랜, 클록플랜 파워링
블록 디자인
블록 상세, 추상화 및 타이밍 모델
칩 어셈블리
GDSII
제조
검증
상세한 타이밍 데이터

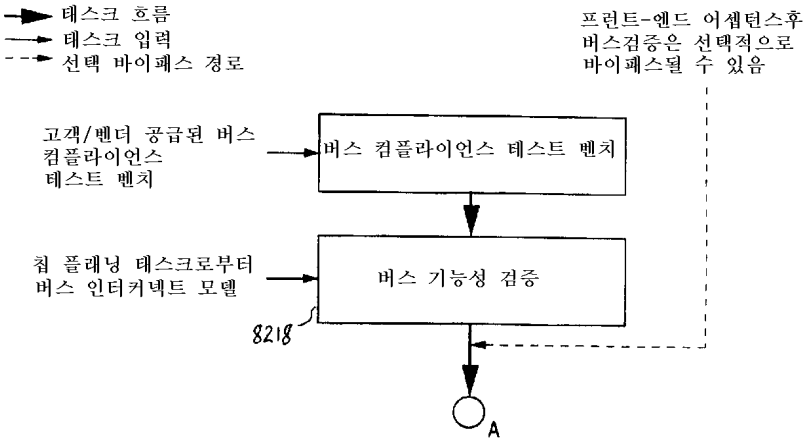
도면11


프리-레이아웃 로직 검증:
게이트-레벨 블록 모델
프리-레이아웃 네트리스트
로직 삽입
(클록, 버스, 및 테스트 로직)
프리-라우팅
(클록, 버스, 파워, 및 테스트)
블록 배치(글로벌 루트,
배치 업데이트)
라우팅
테스트
물리적 검증 및 포스트-
레이아웃 로직 검증

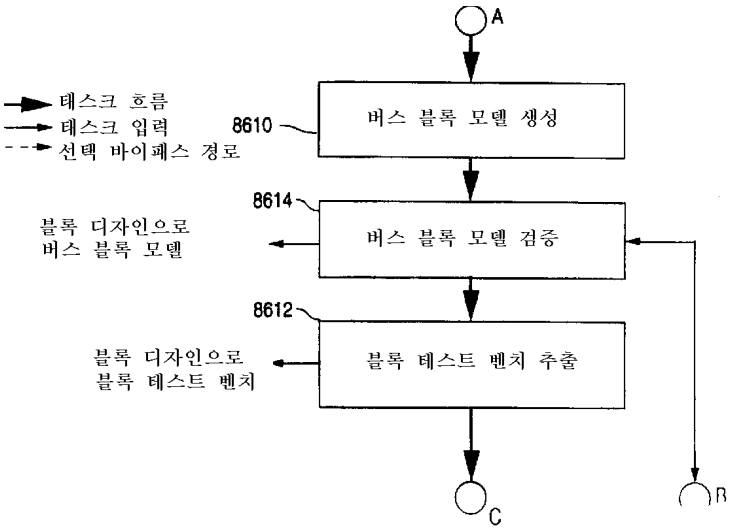
도면12


FEA
칩 기능 모델
칩 테스트 벤치
태스크 흐름
태스크 입력
선택 바이패스 경로
8210
칩 테스트 벤치 검증
칩 플래닝으로부터
탑-레벨 인터커넥트
8212
기능블록으로 칩 레벨 구분화
8214
칩 구조 검증
B
프런트-엔드 이셈턴스후, 버스 검증은
선택적으로 바이패스될 수 있음

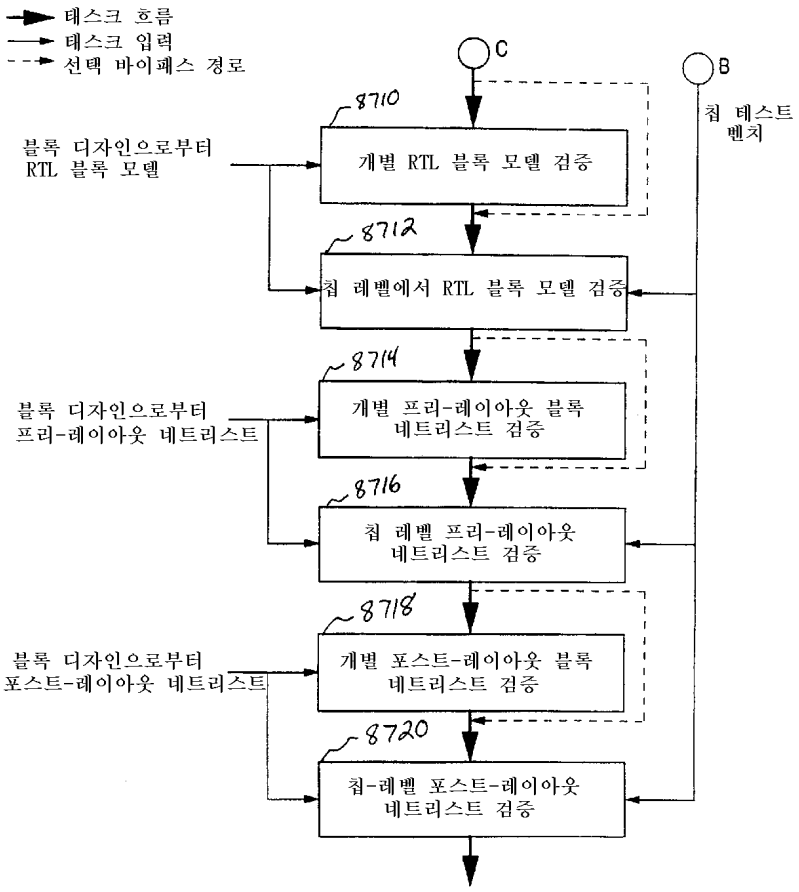
도면13


태스크 흐름
태스크 입력
선택 바이패스 경로
프런트-엔드 이셉턴스후
버스검증은 선택적으로
바이패스될 수 있음
고객/벤더 공급된 버스
컴플라이언스
테스트 벤치
버스 컴플라이언스 테스트 벤치
칩 플래닝 태스크로부터
버스 인터커넥트 모델
버스 기능성 검증
8218
A

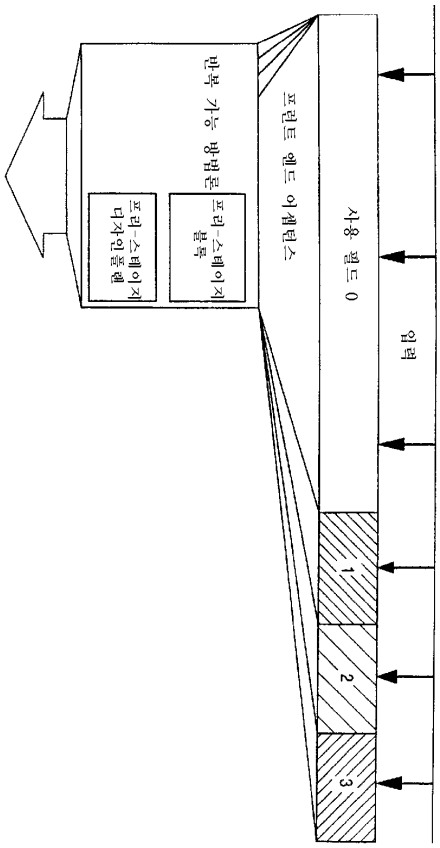
도면14


A
태스크 흐름
태스크 입력
선택 바이패스 경로
8610
버스 블록 모델 생성
8614
블록 디자인으로
버스 블록 모델
버스 블록 모델 검증
8612
블록 디자인으로
블록 테스트 벤치
블록 테스트 벤치 추출
C
B

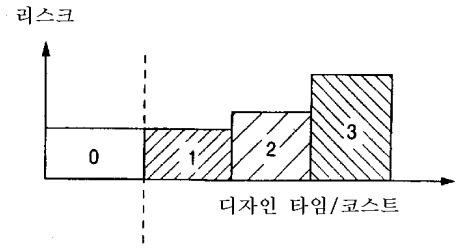
도면15


테스크 흐름
테스크 입력
선택 바이패스 경로
C
B
칩 테스트 벤치
8710
블록 디자인으로부터 RTL 블록 모델
개별 RTL 블록 모델 검증
8712
칩 레벨에서 RTL 블록 모델 검증
8714
블록 디자인으로부터 프리-레이아웃 네트리스트
개별 프리-레이아웃 블록 네트리스트 검증
8716
칩 레벨 프리-레이아웃 네트리스트 검증
8718
블록 디자인으로부터 포스트-레이아웃 네트리스트
개별 포스트-레이아웃 블록 네트리스트 검증
8720
칩-레벨 포스트-레이아웃 네트리스트 검증

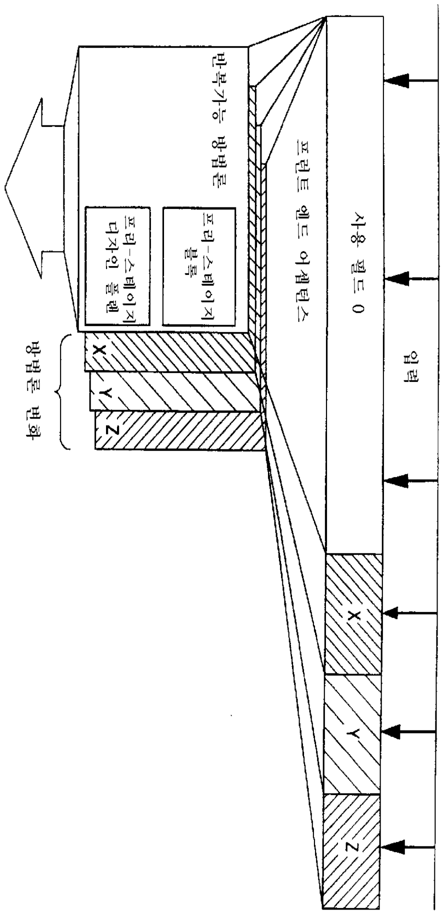
도면16


반복 가능 방법론
프리-스테이지 디자인룰북
프리-스테이지 룰북
프린트 앤드 어셈블스
사용 필드 0
입력
1
2
3

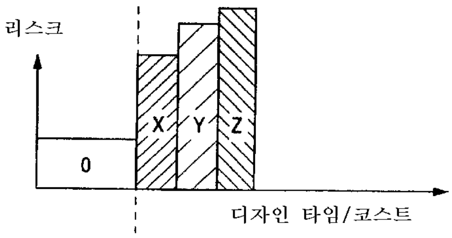
도면17


리스크
0
1
2
3
디자인 타임/코스트

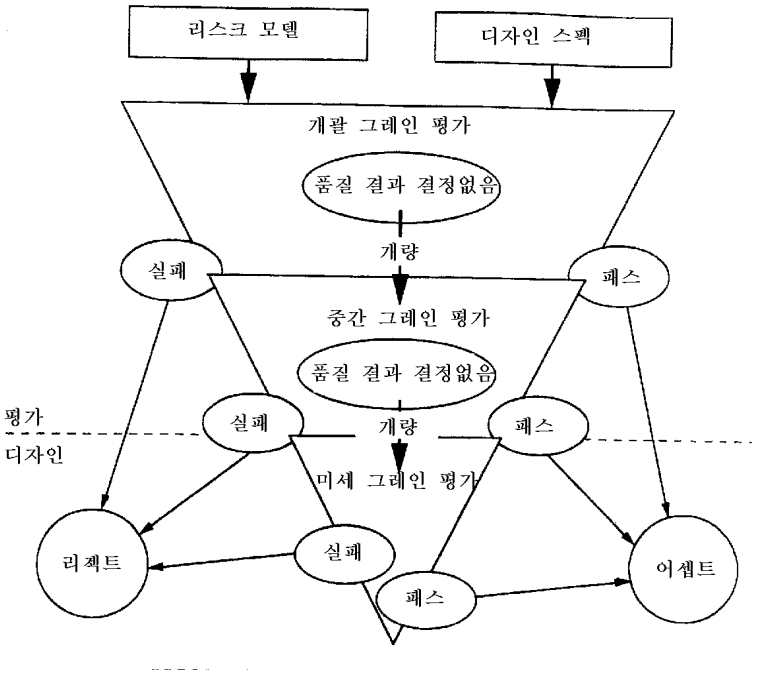
도면18


입력
사용 필드 0
프런트 엔드 어셈블스
반복가능 방법론
프리-스테이지 블록
프리-스테이지 디자인 플렌
X
Y
Z
방법론 변화

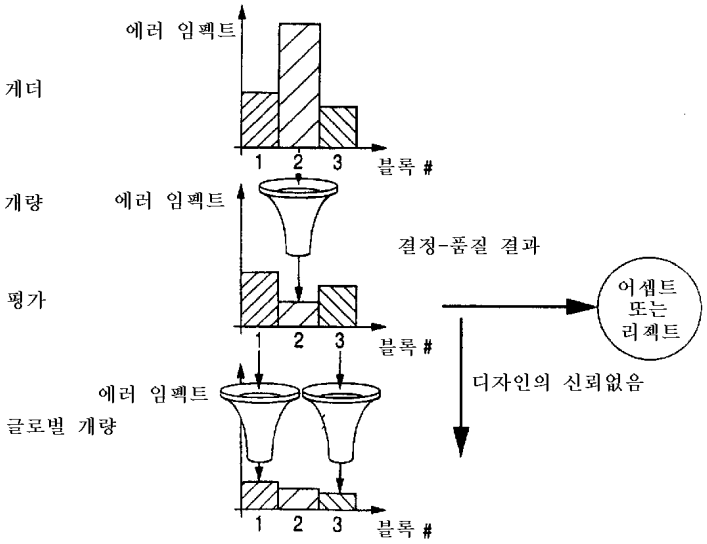
도면19


리스크
0
X
Y
Z
디자인 타임/코스트

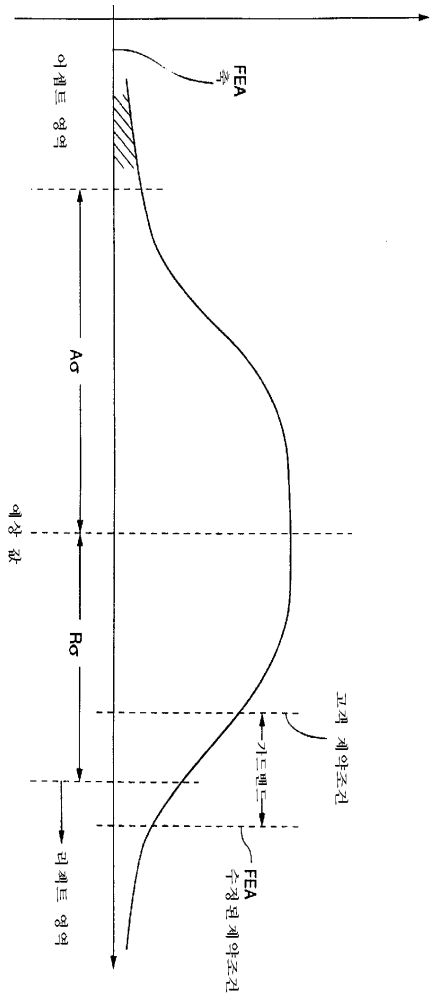
도면20


리스크 모델
디자인 스펙
개괄 그레인 평가
품질 결과 결정없음
개량
실패
패스
중간 그레인 평가
품질 결과 결정없음
평가
디자인
실패
개량
패스
미세 그레인 평가
리젝트
실패
어셉트
패스

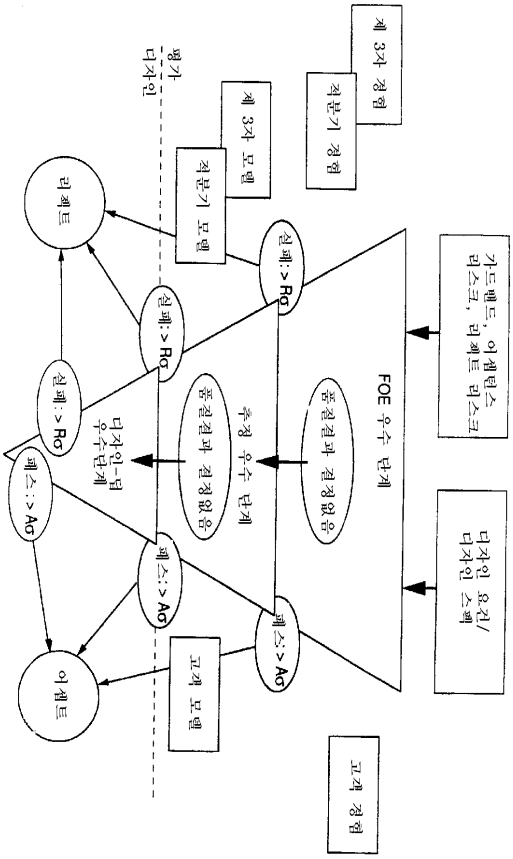
도면21


에러 임펙트
게더
1 2 3
블록 #
개량
에러 임펙트
결정-품질 결과
평가
어셉트 또는 리젝트
1 2 3
블록 #
디자인의 신뢰없음
에러 임펙트
글로벌 개량
1 2 3
블록 #

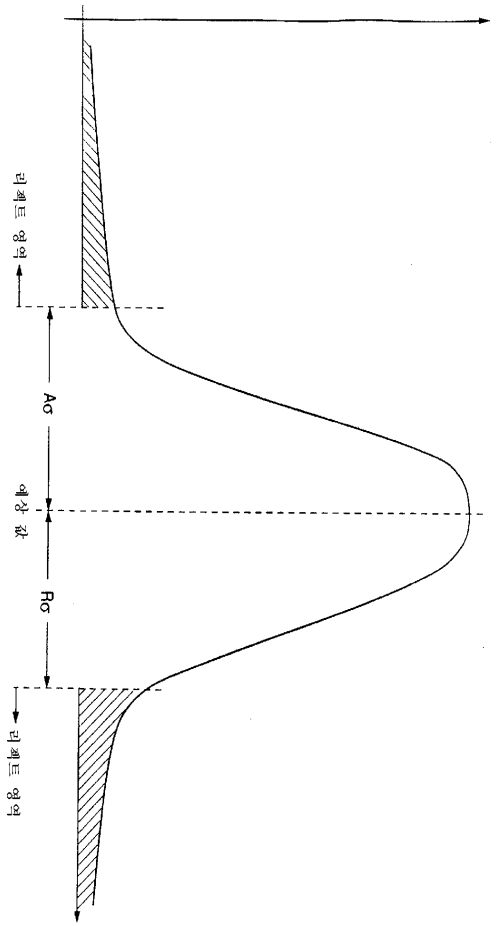
도면22



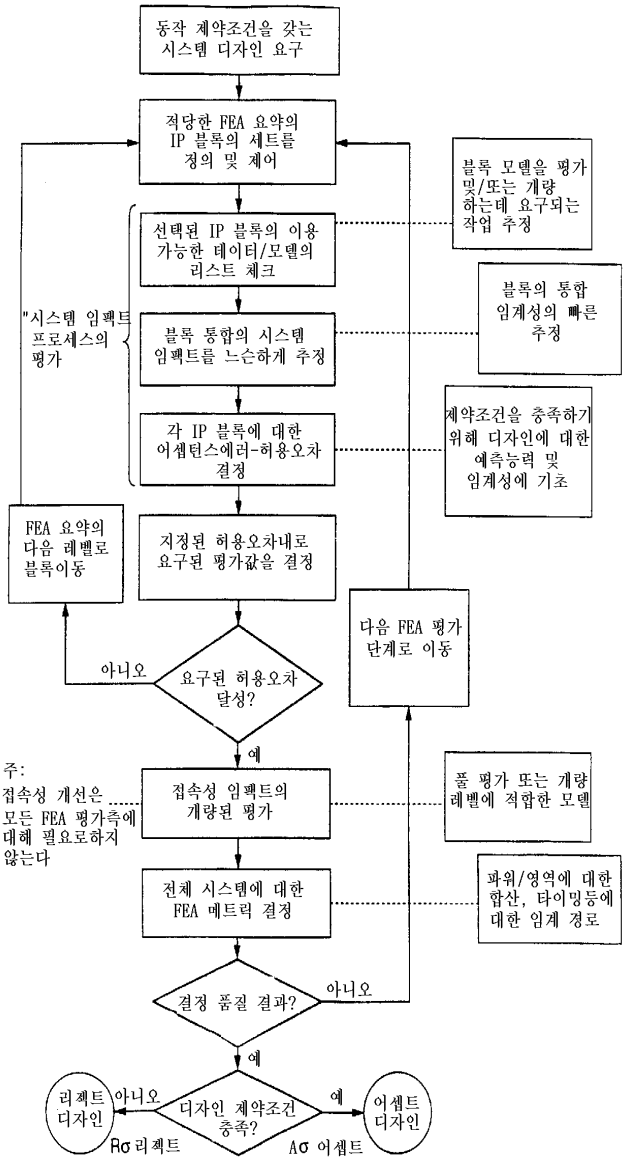
도면23


체 3차 경험
적분기 경험
가드밴드, 어셈턴스 리스크, 리쉐트 리스크
디자인 요건/ 디자인 스펙
고객 경험
FOE 우수 단계
품질결과 결정없음
추정 우수 단계
품질결과 결정 없음
체 3차 모델
적분기 모델
고객 모델
평가 디자인
리쉐트
어셈트
실패: > Rσ
실패: > Rσ
실패: > Rσ
페스: > Aσ
페스: > Aσ
페스: > Aσ
디자인-더 우수단계

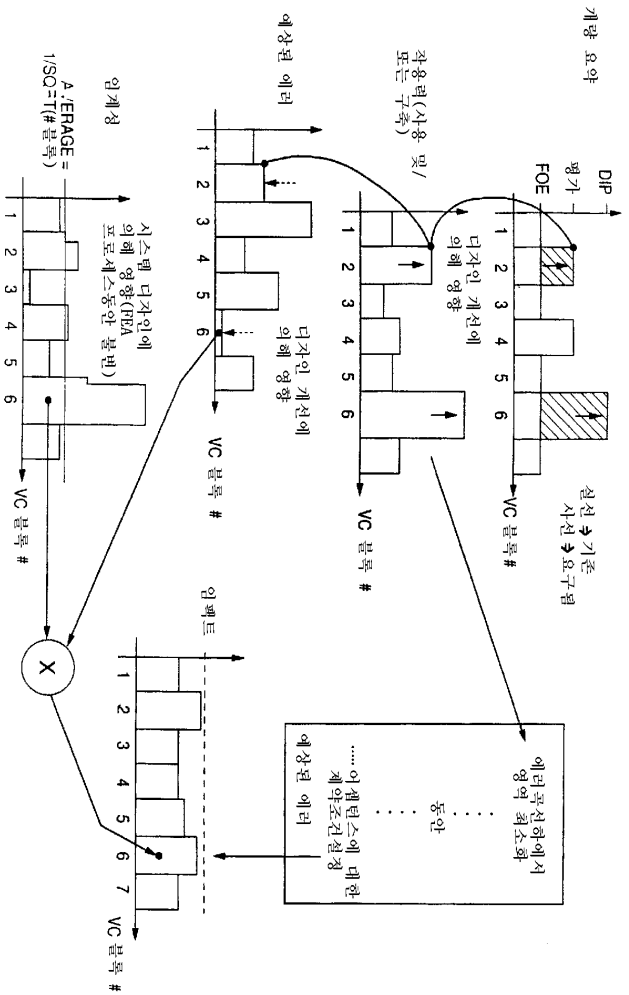
도면24



도면25


동작 제약조건을 갖는 시스템 디자인 요구
적당한 FEA 요약의 IP 블록의 세트를 정의 및 제어
선택된 IP 블록의 이용 가능한 데이터/모델의 리스트 체크
블록 통합의 시스템 임팩트를 느슨하게 추정
각 IP 블록에 대한 어셈턴스에러-허용오차 결정
지정된 허용오차내로 요구된 평가값을 결정
요구된 허용오차 달성?
아니오
예
접속성 임팩트의 개량된 평가
전체 시스템에 대한 FEA 메트릭 결정
결정 품질 결과?
아니오
예
디자인 제약조건 충족?
아니오
예
리젝트 디자인
Aσ 리젝트
어셈트 디자인
Aσ 어셈트
"시스템 임팩트 프로세스의 평가
FEA 요약의 다음 레벨로 블록이동
다음 FEA 평가 단계로 이동
주: 접속성 개선은 모든 FEA 평가측에 대해 필요로하지 않는다
블록 모델을 평가 및/또는 개량 하는데 요구되는 작업 추정
블록의 통합 임계성의 빠른 추정
제약조건을 충족하기 위해 디자인에 대한 예측능력 및 임계성에 기초
풀 평가 또는 개량 레벨에 적합한 모델
파워/영역에 대한 합산, 타이밍등에 대한 임계 경로

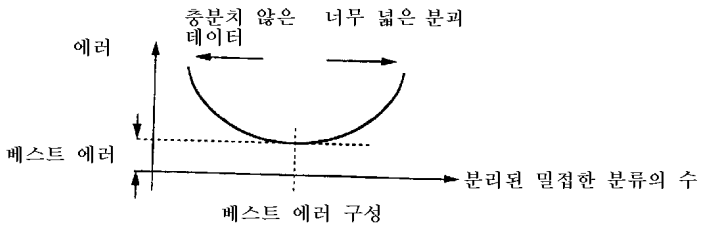
도면26


개량 요약
DIP
평가
FOE
VC 블록#
설선 → 기존
사선 → 요구됨
디자인 개선에
의해 영향
잔용량(사용 및/
또는 구축)
VC 블록 #
에상된 에러
디자인 개선에
의해 영향
VC 블록 #
임계성
시스템 디자인에
의해 영향(FDA
프로세스동안 불변)
VC 블록 #
임펙트
VC 블록 #
에러곡선하에서
영역 최소화
.....
동안
.....
.....에 셀된스에 대한
제약조건 설정
에상된 에러

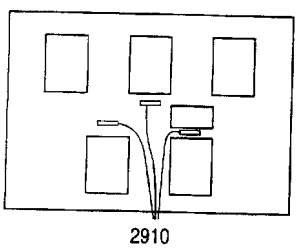
도면27

평가축	제약조건 타입	제약조건 분류	라우팅 계산	예 임계성 측정
파워	PBR 동작의 모드	상대적	중간	(예 상원-블록-파워 에러)+ (블록-라우팅 임계성)*0.5*V²*E블록
성능	천탈 지연	절대적	미세	N/A (즉, 시스템 합산기준이 아닌 로직 점속)
	레이턴시	상대적	중간	시스템에 대한 레이턴시-경로의 임계성: 1/((1-Pf(결함))* ((요구된 경로-레이턴시)+ (Σ(블록 레이턴시)+Σ(버스 레이턴시)))) (레이턴시 에러)/(1-Pf(결함))
	스루풋	절대적	개괄	N/A (즉 단일 병목 블록에 의해 조절)
영역	영역	상대적	중간	(영역 에러)+(블록-라우팅 임계성)*α
코스트	NRE	상대적	개괄	
	유닛당 코스트	상대적	개괄	
스케줄	자원 할당	혼합	개괄	
	천탈 타임라인	상대적	개괄	
리스크	에러 가능성	혼합	개괄	
	에러 임팩트	혼합	개괄	

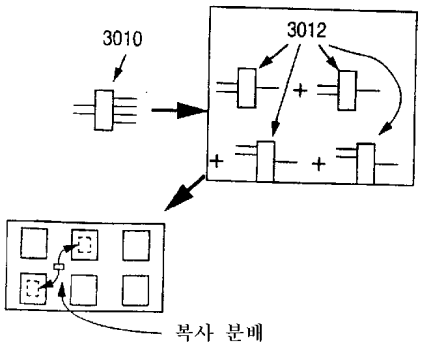
도면28



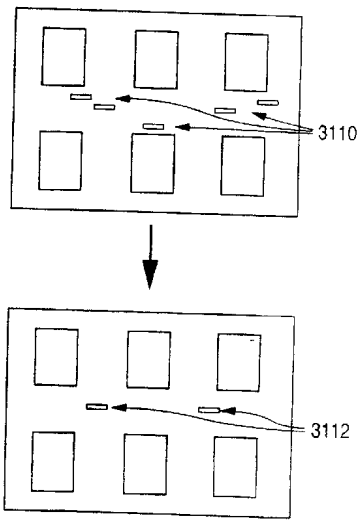
도면29



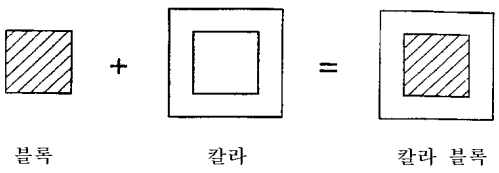
도면30



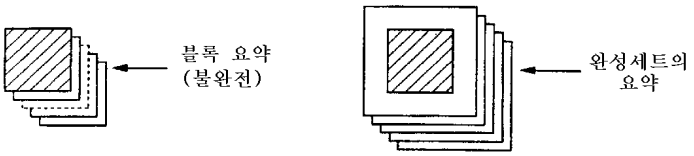
도면31



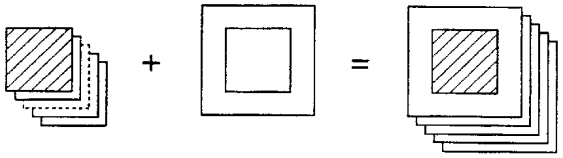
도면32



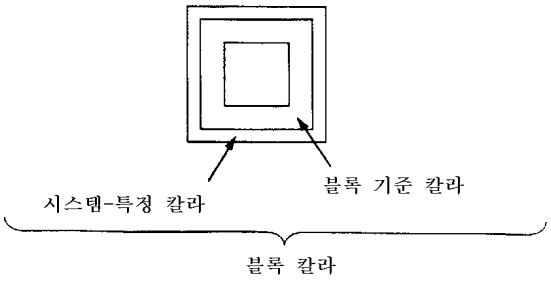
도면33



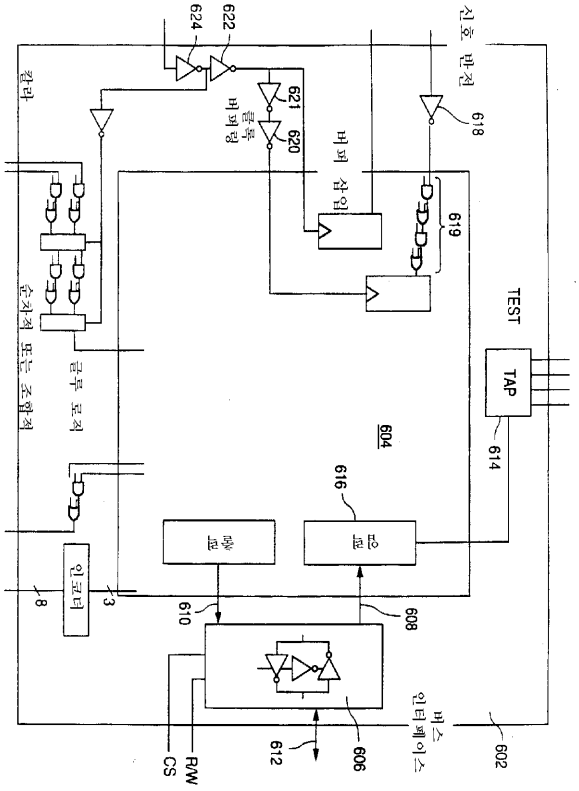
도면34



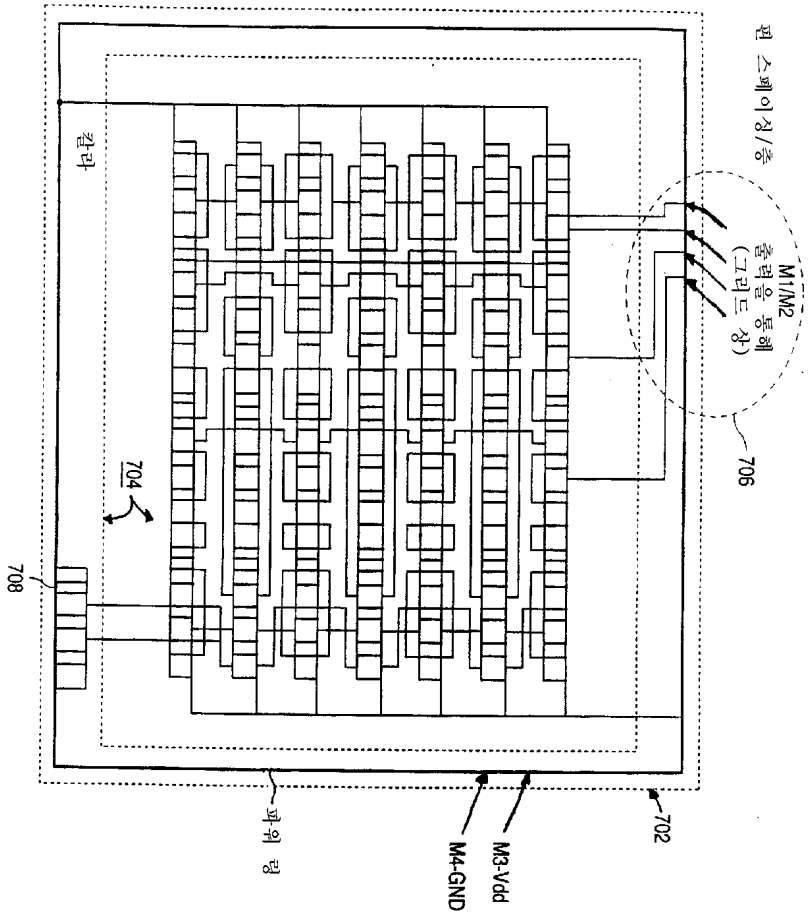
도면35



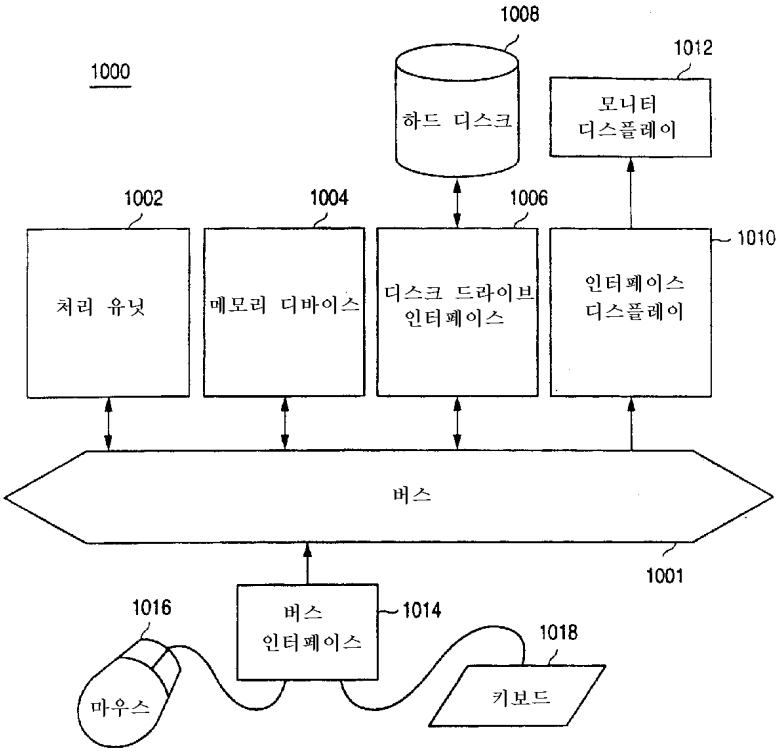
도면36


624
622
신호 반전
칼라
621
620
버퍼링
클록
버퍼 삽입
618
619
TEST
TAP
614
604
순차적 또는 조합적
글루 로직
616
출력
입력
인코더
8
3
610
608
CS
R/W
612
606
버스 인터페이스
602

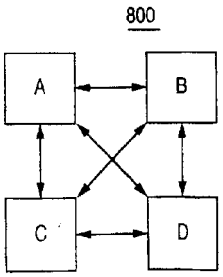
도면37



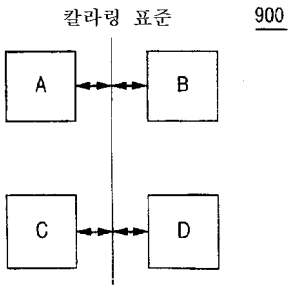
도면38



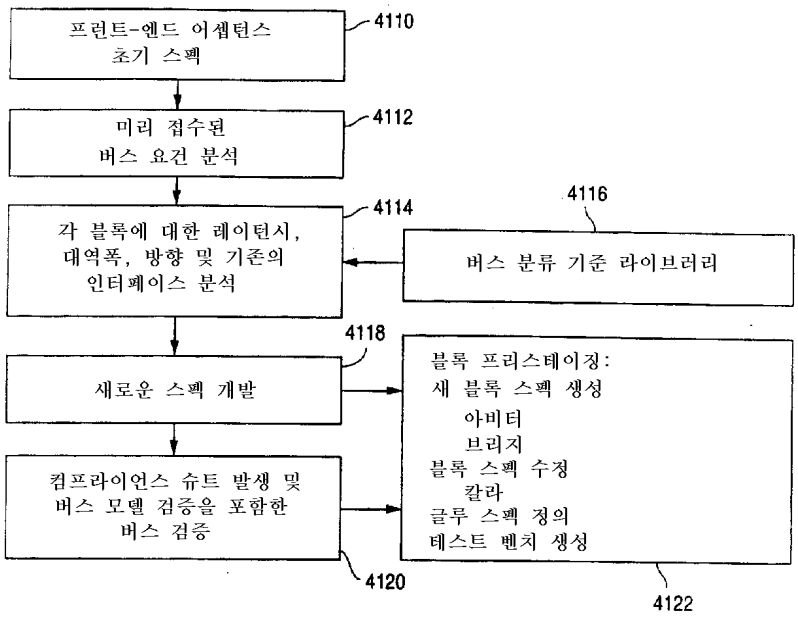
도면39



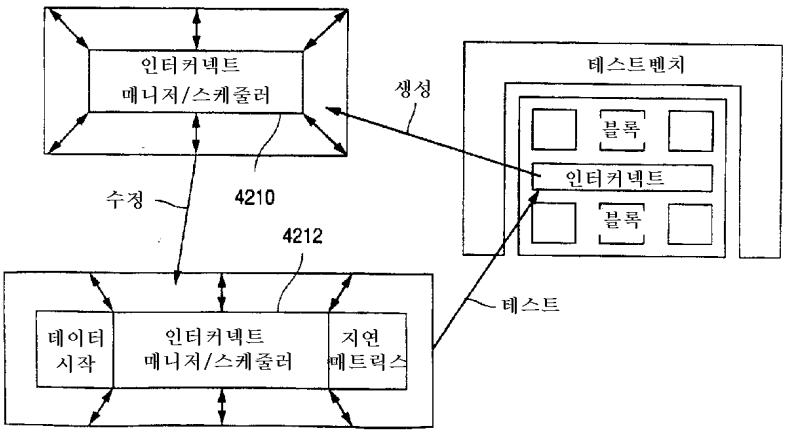
도면40



도면41



도면42



도면43

에서 \ 로	블록 1	블록 2	블록 3	메모리	PCI	PIO
블록 1	0	10,000	100	10,000	100	100
블록 2	8,000	0	100	2,000	10,000	100
블록 3	200	100	0	100	200	100
메모리	6,000	6,000	100	0	100	100
PCI	6,100	4,100	0	200	0	0
PIO	0	0	0	0	0	0

도면44

에서 \ 로	블록 1	블록 2	블록 3	메모리	PCI	PIO
블록 1	0	200	25	200	4	25
블록 2	160	0	100	40	200	25
블록 3	25	100	0	25	25	25
메모리	120	180	25	0	6	25
PCI	122	82	0	50	0	0
PIO	0	0	0	0	0	0

도면45

에서 \ 로	블럭 1	블럭 2	블럭 3	메모리	PCI	PIO
블럭 1	n/a	50	1,000	50	100	1,000
블럭 2	50	n/a	1,000	300	100	1,000
블럭 3	1,000	1,000	n/a	500	500	1,000
메모리	50	50	500	n/a	100	1,000
PCI	100	100	n/a	50	n/a	n/a
PIO	n/a	n/a	n/a	n/a	n/a	n/a

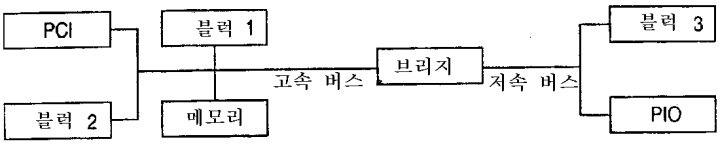
도면46

에서 \ 로	사이트1	사이트2	사이트3	사이트4	사이트5	사이트6
사이트 1	0	1	4	9	16	25
사이트 2	1	0	1	4	9	16
사이트 3	4	1	0	1	4	9
사이트 4	9	4	1	0	1	4
사이트 5	16	9	4	1	0	1
사이트 6	25	16	9	4	1	0

도면47

에서 \ 로	PC1	블럭 2	블럭 1	메모리	블럭 3	PIO
PCT	0	4,100	6,100	200	0	0
블럭 2	10,000	0	8,000	2,000	100	100
블럭 1	100	10,000	0	10,000	100	100
메모리	100	6,000	6,000	0	100	100
블럭 3	200	100	200	100	0	100
PIO	0	0	0	0	0	0

도면48



도면49

에서 \ 로	A	B	C	D	E	F	G	H
A	0	##	##					
B	##	0	##					
C	##	##	0					
D				0	##			
E				##	0	##		
F					##	0	6	5
G						6	0	##
H						5	##	0

도면50

에서 \ 로	A	B	C	D	E	F	G	H
A	0	##	##					
B	##	0	##					
C	##	##	0					
D				0	##			
E				##	0	##		
F					##			
G								
H								

도면51

에서 \ 로	A	B	C	D	E	F	G	H
A	0	##	##					
B	##	0	##					
C	##	##	0					
D				0	##			
E				##	0	##		
F					##	0	6	5
G						6	0	##
H						5	##	0

도면52

에서 \ 로	A	B	C	D	E	E'	F	G	H
A	0	##	##						
B	##	0	##						
C	##	##	0						
D				0	##				
E				##	0				
E'						0	##		
F						##	0	6	5
G							6	0	##
H							5	##	0

도면53

에서 \ 로	PCI	블럭 2	블럭 1	메모리	블럭 3	PIO
PCI	0	4,100	6,100	200	0	0
블럭 2	10,000	0	8,000	2,000	100	100
블럭 1	100	10,000	0	10,000	100	100
메모리	100	6,000	6,000	0	100	100
블럭 3	200	100	200	100	0	100
PIO	0	0	0	0	0	0

도면54

에서 \ 로	PCI	블럭 2	블럭1	메모리	블럭 3	PIO
PCI	0	4,100	6,100	200	0	0
블럭 2	10,000	0	8,000	2,000	100	100
블럭 1	100	10,000	0	10,000	100	100
메모리	100	6,000	6,000	0	100	100
블럭 3	200	100	200	100	0	100
PIO	0	0	0	0	0	0

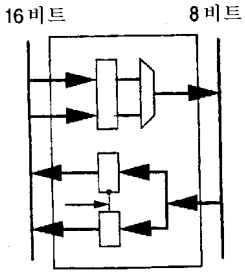
도면55

에서 \ 로	버스 1	버스 2
버스 1	62,600	600
버스 2	600	100

도면56

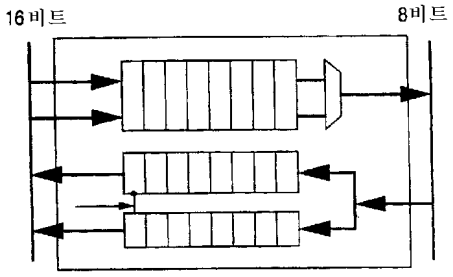
에서 \ 로	블록 1	블록2	블록3	메모리	PCI	PIO
블록 1	n/a	31	988	31	91	988
블록 2	31	n/a	997	281	81	988
블록 3	976	997	n/a	488	476	988
메모리	31	38	488	n/a	94	988
PCI	81	81	n/a	49	n/a	n/a
PIO	n/a	n/a	n/a	n/a	n/a	n/a

도면57



8 TO 16 비트 브리지

도면58



8 깊도 FIFOs를 갖는 브리지

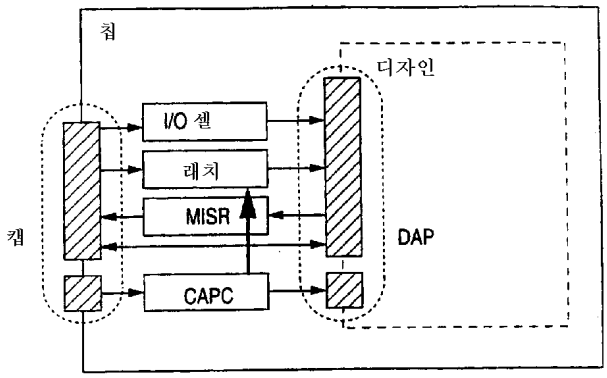
도면59

버스 타입	사용 최소	사용 최대	레이턴시 데이터	레이턴시 전달
1. 직렬, 동기, 클록-재생 버스	5%	25%	50	200
2. 다중 라인 동기 클록-재생 버스	5%	25%	20	100
3. 결합된 데이터 및 어드레스 라인 갖는 다중 라인 동기 버스	10%	25%	5	25
4. 별도의 데이터 및 어드레스 라인을 갖는 동기 버스	25%	50%	2.5	10
5. 단일-레벨 파이프라인 데이터 및 어드레스 라인 갖는 양방향 버스	25%	75%	2	5
6. 복잡한 조정을 갖는 다중 레벨 파이프 라인 버스	50%	75%	1.5	2.5
7. 크로스바 스위치	75%	100%	1	2
8. 포인트-투-포인트 단방향 와이어	100%	100%	0.5	1

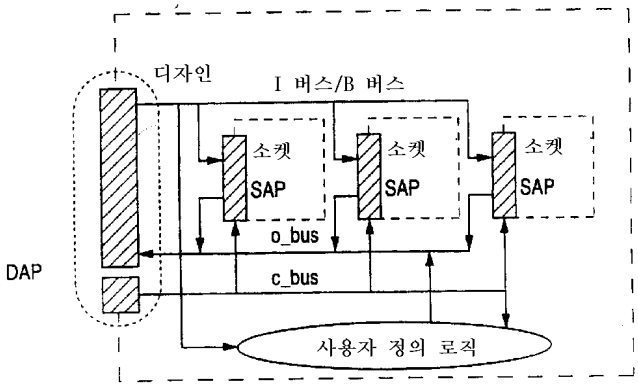
도면60

파라미터	정상 모드	테스트 모드	절연 모드	경계 모드
테스트 모델	BSR MODE=0	BSR MODE=1	BSR MODE=1	BSR MODE=1
테스트 제어기 디자인	JTAG ir = bypass	JTAG ir=vc_test	JTAG ir=vc_isol	JTAG ir=udl_test
테스트 절연	N/A	출력 절연	입력 절연	입력 절연
테스트 유효화	기능 테스트 벤치	VC 테스트 벡터	입력 절연	업데이트 바이패스

도면61



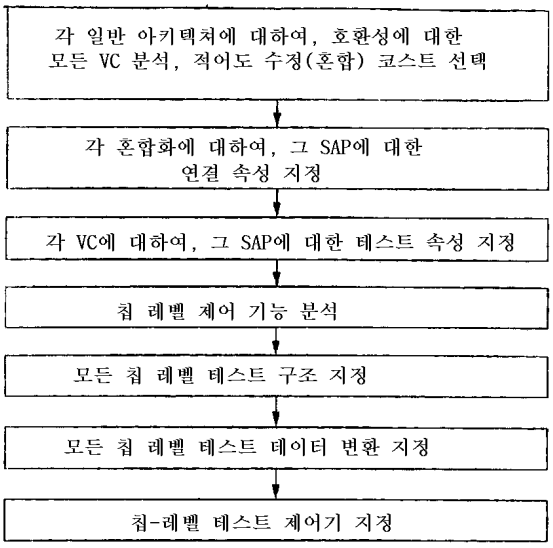
도면62



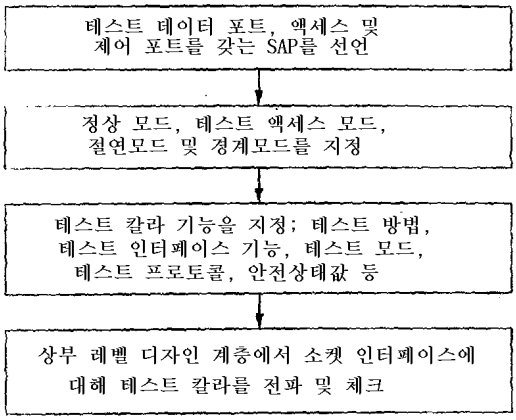
도면63

일반 아키텍처	타겟 블록	테스트 인터페이스	테스트 방법
테스트 버스	레이턴시 블록, 비스캔 블록	칩 I/O로부터 액세스 정향	기능 벡터
BSR	스캔-기반 블록	다중 스캔 + BSR 체인	연결된 스캔 벡터
비스트	RAM 및 FIFO와 같은 규칙 블록	비스트 제어기	알고리즘 테스트 패턴 구축
	풀 스캔과 같은 로직 비스트 구축	PRPG/MISR에 대한 비스트 제어기	랜덤 테스트 패턴 구축
탭	디버그 및 진단 블록 구축	TAP (TDI, TRST, TMS, TCK, TDO)	JTAG 프로토콜

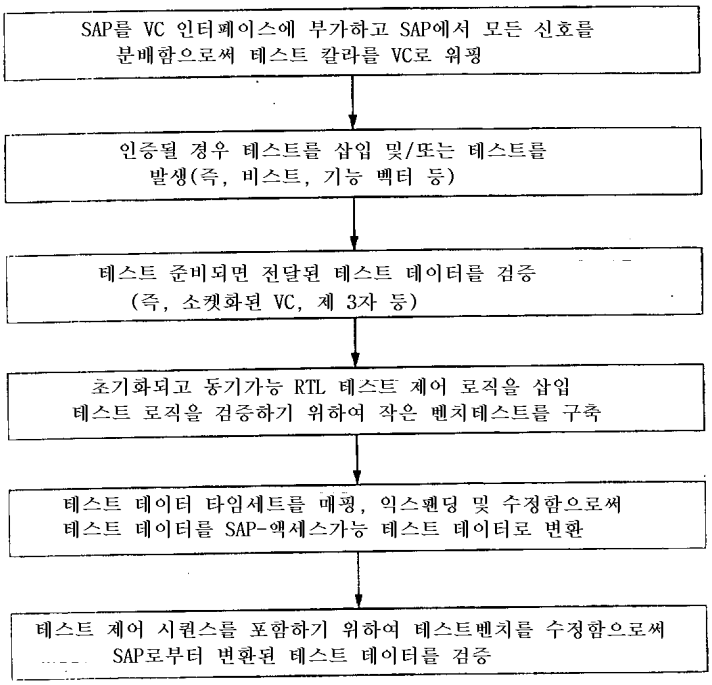
도면64



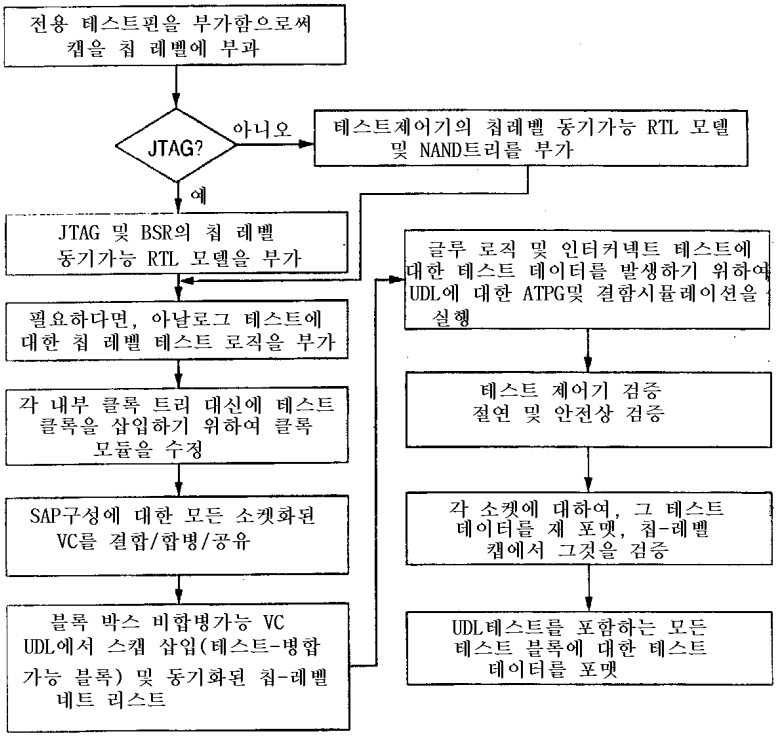
도면65


테스트 데이터 포트, 액세스 및
제어 포트를 갖는 SAP를 선언
정상 모드, 테스트 액세스 모드,
절연모드 및 경계모드를 지정
테스트 칼라 기능을 지정; 테스트 방법,
테스트 인터페이스 기능, 테스트 모드,
테스트 프로토콜, 안전상태값 등
상부 레벨 디자인 계층에서 소켓 인터페이스에
대해 테스트 칼라를 전파 및 체크

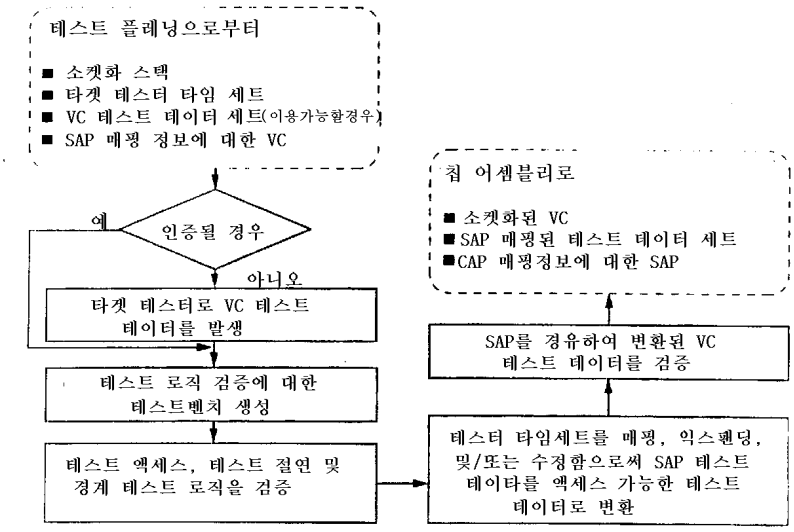
도면66


SAP를 VC 인터페이스에 부가하고 SAP에서 모든 신호를
분배함으로써 테스트 칼라를 VC로 워핑
인증될 경우 테스트를 삽입 및/또는 테스트를
발생(즉, 비스트, 기능 벡터 등)
테스트 준비되면 전달된 테스트 데이터를 검증
(즉, 소켓화된 VC, 제 3자 등)
초기화되고 동기가능 RTL 테스트 제어 로직을 삽입
테스트 로직을 검증하기 위하여 작은 벤치테스트를 구축
테스트 데이터 타임세트를 매핑, 익스펜딩 및 수정함으로써
테스트 데이터를 SAP-액세스가능 테스트 데이터로 변환
테스트 제어 시퀀스를 포함하기 위하여 테스트벤치를 수정함으로써
SAP로부터 변환된 테스트 데이터를 검증

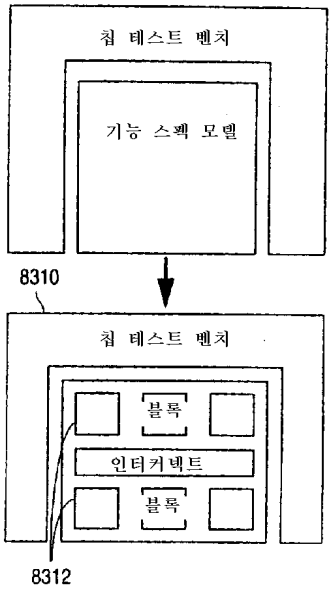
도면67


전용 테스트핀을 부가함으로써 캡을 칩 레벨에 부과
JTAG?
아니오
테스트제어기의 칩레벨 동기가능 RTL 모델 및 NAND트리를 부가
예
JTAG 및 BSR의 칩 레벨 동기가능 RTL 모델을 부가
필요하다면, 아날로그 테스트에 대한 칩 레벨 테스트 로직을 부가
각 내부 클록 트리 대신에 테스트 클록을 삽입하기 위하여 클록 모듈을 수정
SAP구성에 대한 모든 소켓화된 VC를 결합/합병/공유
블록 박스 비합병가능 VC UDL에서 스캔 삽입(테스트-병합 가능 블록) 및 동기화된 칩-레벨 네트 리스트
글루 로직 및 인터커넥트 테스트에 대한 테스트 데이터를 발생하기 위하여 UDL에 대한 ATPG및 결함시뮬레이션을 실행
테스트 제어기 검증 절연 및 안전상 검증
각 소켓에 대하여, 그 테스트 데이터를 재 포맷, 칩-레벨 캡에서 그것을 검증
UDL테스트를 포함하는 모든 테스트 블록에 대한 테스트 데이터를 포맷

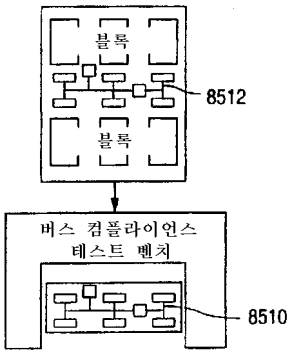
도면68


테스트 플래닝으로부터
■ 소켓화 스택
■ 타겟 테스터 타임 세트
■ VC 테스트 데이터 세트(이용가능할경우)
■ SAP 매핑 정보에 대한 VC
예
인증될 경우
아니오
타겟 테스터로 VC 테스트 테이터를 발생
테스트 로직 검증에 대한 테스트벤치 생성
테스트 액세스, 테스트 절연 및 경계 테스트 로직을 검증
칩 어셈블리로
■ 소켓화된 VC
■ SAP 매핑된 테스트 데이터 세트
■ CAP 매핑정보에 대한 SAP
SAP를 경유하여 변환된 VC 테스트 데이터를 검증
테스터 타임세트를 매핑, 익스펜딩, 및/또는 수정함으로써 SAP 테스트 데이타를 액세스 가능한 테스트 데이터로 변환

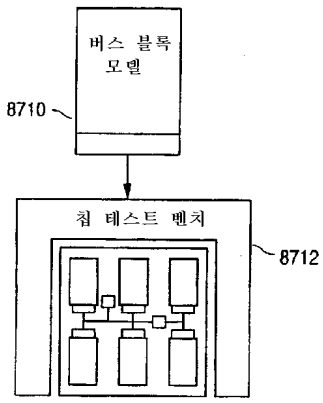
도면69



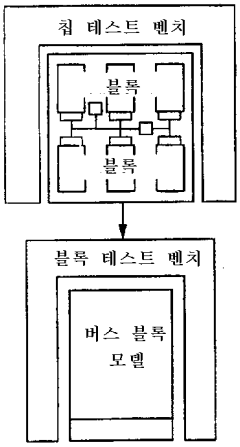
도면70



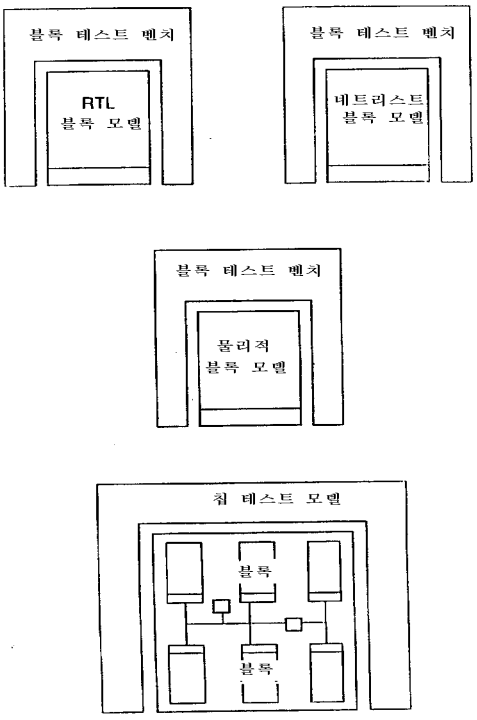
도면71



도면72


칩 테스트 벤치
블록
블록
블록 테스트 벤치
버스 블록
모델

도면73


블록 테스트 벤치
RTL
블록 모델
블록 테스트 벤치
네트리스트
블록 모델
블록 테스트 벤치
물리적
블록 모델
칩 테스트 모델
블록
블록