



(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。 Int. Cl. H01L 21/66 (2006.01)	(45) 공고일자 (11) 등록번호 (24) 등록일자	2007년07월06일 10-0736679 2007년07월02일
---	-------------------------------------	--

(21) 출원번호 (22) 출원일자 심사청구일자	10-2006-0075056 2006년08월09일 2006년08월09일	(65) 공개번호 (43) 공개일자
----------------------------------	---	------------------------

(73) 특허권자	주식회사 유니테스트 경기도 용인시 기흥구 고매동 352-9
(72) 발명자	강종구 경기 용인시 기흥구 지곡동 써니밸리아파트 114-1405
(74) 대리인	특허법인지명
(56) 선행기술조사문헌 JP01287486 A	

심사관 : 김동국

전체 청구항 수 : 총 18 항

(54) 반도체 테스트를 위한 패턴 생성 장치 및 패턴 생성 방법

(57) 요약

본 발명은 서로 다른 프레임에 대해서는 제약 조건 없이 모든 데이터 연산이 가능하고 동일한 프레임 내의 인터리브 사이클에 대해서는 미리 지정된 데이터 예측 조건에 적합한 데이터 연산만이 가능하도록 데이터 연산을 예측하여 컴파일된 테스트 패턴 프로그램을 판독하는 테스트 패턴 프로그램 판독부와, 상기 테스트 패턴 프로그램을 기초로 인터리브 방식으로 데이터 연산을 수행한 후 패턴을 생성하는 다수의 알고리즘 패턴 생성부와, 상기 패턴을 다중화하여 반도체 소자의 테스트를 위한 테스트 패턴 데이터를 생성하는 다중화부를 포함하는 반도체 테스트를 위한 패턴 생성 장치에 관한 것이다.

본 발명에 따르면, 테스트 수행자가 테스트를 위한 소스 코드를 작성시 패턴 생성을 위한 내부 연산을 인터리브 동작 각각에 대해서 구체적으로 분석하여 작성하지 않더라도 데이터 연산을 예측하여 테스트 프로그램이 컴파일되어 인터리브 방식을 통하여 테스트 패턴 데이터의 생성이 가능하다.

대표도

도 3

특허청구의 범위

청구항 1.

서로 다른 프레임에 대해서는 제약 조건 없이 모든 데이터 연산이 가능하고 동일한 프레임 내의 인터리브(interleave) 사이클에 대해서는 미리 지정된 데이터 예측 조건에 적합한 데이터 연산만이 가능하도록 데이터 연산을 예측하여 컴파일된 테스트 패턴 프로그램을 판독하는 테스트 패턴 프로그램 판독부와,

상기 테스트 패턴 프로그램을 기초로 인터리브 방식으로 데이터 연산을 수행한 후 패턴을 생성하는 다수의 알고리즘 패턴 생성부와,

상기 패턴을 다중화하여 반도체 소자의 테스트를 위한 테스트 패턴 데이터를 생성하는 다중화부를 포함하는 반도체 테스트를 위한 패턴 생성 장치.

청구항 2.

제1항에 있어서,

상기 다수의 알고리즘 패턴 생성부는 서로 논리적으로 연결되어 상기 테스트 패턴 프로그램의 각 프레임의 해당되는 상기 인터리브 사이클에 대응하는 데이터 연산을 각각 수행하며,

마지막 인터리브 사이클에 대응하는 데이터 연산 결과는 첫 번째 인터리브 사이클에 대응하는 데이터 연산에 피드백되는 것인 반도체 테스트를 위한 패턴 생성 장치.

청구항 3.

제1항에 있어서,

상기 알고리즘 패턴 생성부는 다수의 필드에 대해서 각각 데이터 연산을 수행하는 것인 반도체 테스트를 위한 패턴 생성 장치.

청구항 4.

제1항에 있어서,

상기 테스트 패턴 프로그램은 다른 필드를 참조하는 데이터 연산에 대해서는 이전 프레임에서의 상기 다른 필드의 연산 결과를 참조하여 현재 프레임의 각 인터리브 사이클에 대응하는 데이터 연산이 상기 미리 지정된 데이터 예측 조건에 적합하도록 데이터 연산을 예측하여 컴파일되는 것인 반도체 테스트를 위한 패턴 생성 장치.

청구항 5.

제1항에 있어서,

상기 반도체 테스트를 위한 패턴 생성 장치는 테스트 패턴 프로그램 처리 장치와 연결되며,

상기 테스트 패턴 프로그램 처리 장치는,

소스 코드를 상기 테스트 패턴 프로그램으로 컴파일하는 컴파일러와,

상기 소스 코드가 상기 미리 지정된 데이터 예측 조건에 적합한 데이터 연산만으로 작성되었는지 검사하는 테스트 패턴 프로그램 검사부

를 포함하는 것인 반도체 테스트를 위한 패턴 생성 장치.

청구항 6.

제5항에 있어서,

상기 테스트 패턴 프로그램 검사부에서 상기 소스 코드가 상기 미리 지정된 데이터 예측 조건에 적합한 데이터 연산만으로 작성되지 않은 것으로 판단한 경우,

상기 컴파일러는 컴파일을 수행하지 않는 것인 반도체 테스트를 위한 패턴 생성 장치.

청구항 7.

제5항에 있어서,

상기 컴파일러는 상기 소스 코드를 상기 인터리브 사이클에 대하여 미리 지정된 데이터 예측 조건을 기초로 변환하여 상기 테스트 패턴 프로그램으로 컴파일하는 것인 반도체 테스트를 위한 패턴 생성 장치.

청구항 8.

제1항에 있어서,

상기 미리 지정된 데이터 예측 조건에 적합한 데이터 연산은 다음 표에 의해서 정의되는 연산인 것인 반도체 테스트를 위한 패턴 생성 장치.

		이전 코드												
		A=A	A=/A	A=A*2	A=A/2	A=B	A=A+B	A=A-B	A=A&B	A=A B	A=A^B	A=imm	A=A+imm	A=A-imm
현재 코드	A=A	A=A	A=/A	A=C	A=C	A=B	A=A+B	A=A-B	A=A&B	A=A B	A=A^B	A=imm	A=C	A=C
	A=/A	A=/A	A=A	X	X	X	X	X	X	X	X	X	X	X
	A=A*2	A=C*2	X	A=C*2	A=C	X	X	X	X	X	X	X	X	X
	A=A/2	A=C/2	X	A=C	A=C/2	X	X	X	X	X	X	X	X	X
	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B
	A=A+B	A=A+B	X	X	X	X	X	X	X	X	X	X	X	X
	A=A-B	A=A-B	X	X	X	X	X	X	X	X	X	X	X	X
	A=A&B	A=A&B	X	X	X	X	X	X	X	X	X	X	X	X
	A=A B	A=A B	X	X	X	X	X	X	X	X	X	X	X	X
	A=A^B	A=A^B	X	X	X	X	X	X	X	X	X	X	X	X
	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm
	A=A+imm	A=C+imm	X	X	X	X	X	X	X	X	X	X	A=C+imm	A=C+imm
	A=A-imm	A=C-imm	X	X	X	X	X	X	X	X	X	X	A=C-imm	A=C-imm

(여기서 'A', 'B'는 데이터 필드, 'C'는 이전에 컴파일된 코드, 'X'는 예측되지 않는 경우를 나타내며, 'imm'는 즉시값, '/'는 나누기 연산자, '*'는 곱하기 연산자, '+'는 더하기 연산자, '-'는 빼기 연산자, '&'는 bitwise AND 연산자, '|'는 bitwise OR 연산자, '^'는 bitwise XOR 연산자를 각각 나타냄)

청구항 9.

제1항에 있어서,

상기 다수의 알고리즘 패턴 생성부는 2개 또는 4개인 것인 반도체 테스트를 위한 패턴 생성 장치.

청구항 10.

(a) 서로 다른 프레임에 대해서는 제약 조건 없이 모든 데이터 연산이 가능하고 동일한 프레임 내의 인터리브 사이클에 대해서는 미리 지정된 데이터 예측 조건에 적합한 데이터 연산만이 가능하도록 데이터 연산을 예측하여 컴파일된 테스트 패턴 프로그램을 판독하는 단계와,

(b) 상기 테스트 패턴 프로그램을 기초로 인터리브 방식으로 데이터 연산을 수행하여 패턴을 생성하는 단계와,

(c) 상기 패턴을 다중화하여 반도체 소자의 테스트를 위한 테스트 패턴 데이터를 생성하는 단계

를 포함하는 반도체 테스트를 위한 패턴 생성 방법.

청구항 11.

제10항에 있어서, 상기 단계 (b)는,

(b-1) 다수의 패턴 생성 단계가 서로 논리적으로 연결되어 상기 테스트 패턴 프로그램의 각 프레임의 해당되는 상기 인터리브 사이클에 대응하여 데이터 연산을 각각 수행하는 단계를 포함하며,

상기 인터리브 사이클의 마지막 인터리브 사이클에 대응하는 데이터 연산 결과는 첫 번째 인터리브 사이클에 대응하는 데이터 연산에 피드백되는 것인 반도체 테스트를 위한 패턴 생성 방법.

청구항 12.

제10항에 있어서, 상기 단계 (b)는

(b-2) 다수의 필드에 대해서 각각 데이터 연산을 수행하는 단계

를 포함하는 것인 반도체 테스트를 위한 패턴 생성 방법.

청구항 13.

제10항에 있어서,

상기 테스트 패턴 프로그램은 다른 필드를 참조하는 데이터 연산에 대해서는 이전 프레임에서의 상기 다른 필드의 연산 결과를 참조하여 현재 프레임의 각 인터리브 사이클에 대응하는 데이터 연산이 상기 미리 지정된 데이터 예측 조건에 적합하도록 데이터 연산을 예측하여 컴파일되는 것인 반도체 테스트를 위한 패턴 생성 방법.

청구항 14.

제10항에 있어서, 상기 단계 (a) 이전에,

(d) 소스 코드를 판독하여 상기 소스 코드가 상기 미리 지정된 데이터 예측 조건에 적합한 데이터 연산만으로 작성되었는지 검사하는 단계와,

(e) 상기 단계 (d)에서 상기 미리 지정된 데이터 예측 조건에 적합한 데이터 연산만으로 작성된 것으로 판단한 경우 상기 소스 코드를 상기 테스트 패턴 프로그램으로 컴파일하는 단계

를 더 포함하는 반도체 테스트를 위한 패턴 생성 방법.

청구항 15.

제14항에 있어서,

(f) 상기 단계 (d)에서 상기 미리 지정된 데이터 예측 조건에 적합한 데이터 연산만으로 작성되지 않은 것으로 판단한 경우, 상기 소스 코드의 컴파일을 수행하지 않는 단계

를 더 포함하는 반도체 테스트를 위한 패턴 생성 방법.

청구항 16.

제15항에 있어서, 상기 단계 (e)는,

(e-1) 상기 소스 코드를 상기 인터리브 사이클에 대하여 미리 지정된 데이터 예측 조건을 기초로 변환하여 상기 테스트 패턴 프로그램으로 컴파일하는 단계

를 포함하는 것인 반도체 테스트를 위한 패턴 생성 방법.

청구항 17.

제10항에 있어서,

상기 미리 지정된 데이터 예측 조건에 적합한 데이터 연산은 다음 표에 의해서 정의되는 연산인 것인 반도체 테스트를 위한 패턴 생성 방법.

		이전 코드												
		A=A	A=/A	A=A*2	A=A/2	A=B	A=A+B	A=A-B	A=A&B	A=A B	A=A^B	A=imm	A=A+imm	A=A-imm
현재 코드	A=A	A=A	A=/A	A=C	A=C	A=B	A=A+B	A=A-B	A=A&B	A=A B	A=A^B	A=imm	A=C	A=C
	A=/A	A=/A	A=A	X	X	X	X	X	X	X	X	X	X	X
	A=A*2	A=C*2	X	A=C*2	A=C	X	X	X	X	X	X	X	X	X
	A=A/2	A=C/2	X	A=C	A=C/2	X	X	X	X	X	X	X	X	X
	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B
	A=A+B	A=A+B	X	X	X	X	X	X	X	X	X	X	X	X
	A=A-B	A=A-B	X	X	X	X	X	X	X	X	X	X	X	X
	A=A&B	A=A&B	X	X	X	X	X	X	X	X	X	X	X	X
	A=A B	A=A B	X	X	X	X	X	X	X	X	X	X	X	X
	A=A^B	A=A^B	X	X	X	X	X	X	X	X	X	X	X	X
	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm
	A=A+imm	A=C+imm	X	X	X	X	X	X	X	X	X	X	A=C+imm	A=C+imm
	A=A-imm	A=C-imm	X	X	X	X	X	X	X	X	X	X	A=C-imm	A=C-imm

(여기서 'A', 'B'는 데이터 필드, 'C'는 이전에 컴파일된 코드, 'X'는 예측되지 않는 경우를 나타내며, 'imm'는 즉시값, '/'는 나누기 연산자, '*'는 곱하기 연산자, '+'는 더하기 연산자, '-'는 빼기 연산자, '&'는 bitwise AND 연산자, '|'는 bitwise OR 연산자, '^'는 bitwise XOR 연산자를 각각 나타냄)

청구항 18.

제10항에 있어서,

상기 인터리브 방식은 2개 또는 4개의 패턴 생성 단계를 인터리브 방식으로 연결한 것인 반도체 테스트를 위한 패턴 생성 방법.

명세서

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 반도체 테스트를 위한 패턴 생성 장치 및 패턴 생성 방법에 관한 것으로, 더욱 구체적으로는 테스트 수행자가 테스트를 위한 소스 코드를 작성시 패턴 생성을 위한 내부 연산을 인터리브 동작 각각에 대해서 구체적으로 분석하여 작성하지 않더라도 데이터 연산을 예측하여 테스트 프로그램이 컴파일되어 인터리브 방식을 통하여 테스트 패턴 데이터의 생성이 가능한 반도체 테스트를 위한 패턴 생성 장치 및 패턴 생성 방법에 관한 것이다.

반도체 소자 테스트 장치는 제조된 반도체 소자의 불량 여부를 테스트하는 장치이다. 이러한 반도체 소자 테스트 장치는 특히 메모리 소자의 테스트를 위해서 사용되는 경우가 많기 때문에 메모리 소자의 개발 상황, 특히 메모리 소자의 상당 부분을 차지하는 DRAM의 개발 상황에 따라서 설계되어 개발된다.

현재의 DRAM 발전 상황은 EDO(Extended Data Output) 기능을 탑재한 DRAM이나, SDRAM(Synchronous DRAM), 램버스(Rambus) DRAM에 이어서 DDR(Double Data Rate) DRAM으로 발전되고 있다.

이러한 DRAM을 테스트하기 위해서는 메모리의 고속화에 대응하여 반도체 소자 테스트 장치도 고속 및 고정밀도가 요구된다. 또한 메모리의 대용량화에 따라 테스트 시간이 증가하게 되므로 테스트의 속도 역시 빨라져야 한다. 또한 소형화되고 경제적인 반도체 소자 테스트 장치를 구현하여 테스트 비용을 절감할 수 있어야 한다.

반도체 소자 테스트 장치, 특히 그 중에서도 메모리 테스트 장치는 전형적으로 메모리 컴포넌트 또는 SIMM 또는 DIMM 구성으로 되어 있는 메모리 모듈을 테스트하고 검증하는데 사용된다. 이러한 반도체 소자 테스트 장치는 메모리 모듈 또는 메모리 컴포넌트가 실제 컴퓨터 시스템 등에 장착되어 사용되기 전에 메모리 모듈 또는 컴포넌트 상의 기능상 결함이 존재하는지의 여부를 검출하게 된다.

반도체 소자 테스트 장치는 크게 하드웨어 반도체 소자 테스트 장치와 PC 환경에서 실행되는 소프트웨어 진단 프로그램으로 구별될 수 있다. 그러나 소프트웨어 진단 프로그램은 메모리 모듈 또는 컴포넌트가 실제 컴퓨터 상에 장착되어 사용되는 경우 메모리의 상태를 진단하기 때문에 반도체 메모리 생산과정에서는 하드웨어 메모리 테스트 장치를 주로 사용하게 된다.

이러한 하드웨어 반도체 소자 테스트 장치는 ATE(automatic test equipment)라고 불리는 고급 사양의 테스트 장치와, 중급(medium range) 메모리 테스트 장치, 저급(low-end) 메모리 테스트 장치 등으로 구분할 수 있다.

메모리 소자의 테스트 공정을 수행하기 위해 전형적으로 고급 사양의 테스트 장치인 ATE를 사용한다. 이러한 종래의 ATE는 DC 파라미터들이 회로의 디지털적 동작에 적합한 지를 테스트하는 DC 테스트, 신호의 전달 지연 시간, 셋업(set-

up) 시간과 홀드(hold) 시간 등과 관련된 AC 마진(margin) 테스트를 포함하며 이를 위한 테스트 패턴의 생성과 타이밍 생성 등 다양한 기능을 가지고 있다. 그러나 메인프레임 등 부피가 크고 고가인 전용 장비를 사용하여 제작되는 것이므로 제작 비용이 높은 단점이 있다.

도 1은 종래 기술에 반도체 소자 테스트 장치의 예시적인 블록도이다.

도시되듯이 종래의 반도체 소자 테스트 장치는 패턴 생성부(110)와, 타이밍 생성부(120)와, 포맷 제어부(130)와, 드라이버부(140)와, 비교부(150)와, 테스트 결과 저장부(160)를 포함한다. 이러한 구성 요소 이외에도 예컨대 DC 테스트를 위한 전원 제어부 구성이나 클럭 생성을 위한 구성이나 제어를 위한 구성, 테스트될 반도체 소자인 DUT(180)의 동작을 위해서 전원을 공급하는 구성이나 DUT(180)로 테스트 패턴 데이터를 중계하고 DUT(180)로부터 테스트 수행 결과를 수신하는 구성이나 외부로부터 테스트 패턴 프로그램을 수신하는 구성이나 테스트 결과를 외부로 전송하는 구성 등이 포함될 수 있으나 이에 대한 설명은 생략한다.

패턴 생성부(110)는 테스트 패턴 프로그램을 기초로 DUT(180)의 테스트에 필요한 테스트 패턴 데이터를 생성한다. 테스트 패턴 프로그램은 예컨대 테스트를 수행하기 위해서 다양한 형태의 동작을 수행하는 인스트럭션을 포함하도록 작성되며, 패턴 생성부(110)는 이러한 테스트 패턴 프로그램을 예컨대 외부의 저장 장치 등으로부터 수신하여 이를 해석하여 테스트 패턴 데이터를 생성한다. 이러한 테스트 패턴 데이터는 DUT(180)에 입력되는 명령어, 주소, 데이터 등의 데이터를 포함하며, 또한 생성되는 테스트 패턴 데이터와 대응하여 테스트 기대치 데이터가 생성된다.

타이밍 생성부(120)는 패턴 생성부(110)에서 생성된 테스트 패턴 데이터를 다양한 형태의 파형으로 변환하기 위한 기준이 되는 타이밍 에지(edge)를 생성한다. 이러한 타이밍 에지는 원활한 파형 변환을 위해서 다수의 클럭을 이용하여 생성된다.

포맷 제어부(130)는 테스트 패턴 데이터를 타이밍 에지를 기초로 변환하여 원하는 파형으로 변환한다.

드라이버부(140)는 변환된 테스트 파형을 DUT(180)에 전송하기 위한 구성이다.

비교부(150)는 DUT(180)에 인가된 테스트 파형에 의해서 DUT(180)의 동작이 수행된 후 DUT(180)로부터 출력된 테스트 수행 데이터와 패턴 생성부(110)에서 생성되는 테스트 기대치 데이터와 비교하여 DUT(180)의 테스트를 수행한다.

테스트 결과 저장부(160)는 비교부(150)의 결과를 기초로 테스트 결과를 저장한다. 예컨대 불량 발생 메모리 소자에 대한 정보를 저장한다.

전술한 바와 같이 이러한 종래의 ATE는 매우 고가의 장비이므로 반도체 제조업체의 측면에서 생산 비용을 최소화하여 경쟁력을 높이기 위해서는 이러한 고비용의 ATE를 효율적으로 설계하는 것이 바람직하다. 이러한 반도체 소자 테스트 장치 특히 ATE의 효율적인 설계를 위해서는 테스트 패턴의 생성 기능을 최적화하는 것이 필요하다.

특히 고속으로 동작하는 반도체 소자의 테스트를 수행하기 위해서는 테스트 패턴 데이터의 생성 역시 이러한 반도체 소자의 동작에 대응하여 생성되어야 한다. 그러나 테스트 패턴 데이터를 생성하는 패턴 생성부(110)는 일반적으로 저속으로 동작하는 프로그램 가능한 로직 소자, 예컨대 FPGA 등을 사용되어 구현하는 경우가 많다. 따라서 패턴 생성부(110)의 구현에 있어서, 알고리즘 패턴 생성부(algorithm pattern generator, ALPG)를 다수개 사용하여 고속으로 동작하는 반도체 소자의 테스트 패턴을 생성하는 방식, 즉 인터리브(interleave) 방식을 사용하는 경우가 존재한다.

이러한 패턴 생성부(110)는 본원 발명의 명세서에서 패턴 생성 장치(pattern generator)라고도 지칭된다.

이러한 다수의 알고리즘 패턴 생성부를 이용한 패턴 생성 장치에 대해서는 예컨대 안도덴키 가부시기가이샤에 의해서 1996년 11월 26일자로 출원되고 1999년 6월 19일자로 등록된 "패턴 발생 회로"라는 명칭의 특허등록번호 제10-220201호 또는 가부시기가이샤 아드반테스트에 의해서 1997년 1월 14일자로 출원되고 1999년 11월 14일자로 등록된 "반도체 시험 장치"라는 명칭의 특허등록번호 제10-242604호에 개시되어 있다.

도 2는 종래 기술에 따른 패턴 생성 장치의 예시적인 블록도이다.

도시되듯이 종래의 패턴 생성 장치는 시퀀스 제어부(210)와, 다수의 알고리즘 패턴 생성부(220a 내지 220d)와, 다중화부(230)를 포함한다. 이 외에도 테스트 패턴 프로그램을 저장하기 위한 인스트럭션 메모리 등의 구성요소를 더 포함할 수 있으나 이에 대한 도시 및 설명은 생략한다.

시퀀스 제어부(210)는 테스트 패턴 프로그램의 인스트럭션을 페치(fetch)하여 인스트럭션 포인터(instruction pointer, IP)를 각 알고리즘 패턴 생성부(220a 내지 220d)에 전달한다.

알고리즘 패턴 생성부(220a 내지 220d)는 시퀀스 제어부(210)에서 페치된 인스트럭션 포인터를 기초로 자신에게 지정된 데이터 연산을 수행하여 출력한다.

출력되는 데이터는 예컨대 전술한 테스트 패턴 데이터를 이루는 DUT에 입력되는 명령어, 주소, 데이터 등의 데이터를 포함하며, 또한 생성되는 테스트 패턴 데이터와 대응하여 테스트 기대치 데이터가 생성될 수 있다. 다만 알고리즘 패턴 생성부(220a 내지 220d) 각각은 저속으로 동작하는 소자를 이용하여 구현되므로 이러한 저속으로 동작하는 소자의 출력 데이터를 DUT에 적합한 형태로 인가하기 위한 구성이 필요하다.

다중화부(230)는 각 알고리즘 패턴 생성부(220a 내지 220d)의 연산 출력, 예컨대 A1 내지 A4를 다중화, 즉 직렬화(serialize)하여 DUT에 적합한 테스트 패턴 데이터를 생성한다.

즉 알고리즘 패턴 생성부(220a 내지 220d)에서는 병렬적으로 데이터 연산을 수행하며 이러한 병렬적으로 수행된 데이터 연산 결과는 이후 다중화부(230)에서 합성되어 테스트 패턴 데이터로 출력된다.

예컨대 각 알고리즘 패턴 생성부(220a 내지 220d)가 FPGA로 구현된 경우 대략 300 MHz 까지의 동작 속도를 가진다고 하면, 각 알고리즘 패턴 생성부(220a 내지 220d)의 출력은 300 MHz 정도의 동작에 적합한 데이터이다. 이러한 데이터를 다중화부(230)를 통하여 다중화하는 경우 대략 1.2 GHz의 주파수에서 동작하는 반도체 소자의 테스트를 위한 테스트 패턴 데이터를 생성할 수 있다.

이러한 패턴 생성 장치에 있어서 "프레임"은 알고리즘 패턴 생성부(220a 내지 220d) 전체의 동작 주기를 의미한다. 또한 "사이클"은 알고리즘 패턴 생성부(220a 내지 220d)의 각 패턴 라인이 수행되는 주기를 의미한다.

예컨대 전술한 인터리브 방식을 사용하지 않는 경우라면 프레임과 사이클은 동일하지만, 인터리브 방식을 사용하는 경우라면, 예컨대 4개의 알고리즘 패턴 생성부를 인터리브 방식을 이용하여 하나의 패턴 생성 장치처럼 사용하는 경우 4개의 사이클이 1 프레임을 구성한다.

따라서 1 프레임 동안에 의해서 각 알고리즘 패턴 생성부(220a 내지 220d)가 각 사이클에 대응하여 데이터 연산을 수행하고 이를 합성하여 테스트 패턴 데이터를 생성할 수 있다.

이러한 종래 구조에 따른 패턴 생성 장치에 있어서, 각 알고리즘 패턴 생성부(220a 내지 220d)는 하드웨어적으로 서로 구분되어 있으며 따라서 서로 독립적으로 동작한다. 그러나 비록 하드웨어 적으로 서로 구분되는 알고리즘 패턴 생성부(220a 내지 220d)일 지라도 각 사이클 내에서 서로 연산 결과를 예측하여 사용할 수 있다면 알고리즘 패턴 생성부(220a 내지 220d)의 동작 효율이 더 높아질 수 있으며 또한 비록 하드웨어적으로 서로 구분되어 있더라도 하나의 알고리즘 패턴 생성부로 간주할 수 있으므로 테스트 패턴 데이터를 효율적으로 생성할 수 있다.

따라서 이러한 이유로 패턴 생성 장치의 각 알고리즘 패턴 생성부(220a 내지 220d)는 서로 소프트웨어적으로 연관되도록 구성한다.

이 경우 소프트웨어적인 연관성을 구현하는 것은 다음과 같은 방법이 사용될 수 있다.

우선 테스트 패턴 프로그램을 작성시 테스트 수행자가 각 사이클에 대해서 알고리즘 패턴 생성부(220a 내지 220d)의 연산을 각각 독립적으로 계산하도록 테스트 프로그램을 작성하고 이를 다중화부에서 적절히 조합하여 테스트 패턴 데이터를 생성하는 것이다.

이러한 테스트 수행자에 의한 연관 관계 계산 및 이를 통한 연관 구성은 구현이 간단하고 연관 가능한 인스트럭션, 즉 데이터 연산에 제한이 없다는 장점이 있다.

그러나 이러한 경우 테스트 수행자는 직관적으로 테스트 패턴 프로그램을 작성하지 못하고 각 내부의 연산 관계를 고려하여 테스트 프로그램을 작성하게 되므로 테스트 프로그램의 작성이 어렵다는 단점이 있다. 즉 알고리즘 패턴 생성부(220a

내지 220d)에 구비되는 각 연산 레지스터 사이에 연관 관계가 없으므로 테스트 패턴 프로그램을 작성하는 사용자가 관련된 모든 사이클의 데이터 연산을 독립적으로 계산하여야 하므로 이를 고려하여 테스트 패턴 프로그램을 작성하는 복잡한 작업이 수행되어야 하는 단점이 있다.

또한 테스트 패턴 프로그램의 컴파일에 있어서 테스트 수행자가 작성한 테스트 프로그램의 연산을 사이클 단위로 미리 예측하여 각 알고리즘 패턴 생성부(220a 내지 220d)에서 연관 관계를 가지도록 변환하여 컴파일할 수 있다. 이러한 경우 테스트 수행자는 각 알고리즘 패턴 생성부(220a 내지 220d)를 하나의 알고리즘 패턴 생성부로 간주하여 테스트 패턴 프로그램을 작성할 수 있다.

예컨대 테스트 수행자가 작성한 테스트 패턴 프로그램의 연산 코드로 직접 컴파일하는 것이 아니라 각 알고리즘 패턴 생성부(220a 내지 220d)가 수행하여야 할 연산을 이전 사이클과 현재 사이클 등 사이클 단위로 예측하여 예측 결과를 기초로 새로운 연산 코드로 변환하여 컴파일을 수행하며, 알고리즘 패턴 생성부(220a 내지 220d)는 이러한 방식으로 변환된 테스트 패턴 프로그램을 기초로 연산을 수행하는 것이다. 이러한 경우는 테스트 수행자의 테스트 패턴 프로그램 작성은 간편하지만 하드웨어 적인 구현이 어렵고 또한 사용할 수 있는 인스트럭션에 제한이 있다는 단점이 있다.

이러한 경우 특히 테스트 패턴 데이터의 생성을 위해서 다수의 필드를 이용한 데이터 연산을 수행하는 경우라면 데이터 연산을 예측하여 변환하기에 적합한, 즉 사용할 수 있는 인스트럭션의 개수가 더욱 제한될 수밖에 없으며 또한 다수의 필드를 이용하기 때문에 이전 프레임의 사이클에서의 연산 코드와 현재 프레임에서의 사이클에서의 연산 코드 사이에도 제한되는 인스트럭션의 개수가 많아지게 된다.

발명이 이루고자 하는 기술적 과제

본 발명의 목적은 테스트 수행자가 테스트를 위한 소스 코드를 작성시 패턴 생성을 위한 내부 연산을 인터리브 동작 각각에 대해서 구체적으로 분석하여 작성하지 않더라도 데이터 연산을 예측하여 테스트 프로그램이 컴파일되어 인터리브 방식을 통하여 테스트 패턴 데이터의 생성이 가능한 반도체 테스트를 위한 패턴 생성 장치 및 패턴 생성 방법을 제공하는 데 있다.

발명의 구성

상기 기술적 과제를 달성하기 위하여, 본 발명은 서로 다른 프레임에 대해서는 제약 조건 없이 모든 데이터 연산이 가능하고 동일한 프레임 내의 인터리브 사이클에 대해서는 미리 지정된 데이터 예측 조건에 적합한 데이터 연산만이 가능하도록 데이터 연산을 예측하여 컴파일된 테스트 패턴 프로그램을 판독하는 테스트 패턴 프로그램 판독부와, 상기 테스트 패턴 프로그램을 기초로 인터리브 방식으로 데이터 연산을 수행한 후 패턴을 생성하는 다수의 알고리즘 패턴 생성부와, 상기 패턴을 다중화하여 반도체 소자의 테스트를 위한 테스트 패턴 데이터를 생성하는 다중화부를 포함하는 반도체 테스트를 위한 패턴 생성 장치를 제공한다.

또한 본 발명은 (a) 서로 다른 프레임에 대해서는 제약 조건 없이 모든 데이터 연산이 가능하고 동일한 프레임 내의 인터리브 사이클에 대해서는 미리 지정된 데이터 예측 조건에 적합한 데이터 연산만이 가능하도록 데이터 연산을 예측하여 컴파일된 테스트 패턴 프로그램을 판독하는 단계와, (b) 상기 테스트 패턴 프로그램을 기초로 인터리브 방식으로 데이터 연산을 수행하여 패턴을 생성하는 단계와, (c) 상기 패턴을 다중화하여 반도체 소자의 테스트를 위한 테스트 패턴 데이터를 생성하는 단계를 포함하는 반도체 테스트를 위한 패턴 생성 방법을 제공한다.

이하, 본 발명의 반도체 테스트를 위한 패턴 생성 장치 및 패턴 생성 방법을 첨부한 도면을 참조로 보다 구체적으로 설명한다.

도 3은 본 발명에 따른 반도체 테스트를 위한 패턴 생성 장치의 예시적인 블록도이다.

도시되듯이 본 발명에 따른 반도체 테스트를 위한 패턴 생성 장치는 테스트 패턴 프로그램 판독부(310)와, 다수의 알고리즘 패턴 생성부(320a 내지 320d)와, 다중화부(330)를 포함한다.

테스트 패턴 프로그램 판독부(310)는 테스트 패턴 프로그램을 판독한다.

테스트 패턴 프로그램은 예컨대 본 발명에 따른 반도체 테스트를 위한 패턴 생성 장치와 연결되는 테스트 패턴 프로그램 처리 장치로부터 생성된다.

테스트 패턴 프로그램 처리 장치는 예컨대 통신 네트워크를 통하여 본 발명에 따른 반도체 테스트를 위한 패턴 생성 장치와 연결되며 테스트 수행자가 테스트를 수행하기 위한 소스 코드를 작성하고 이를 컴파일하여 테스트 패턴 프로그램을 생성할 수 있도록 구현된 장치이다.

바람직하게는 본 발명에 따른 반도체 테스트를 위한 패턴 생성 장치를 구비하는 반도체 소자 테스트 장치에 테스트 패턴 프로그램 처리 장치가 구비될 수 있으나 반도체 소자 테스트 장치와는 별도로 테스트 패턴 프로그램 처리 장치가 구비될 수도 있다.

테스트 패턴 프로그램은 서로 다른 프레임에 대해서는 제약 조건 없이 모든 데이터 연산이 가능하고 동일한 프레임 내의 인터리브 사이클에 대해서는 미리 지정된 데이터 예측 조건에 적합한 데이터 연산만이 가능하도록 데이터 연산을 예측하여 컴파일된 것을 특징으로 한다.

즉 종래의 경우 테스트 패턴 프로그램이 동일한 프레임 내에서 뿐만 아니라 서로 다른 프레임에 대해서도 서로 데이터 예측을 기초로 인터리브를 구현하도록 구성되어 있다. 그러나 이러한 방식은 이전 프레임의 마지막 인터리브 사이클에 대응하는 데이터 연산 코드와 현재 프레임의 최초 데이터 연산 코드까지도 서로 데이터 예측을 기초로 인터리브를 구현하기 때문에 서로 다른 프레임에 대한 데이터 연산에 관련되어 사용할 수 있는 인스트럭션, 즉 데이터 연산에 제한되는 경우가 많은 단점이 있었다.

그러나 본 발명에 따른 반도체 테스트를 위한 패턴 생성 장치에서는 테스트 프로그램이 서로 다른 프레임에 대해서는 모든 데이터 연산이 제한없이 가능하고 동일한 프레임 내의 인터리브 사이클에 대해서는 미리 지정된 데이터 예측 조건에 적합한 데이터 연산만이 가능하도록 데이터 연산을 예측하여 컴파일하기 때문에, 데이터 연산의 제한은 동일한 프레임 내의 각 인터리브 사이클에 대해서만 존재한다. 따라서 예측을 통한 인터리브 수행에 제한되는 데이터 연산은 이전의 방식에 비해서 감소하게 된다.

이러한 테스트 패턴 프로그램을 예컨대 인스트럭션 메모리 등에 저장한 이후 테스트 패턴 프로그램 관독부(310)는 테스트 패턴 데이터의 생성을 위해서 관독한다.

이러한 관독은 예컨대 시퀀스 제어 등을 통하여 테스트 패턴 프로그램으로부터 테스트 패턴 데이터의 생성을 위한 인스트럭션을 폐지하거나 데이터를 폐지하는 등의 과정을 포함한다.

다수의 알고리즘 패턴 생성부(320a 내지 320d)는 테스트 패턴 프로그램을 기초로 인터리브 방식으로 데이터 연산을 수행한 후 패턴을 생성한다. 비록 알고리즘 패턴 생성부(320a 내지 320d)는 4개를 포함하도록 도시되지만 2개를 포함할 수도 있으며 또는 3개를 포함할 수도 있다. 그러나 효율적인 인터리브 수행을 위해서는 2개 또는 4개를 포함하는 것이 바람직하다.

인터리브 방식은 테스트 패턴 프로그램에서 각 프레임을 기초로 수행된다. 즉 각 프레임에는 알고리즘 패턴 생성부(320a 내지 320d)의 개수에 대응한 인터리브 사이클이 포함되며, 알고리즘 패턴 생성부(320a 내지 320d) 각각은 자신에게 해당하는 인터리브 사이클에 대응하는 데이터 연산을 병렬적으로 각각 수행한다. 이러한 방식으로 다수의 알고리즘 패턴 생성부(320a 내지 320d) 각각은 비록 서로간에 하드웨어적으로는 구분되도록 구성되더라도 소프트웨어적으로, 즉 논리적으로는 서로 연결되어 인터리브 방식을 구현할 수 있다.

이 경우 도시되듯이 마지막 인터리브 사이클에 대응하는 데이터 연산 결과는 첫 번째 인터리브 사이클에 대응하는 데이터 연산에 피드백된다. 이러한 피드백을 통하여 각 프레임은 서로 연관될 수 있으며, 이러한 연관 관계의 구성을 통하여 프레임 사이에는 데이터 연산을 예측하는데 있어서 제약 조건이 없도록 구현이 가능하다. 즉 종래의 경우 프레임과 프레임 사이에도 예측을 통한 인터리브 방식에 있어서 데이터 연산이 제한되는 경우가 있는 반면에 본 발명에 따른 패턴 생성 장치의 경우 이러한 데이터 연산의 제한은 피드백을 통하여 해소될 수 있다.

또한 다수의 알고리즘 패턴 생성부(320a 내지 320d) 각각은 도시되듯이 다수의 필드, 예컨대 필드 A 및 필드 B에 대해서 각각 데이터 연산을 수행할 수 있다. 이 경우 필드 B에 대한 데이터 연산 결과는 필드 A에 대한 데이터 연산에 영향을 줄 수 있다.

이러한 다수의 필드에 대한 연산을 위해서 알고리즘 패턴 생성부(320a 내지 320d) 각각은 다수의 연산 레지스터를 포함할 수 있으며, 이 경우 동일한 프레임 내에서의 각 인터리브 사이클에 대한 데이터 연산은 다수의 연산 레지스터에서 각각 다수의 필드에 대해서 병렬적으로 수행된다.

이러한 경우 본 발명에 따른 반도체 테스트를 위한 패턴 생성 장치에 사용되는 테스트 패턴 프로그램은 필드 A 또는 필드 B 각각에 대한 연산은 동일한 프레임 내에서는 각 인터리브 사이클마다 연관되도록 예측하여 컴파일된다.

그러나 만약 예컨대 필드 A에 대한 데이터 연산에서 필드 B를 참조하는 경우라면 다음과 같이 수행된다. 필드 A에서의 데이터 연산 수행시 필드 B에 대한 데이터 연산 결과를 참조하려면 이전 프레임에서의 필드 B에 대한 데이터 연산 결과를 현재 프레임에서의 필드 A에서의 데이터 연산에서 참조한다. 이러한 이전 프레임의 결과를 참조하는 것에 의해서, 현재 프레임에서의 각 인터리브 사이클에서는 미리 지정된 데이터 예측 조건에 적합하도록 데이터 연산을 예측하여 수행이 가능하며, 테스트 패턴 프로그램은 이러한 서로 다른 필드를 참조하는 연산에 대해서 이전 프레임을 참조하도록 컴파일된다.

즉 도시되듯이 알고리즘 패턴 생성부(320a)의 경우 필드 B에 대해서 연산을 수행하여 B1 값이 출력되며, 이는 다음 프레임에서 필드 A에서 참조하여 연산되어 A1 값이 출력된다. 또한 다수의 알고리즘 패턴 생성부(320a 내지 320d)의 출력 A1 내지 A4는 이후 다중화되어 테스트 패턴 데이터로서 출력된다.

이와 같이 동일한 필드 내에서의 연산의 경우는 물론이고 동일한 필드가 아닌 다른 필드의 데이터를 참조하는 경우라도 다른 필드의 현재 프레임에서의 값이 아니라 이전 프레임에서의 연산 결과를 참조하는 것에 의해서 적어도 연관 관계를 통한 예측에서는 현재 프레임, 즉 동일한 프레임 내에서 연관 관계를 가지도록 구현할 수 있다. 이러한 동일한 프레임에서의 연관 관계 구현은 현재 프레임에 적용 가능한 인스트럭션, 즉 데이터 연산의 제한을 감소시킬 수 있다.

도 4는 본 발명에 따른 반도체 테스트를 위한 패턴 생성 장치에 있어서 다수의 필드에 대한 동작을 설명하기 위한 도면이다.

도 4에는 테스트 패턴 프로그램 관독부에 의해서 관독된 테스트 패턴 프로그램의 각 시퀀스와 시퀀스에 대응하여 다수의 알고리즘 패턴 생성부, 예컨대 4개의 알고리즘 패턴 생성부에 대해서 테스트 수행자가 작성한 데이터 연산 코드가 도시된다. 또한 다수의 시퀀스에 대해서 이전 프레임, 현재 프레임, 이후 프레임이 도시된다.

도 4에서 화살표는 각 데이터 연산마다 영향을 미치는 관계, 즉 연관 관계를 나타낸다.

도시되듯이 첫 번째 프레임("이전 프레임"으로 표시됨)에 대해서는 4개의 알고리즘 패턴 생성부 각각에서 필드 A에 대해서는 $A=A+1$ 이라는 연산을 수행하도록 테스트 수행자에 의해서 소스 코드가 작성되었으며, 마찬가지로 필드 B에 대해서는 $B=B+1$ 이라는 연산을 수행하도록 테스트 수행자에 의해서 소스 코드가 작성되었다. 이 경우 각 $A=A+1$ 연산 또는 $B=B+1$ 연산의 결과는 도시되듯이 다음 인터리브 사이클, 즉 다음의 알고리즘 패턴 생성부의 연산에 영향을 미치게 된다.

또한 두 번째 프레임("현재 프레임"으로 표시됨)에 대해서는 4개의 알고리즘 패턴 생성부 각각에서 필드 A에 대해서는 $A=B$ 이라는 연산을 수행하도록 테스트 수행자에 의해서 데이터 연산 코드가 작성되었으며, 마찬가지로 필드 B에 대해서는 $B=B-1$ 이라는 연산을 수행하도록 테스트 수행자에 의해서 소스 코드가 작성되었다. 이 경우 각 $A=B$ 연산의 결과는 도시되듯이 다음 연산에 영향을 미치게 되며, 이전 프레임에서의 필드 B에 대한 연산의 결과가 도시되듯이 현재 프레임에서의 필드 A에 대한 연산에 참조되어 연산된다. 또한 필드 B에 대해서는 다른 필드를 참조하지 않기 때문에 현재 프레임 내에서 각 인터리브 사이클에 대해서 서로 예측을 통하여 데이터 연산이 가능하도록 구성할 수 있다.

또한 세 번째 프레임("이후 프레임"으로 표시됨)에 대해서는 4개의 알고리즘 패턴 생성부 각각에서 필드 A에 대해서는 $A=B$ 이라는 연산을 수행하도록 테스트 수행자에 의해서 소스 코드가 작성되었으며, 마찬가지로 필드 B에 대해서는 $B=B*2$ 이라는 연산을 수행하도록 테스트 수행자에 의해서 소스 코드가 작성되었다. 이 경우 각 $A=B$ 연산의 결과는 도시되듯이 다음 연산에 영향을 미치게 되며, 현재 프레임에서의 필드 B에 대한 연산의 결과가 도시되듯이 이후 프레임에서의 필드 A에 대한 연산에 참조되어 연산된다. 또한 필드 B에 대해서는 다른 필드를 참조하지 않기 때문에 현재 프레임 내에서 각 인터리브 사이클에 대해서 서로 예측을 통하여 데이터 연산이 가능하도록 구성할 수 있다.

이러한 도 4의 설명은 본 발명에 따른 인터리브 방식의 설명을 위해서 테스트 수행자가 작성한 소스 코드를 기준으로 하였지만 이는 실제 본 발명에 따른 패턴 생성 장치의 알고리즘 패턴 생성부에서 실제 수행되는 동작과는 거리가 있으며, 실제 동작에 대해서는 이후 도 5를 참조로 설명한다.

다중화부(330)는 다수의 알고리즘 패턴 생성부(320a 내지 320d)의 출력인 다수의 패턴을 다중화하여 고속으로 동작하는 반도체 소자의 테스트를 위한 테스트 패턴 데이터를 생성한다.

또한 도시되지는 않았지만 본 발명에 따른 반도체 테스트를 위한 패턴 생성 장치는 전술하듯이 테스트 패턴 프로그램 처리 장치와 연결된다.

테스트 패턴 프로그램 장치는 테스트 수행자가 테스트를 수행하기 위한 소스 코드를 작성하고 이를 컴파일하여 테스트 패턴 프로그램을 생성할 수 있도록 다음과 같은 구성을 포함한다.

우선 사용자가 작성한 소스 코드를 테스트 패턴 프로그램으로 컴파일하는 컴파일러를 포함하며, 또한 컴파일시 소스 코드가 미리 지정된 데이터 예측 조건에 적합한 데이터 연산 만으로 구성되었는지 검사하는 테스트 패턴 프로그램 검사부를 포함한다.

이 경우 테스트 패턴 프로그램 검사부에서 소스 코드가 미리 지정된 데이터 예측 조건에 적합한 데이터 연산 만으로 구성된 것으로 판단한 경우라면 컴파일러는 컴파일을 수행한다.

컴파일러는 소스 프로그램으로부터 실행 가능한 이진 코드를 생성하는 것이다. 본 발명에 따른 테스트 패턴 프로그램의 컴파일은 테스트 수행자가 작성한 소스 코드를 그대로 컴파일하는 것이 아니라 이를 미리 지정된 데이터 예측 조건을 기초로 변환하여 수행되는 것이 바람직하다.

또한 테스트 패턴 프로그램 검사부에서 소스 코드가 미리 지정된 데이터 예측 조건에 적합한 데이터 연산 만으로 구성된 것이 아니라고 판단한 경우라면 컴파일러는 컴파일을 수행하지 않으며, 예컨대 에러 메시지를 출력하여 테스트 수행자가 소스 코드를 미리 지정된 데이터 예측 조건에 적합한 데이터 연산만으로 구성하도록 유도한다.

도 5는 본 발명에 따른 반도체 테스트를 위한 패턴 생성 장치에 있어서 테스트 프로그램의 변환 컴파일을 나타내는 도면이다.

도 5에는 테스트 패턴 프로그램 관독부에 의해서 관독된 테스트 패턴 프로그램의 각 시퀀스와 시퀀스에 대응하여 다수의 알고리즘 패턴 생성부, 예컨대 4개의 알고리즘 패턴 생성부에 대해서 테스트 수행자가 작성한 소스 코드("사용자 코드"로 표시됨)와, 소스 코드를 서로 연관되도록 예측하여 변환 컴파일한 코드("변환 컴파일된 코드"라 표시됨)가 도시된다. 또한 다수의 시퀀스에 대해서 이전 프레임, 현재 프레임, 이후 프레임에 대해서 소스 코드 및 이에 따른 변환 컴파일된 코드가 표시된다.

우선 "이전 프레임" 내에서는 첫 번째 알고리즘 패턴 생성부는 $A=A+1$ 연산을 수행하고, 두 번째 알고리즘 패턴 생성부는 첫 번째 알고리즘 패턴 생성부의 연산 결과를 받아서 $A=A+1$ 연산을 수행하는 것으로 소스 코드가 작성되어 있다. 소스 코드는 다수의 알고리즘 패턴 생성부가 서로 연관되어 있다고 가정하고 작성한 것이나 실제로는 하드웨어적으로는 연관되어 있지 않으므로 이를 변환하여야 한다.

즉 A의 값에 대해서, 첫 번째 알고리즘 패턴 생성부는 그대로 $A=A+1$ 을 수행하고 두 번째 알고리즘 패턴 생성부는 첫 번째 알고리즘 패턴 생성부의 연산 결과를 소프트웨어적으로 고려하여 $A=A+2$ 를 수행하고 마찬가지로 세 번째 알고리즘 패턴 생성부는 $A=A+3$ 을 수행하고 네 번째 알고리즘 패턴 생성부는 $A=A+4$ 를 수행하도록 소스 코드를 변환하여 컴파일하고 변환 컴파일 코드에 대응하는 데이터 연산을 각각 알고리즘 패턴 생성부에서 수행하는 것이다.

"현재 프레임"에 대해서는 이전 프레임에서의 마지막 데이터 연산값을 기초로 연산이 수행된다.

이 경우 첫 번째 알고리즘 패턴 생성부는 그대로 $A=A+8$ 을 수행하고 두 번째 알고리즘 패턴 생성부는 첫 번째 알고리즘 패턴 생성부의 연산 결과를 소프트웨어적으로 고려하여 $A=A-4$ 가 아니라 $A=A+4$ 를 수행하고 마찬가지로 세 번째 알

고리즘 패턴 생성부는 "A=A+ 1"이 아니라 "A=A+ 5"를 수행하고 네 번째 알고리즘 패턴 생성부는 "A=A+ 1"이 아니라 "A=A+ 6"을 수행하도록 소스 코드를 변환하여 컴파일하고 변환 컴파일 코드에 대응하는 데이터 연산을 각각 알고리즘 패턴 생성부에서 수행하는 것이다.

"이후 프레임"에서는 현재 프레임에서의 마지막 데이터 연산값을 기초로 연산이 수행된다.

이 경우 첫 번째 알고리즘 패턴 생성부는 그대로 "A=A*2"를 수행하고 두 번째 알고리즘 패턴 생성부는 첫 번째 알고리즘 패턴 생성부의 연산 결과를 소프트웨어적으로 고려하여 "A=A/2"가 아니라 "A=A"를 수행하고 마찬가지로 세 번째 알고리즘 패턴 생성부는 두 번째 알고리즘 패턴 생성부의 연산 결과를 소프트웨어적으로 고려하여 "A=A*2"를 수행하고 네 번째 알고리즘 패턴 생성부는 "A=A*2"가 아니라 "A=A*4"를 수행하도록 소스 코드를 변환하여 컴파일하고 변환 컴파일 코드에 대응하는 데이터 연산을 각각 알고리즘 패턴 생성부에서 수행하는 것이다.

이러한 데이터 연산을 위한 컴파일은 동일한 프레임 내에서는 이전 코드와 현재 코드 사이의 연관 관계를 고려하여 소스 코드를 변환하여 컴파일을 수행할 수 있다. 그러나 이러한 변환 컴파일은 모든 데이터 연산에 적용되는 것이 아니다.

다음 [표 1]은 본 발명에 따른 반도체 테스트를 위한 패턴 생성 장치에서 연관하여 예측 가능한 이전 코드와 현재 코드의 조합 및 예측되는 데이터 연산, 즉 미리 지정된 데이터 예측 조건을 나타낸다.

[표 1]

		이전 코드												
		A=A	A=/A	A=A*2	A=A/2	A=B	A=A+B	A=A-B	A=A&B	A=A B	A=A^B	A=imm	A=A+imm	A=A-imm
현재 코드	A=A	A=A	A=/A	A=C	A=C	A=B	A=A+B	A=A-B	A=A&B	A=A B	A=A^B	A=imm	A=C	A=C
	A=/A	A=/A	A=A	X	X	X	X	X	X	X	X	X	X	X
	A=A*2	A=C*2	X	A=C*2	A=C	X	X	X	X	X	X	X	X	X
	A=A/2	A=C/2	X	A=C	A=C/2	X	X	X	X	X	X	X	X	X
	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B	A=B
	A=A+B	A=A+B	X	X	X	X	X	X	X	X	X	X	X	X
	A=A-B	A=A-B	X	X	X	X	X	X	X	X	X	X	X	X
	A=A&B	A=A&B	X	X	X	X	X	X	X	X	X	X	X	X
	A=A B	A=A B	X	X	X	X	X	X	X	X	X	X	X	X
	A=A^B	A=A^B	X	X	X	X	X	X	X	X	X	X	X	X
	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm	A=imm
	A=A+imm	A=C+imm	X	X	X	X	X	X	X	X	X	X	A=C+imm	A=C+imm
	A=A-imm	A=C-imm	X	X	X	X	X	X	X	X	X	X	A=C-imm	A=C-imm

여기서 'A', 'B'는 데이터 필드, 'C'는 이전에 컴파일된 코드, 'X'는 예측되지 않는 경우를 나타내며, 'imm'는 즉시값, '/'는 나누기 연산자, '*'는 곱하기 연산자, '+'는 더하기 연산자, '-'는 빼기 연산자, '&'는 bitwise AND 연산자, '|'는 bitwise OR 연산자, '^'는 bitwise XOR 연산자를 각각 나타낸다.

도시되듯이 이전 컴파일된 코드 등을 기초로 현재 코드를 변환하여 데이터 연산을 수행하도록 구성이 가능하다.

도 6은 본 발명에 따른 반도체 테스트를 위한 패턴 생성 방법의 예시적인 흐름도이다.

우선 사전 준비 단계로서 테스트 수행자가 테스트 수행을 위해서 작성한 소스 코드를 테스트 패턴 프로그램으로 컴파일한다(S110).

테스트 패턴 프로그램은 소스 코드를 그대로 컴파일하는 것이 아니라, 서로 다른 프레임에 대해서는 제약 조건 없이 모든 데이터 연산이 가능하고 동일한 프레임 내의 인터리브 사이클에 대해서는 미리 지정된 데이터 예측 조건에 적합한 데이터 연산만이 가능하도록 데이터 연산을 예측하여 변환하여 컴파일된다. 이러한 변환 컴파일은 도 5 및 이에 대한 설명을 참조할 수 있다. 또한 이 경우 미리 지정된 데이터 예측 조건은 예컨대 표 1을 참조할 수 있다.

이 경우 소스 코드를 판독하여 소스 코드가 미리 지정된 데이터 예측 조건에 적합한 데이터 연산만으로 작성되었는지 검사하는 단계를 포함할 수 있다.

검사 결과 소스 코드가 미리 지정된 데이터 예측 조건에 적합하지 않은 경우라면 컴파일을 수행하지 않고 예컨대 오류 메시지를 출력할 수 있다.

또한 소스 코드가 미리 지정된 데이터 예측 조건에 적합한 경우라면 변환 컴파일을 수행할 수 있다.

이후 테스트 패턴 프로그램을 판독한다(S120).

즉 예컨대 외부의 PC 등을 통하여 테스트 수행자가 소스 코드를 작성하고 이를 컴파일한 후에는 반도체 소자 테스트 장치로 전송하여 저장하며, 이를 판독하여 테스트를 수행하는 것이다.

이후 테스트 패턴 프로그램을 인터리브 방식으로 데이터 연산을 수행하여 패턴을 생성한다(S130).

이러한 인터리브 방식의 패턴 생성은 다수의 패턴 생성 단계, 예컨대 2개 또는 4개의 패턴 생성 단계가 서로 논리적으로 연결되어 테스트 패턴 프로그램의 각 프레임의 해당되는 인터리브 사이클에 대응하여 데이터 연산을 각각 수행할 수 있다. 이 경우 인터리브 사이클의 마지막 인터리브 사이클에 대응하는 데이터 연산 결과는 첫 번째 인터리브 사이클에 대응하는 데이터 연산에 피드백되어 프레임 사이의 데이터 연산 예측을 통한 연관 관계 구현에 있어서 사용이 제한되는 데이터 연산이 없도록 구성할 수 있다.

또한 다수의 필드를 포함하는 데이터 연산인 경우라면 다수의 필드에 대해서 각각 데이터 연산을 수행할 수 있다.

이러한 경우 테스트 패턴 프로그램은 다른 필드를 참조하는 데이터 연산에 대해서는 이전 프레임에서의 전술한 다른 필드의 연산 결과를 참조하여 현재 프레임의 각 인터리브 사이클에 대응하는 데이터 연산이 미리 지정된 데이터 예측 조건에 적합하도록 데이터 연산을 예측하여 컴파일된다.

이후 패턴 다중화를 통하여 테스트 패턴 데이터를 생성한다(S140).

이러한 생성된 테스트 패턴 데이터는 이후 테스트될 반도체 소자로 전송되어 테스트를 수행하게 된다.

비록 본 발명의 구성이 구체적으로 설명되었지만 이는 단지 본 발명을 예시하기 위한 것이며, 본 발명의 보호 범위가 이들에 의해 제한되는 것은 아니며, 본 발명의 보호 범위는 청구범위의 기재를 통하여 정하여진다.

발명의 효과

이상 설명한 바와 같이, 본 발명에 따르면 테스트 수행자가 테스트를 위한 소스 코드를 작성시 패턴 생성을 위한 내부 연산을 인터리브 동작 각각에 대해서 구체적으로 분석하여 작성하지 않더라도 데이터 연산을 예측하여 테스트 프로그램이 컴파일되어 인터리브 방식을 통하여 테스트 패턴 데이터의 생성이 가능하다.

도면의 간단한 설명

도 1은 종래 기술에 반도체 소자 테스트 장치의 예시적인 블록도.

도 2는 종래 기술에 따른 패턴 생성 장치의 예시적인 블록도.

도 3은 본 발명에 따른 반도체 테스트를 위한 패턴 생성 장치의 예시적인 블록도.

도 4는 본 발명에 따른 반도체 테스트를 위한 패턴 생성 장치에 있어서 다수의 필드에 대한 동작을 나타내는 도면.

도 5는 본 발명에 따른 반도체 테스트를 위한 패턴 생성 장치에 있어서 테스트 프로그램의 변환 컴파일을 나타내는 도면.

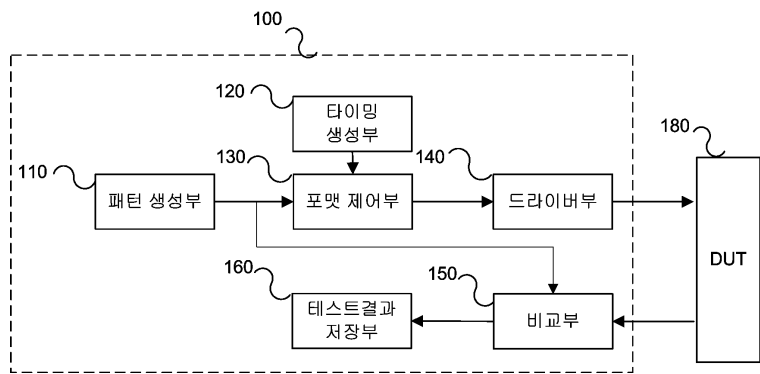
도 6은 본 발명에 따른 반도체 테스트를 위한 패턴 생성 방법의 예시적인 흐름도.

<도면의 주요부분에 대한 부호의 설명>

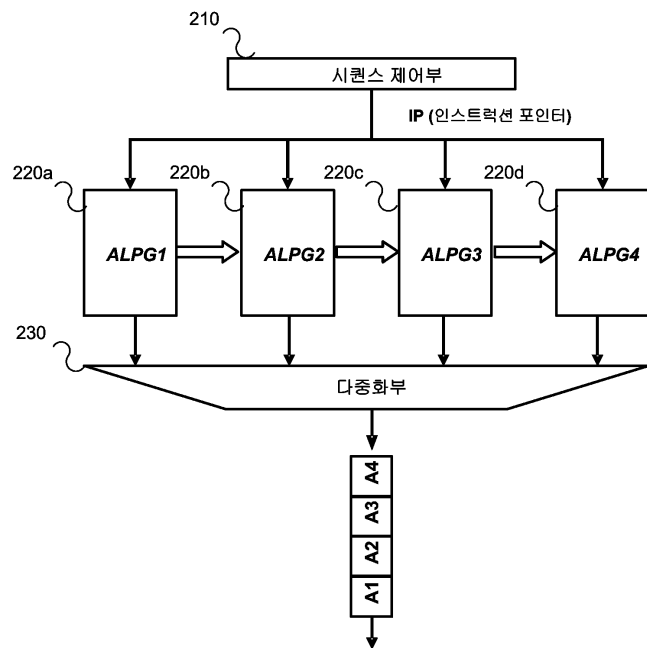
- 110: 패턴 생성부 120: 타이밍 생성부
- 130: 포맷 제어부 140: 드라이버부
- 150: 비교부 160: 테스트 결과 저장부
- 180: DUT 210: 시퀀스 제어부
- 220: 알고리즘 패턴 생성부 230: 다중화부
- 310: 테스트 패턴 프로그램 관독부 320: 알고리즘 패턴 생성부
- 330: 다중화부

도면

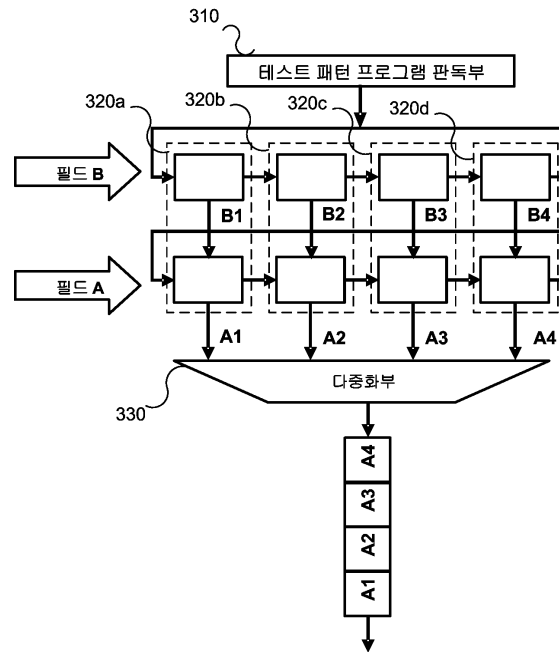
도면1



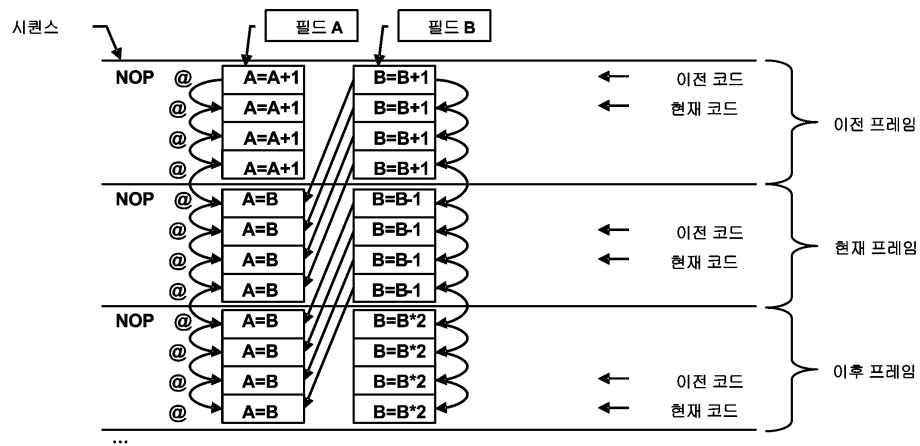
도면2



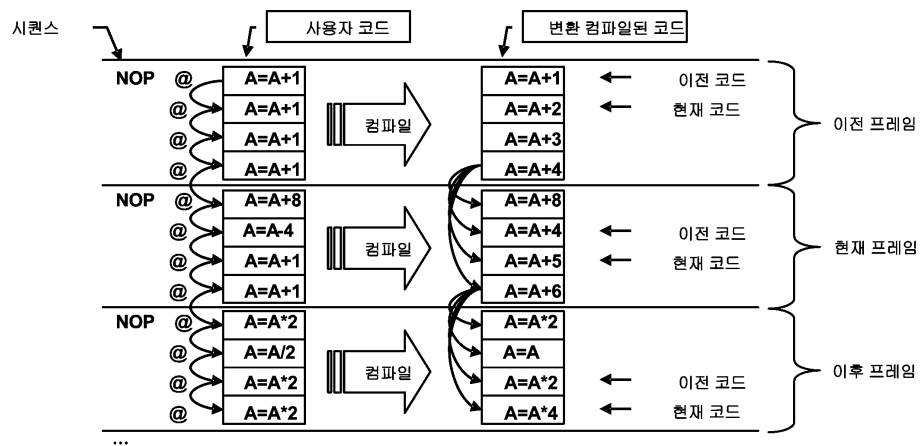
도면3



도면4



도면5



도면6

