경로 지연 고장 테스팅을 위한 부분 확장 주사 방법

김 원 기[†] · 김 명 균[††] · 강 성 호[†††] · 한 건 희[†††]

요 약

반도체 집적 회로가 점점 복잡해지고 고속화되면서 반도체 집적 회로의 동작에 대한 검사 뿐 아니라, 회로가 원하는 시간 내에 동작함을 보장하는 지연 고장 검사의 중요성이 점점 커지고 있다. 본 논문에서는 경로 지연 고장에 대한 효율적인 테스트 입력 생성을 위하여 새로운 부분 확장 주사 방법을 제안한다. 본 논문에서는 유추와 할당을 적용한 테스트 입력 자동 생성기를 기반으로 하여 새로운 부분 확장 주사 방법을 구현하였다. 우선적으로 표준 주사 환경에서 테스트 입력을 생성한 후에 테스트 입력이 제대로 생성되지 않는 주사 사슬에 대하여 테스트 입력 생성기를 수행하는 동안의 정보를 이용하여 확장 주사 플립플롭이 적용될 플립플롭을 결정하였다. 확장 주사 플립플롭을 결정하는 기준으로서는 고장 검출율과 하드웨어 오버헤드를 사용하였다.

순차 회로인 ISCAS 89 벤치 마크 회로를 이용하여 실험을 수행하였으며, 실험을 통하여 표준 주사와 확장 주사 환경, 부분 확장 주사 환경에서 고장 검출율을 비교, 확인하였다. 그리고 새로운 알고리즘이 적용된 부분 확장 주사 방법에서 높은 고장 검출율을 확인함으로써 효율성을 입증하였다.

A Partial Enhanced Scan Method for Path Delay Fault Testing

Won-Gi Kim[†] · Myung-Gyun Kim[††] · Sungho Kang[†††] · Gunhee Han[†††]

ABSTRACT

The more complex and larger semiconductor integrated circuits become, the more important delay test becomes which guarantees that semiconductor integrated circuits operate in time. In this paper, we propose a new partial enhanced scan method that can generate test patterns for path delay faults effectively. We implemented a new partial enhanced scan method based on an automatic test pattern generator(ATPG) which uses implication and justification. First, we generate test patterns in the standard scan environment. And if test patterns are not generated regularly in the scan chain, we determine flip-flops which applied enhanced scan flip-flops using the information derived for running an automatic test pattern generator in the circuit. Determining enhanced scan flip-flops are based on a fault coverage or a hardware overhead.

Through the experiment for ISCAS 89 benchmark sequential circuits, we compared the fault coverage in the standard scan environment and enhanced scan environment, partial enhanced scan environment. And we proved the effectiveness of the new partial enhanced scan method by identifying a high fault coverage.

1. 서 론

오늘날 VLSI 칩 집적도 증가에 따라 칩에 대한 테스트의 중요성이 강조되고 있으며, 우수한 반도체 칩 개발의 필요성은 급속히 증가하고 있어, 반도체 집적 회로의 기능적 특성뿐만 아니라 신호 전달 지연 시간의 검증이 고속 동작을 위한 회로 설계에서 우선적으로 고려해야 하는 사항이 되었다. 그래서 양질의 칩을 얻기 위한 기존의 고착 고장에 대한 테스트 외에도 지연 고장 테스트(delay fault test)가 수행되고 있다. 지

※ 이 논문은 1997년 학술진흥재단의 학술연구조성비에 의하여 지원되었습니다.
† 정 회 원 : 삼성전기
†† 준 회 원 : 연세대학교 대학원 전기전자공학과
††† 정 회 원 : 연세대학교 전기전자공학과 교수
논문접수 : 1999년 12월 17일, 심사완료 : 2000년 9월 20일

연 고장 테스트[1]의 목적은 생산된 회로가 주어진 동작 클럭 레이트(functional clock rate)에서 정상적으로 구동되는지를 확인하는 것이다. 지연 고장 테스트는 신호의 변화가 회로를 통하여 주어진 시간 내에 통과하느냐를 결정하기 때문에 동적 논리(dynamic logic)를 다루는 경우를 제외하고는 적어도 2개의 벡터를 필요로 한다. 따라서 지연 고장 테스트는 설계된 칩이 사용자가 요구하는 동작 주파수의 범위에서 작동하는지를 검사할 수 있다.

앞서 말한 대로, 지연 고장 테스트는 적어도 두 벡터를 필요로 하며, 일반적으로 첫 번째 벡터를 초기값, 두 번째 벡터를 최종값이라고 한다. 지연 고장 테스트를 수행하기 위한 가장 극단적인 방법(brute force method)은 모든 가능한 입력 쌍을 적용시켜 보는 것이다. 한 번의 테스트를 위해 두 입력 벡터를 필요로 하므로, 만약 n개의 입력이 존재한다면 적용해야할 모든 가능한 입력 쌍은 2^{2n}개가 된다. 즉 이 방법을 쓸 경우 입력의 수가 증가함에 따라 적용해야할 입력 쌍의 수가 지수 함수적으로 증가하므로 실시간 내에 테스트하기가 어렵다. 이를 해결하기 위해서는 고장모델을 사용해야 하고 모델링된 고장을 모두 도괄할 수 있는 방법을 개발해야 한다. 이러한 관점에서 볼 때 지연 고장(delay fault)은 회로 내 신호의 진행 지연이 모델링된(또는 허용된) 지연 이상으로 증가하게 하는 제조과정에서 생긴 결함을 모델링한 것으로 정의할 수 있다. 또 지연 고장 테스트는 지연 고장을 검출하거나 위치를 찾아내기 위해 공급되어야 할 입력을 결정하는 과정이다. 회로의 지연을 모델링할 때는 소자의 입력과 출력에 관성지연(inertial delay)이 있다고 가정한다. 그러나 실제 지연모델은 일정 범위 내에서 규정되는 상승지연과 하강지연, 그리고 관성지연으로 모델링 된다. 고착 고장과는 달리 지연 고장은 시스템의 정상 상태에서의 논리동작에는 영향을 미치지 않지만 전체 시스템의 성능을 저하시킨다. 회로내의 모든 경로(path)가 주어진 시간 명세(time specification)보다 신호를 일찍 전달한다면 지연 고장이 없게 된다

순수한 순차회로에서는 조합회로에서 보다 테스트 입력을 생성하기가 매우 까다롭다. 조합회로에서와는 달리 순차회로에서의 출력은 입력뿐 아니라 그 회로의 초기상태에 의해서도 영향을 받게 되고, 한편 순차회로는 각 기억소자의 초기상태를 파악하기가 쉽지 않고, 첫 번째 입력에 의해 상태가 어떻게 변하는지 추적하기가 힘들기 때문에 지연 고장을 테스트하기 위한 입력을 생성하기가 어렵다. 순차회로에서 고착고장을 검사하기 위해 플립플롭을 직렬로 연결하여 각 플립플롭의 초기값을 외부에서 입력할 수 있게 함으로써 조절용이도(controllability)와 관측 용이도(observability)를 높인 주사 플립플롭을 사용하여 효율적으로 테스트 패턴을 생성해 낼 수 있으나, 경로 지연 고장은 테스트를 수행하기 위해 연속적으로 두 패턴을 필요로 하므로 주사 플립플롭의 사용이 테스트 패턴 생성 과정을 크게 향상시키지는 못한다.

이를 해결하기 위한 방법이 확장 주사(enhanced scan)이다. 확장 주사는 원래 회로의 플립플롭에 하나의 플립플롭을 여분으로 더 달아서, 지연 고장 검출에 필요한 2개의 벡터를 연속적으로 입력시켜 줄 수 있게 한 것이다. 그러나 확장 주사를 사용하면 각 플립플롭마다 벡터를 저장하는 플립플롭이 하나씩 더 필요해서 너무 많은 면적을 차지하기 때문에 설계로 많이 사용되지는 않는다. 이를 해결하기 위해 고안된 방법이 전체 플립플롭 중 일부분만 확장 주사 플립플롭으로 바꾸어 주는 부분 확장 주사이다.

본 논문에서는 표준 주사 환경에서 동작하는 경로 지연 고장 검사 입력 생성기를 수정한 후 이를 이용하여 효율적으로 부분 확장 주사를 사용하는 새로운 방법을 제안하였다.

2. 이론적 배경

시스템의 시간에 관련된 결함은 집중된 지연 고장으로 인해 생기거나 경로에 흩어진 분산된 지연 고장들이 합쳐져서 생기게 된다. 지금까지 두 종류의 지연 고장 모델이 개발되어 널리 쓰이고 있는데 이는 게이트 지연 모델(gate delay fault models)[2]과 경로 지연 고장 모델(path delay fault models)[3]이다. 게이트 지연 고장 모델에서는 지연이 회로내의 게이트의 입력이나 출력에 집중된 경우이다. 이는 느린 상승 지연 고장(slow-to-rise delay faults)과 느린 하강 지연 고장(slow-to-fall delay faults)으로 나누어진다. 이 모델의 장점은 고려되어지는 고장 모델의 수가 그리 크지 않으며 고착 고장을 위한 테스트를 이용해서 지연 고장을 위한 테스트를 만들 수 있다는 것이다. 그러나 이 모델은 회로내의 각 게이트에 널리 퍼져있는 작은 지연으로 구성된 결함은 고려하지 못하는 단점이 있다.

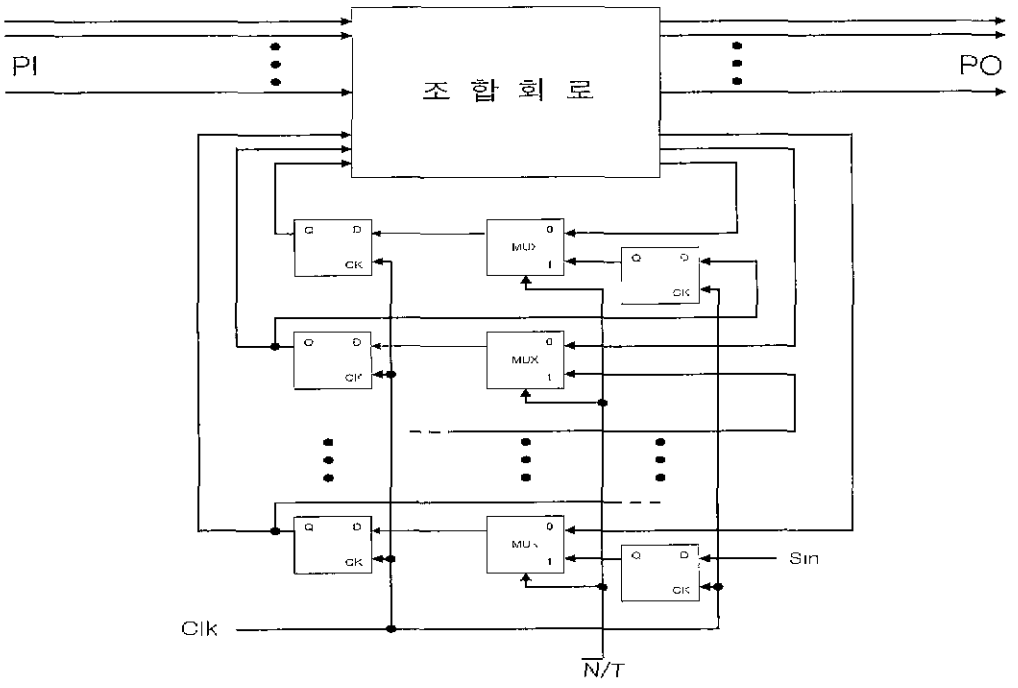
경로 지연 고장 모델은 시험중인 경로를 따라 있는 작은 지연들이 모여서 회로가 오동작을 하게 되는 것을 말한다. 이 모델에 기초를 둔 테스트는 회로의 경로에 따른 집중 또는 분산된 결함을 모두 검출할 수 있다. 이는 회로의 시간 경계 근처에 있는 경로에 해로운 영향을 미치는 변이들을 관측할 수 있게 한다. 또 회로 내의 긴 경로들의 시간 고장에 대한 테스트는 실패가 생길 수 있는 클럭 속도에 대한 정확한 지연값을 제공하므로 속도에 따라 칩을 분류하는데 쓰일 수 있다. 그러나 회로에는 상당히 많은 경로가 있어서 모든 경로를 다 시험하는 것은 거의 불가능하다 왜냐하면 회로가 복잡해짐에 따라 그리고 이에 따른 게이트 수의 증가에 따라 경로의 수는 지수 함수적으로 증가하기 때문이다. 따라서 경로 지연 테스트는 모든 경로 중의 일부분에 초점을 맞추게 되는데 대개 상용 타이밍 분석 장치(timing analyzer)를 사용하여 회로에서 가장 긴 경로를 찾아 테스트 패턴을 생성한다

경로 지연 고장에 대한 테스트에는 크게 무해저드 경성(hazard free robust : HFR)테스트[4], 경성(robust : ROB)테스트, 강연성(strong non robust : SNR)테스트[5], 약연성(weak non robust : WNR)[6]의 네 종류로 나눌 수 있다.

무해저드 경성 테스트는 경로상의 신호들에 동적 해저드(dynamic hazard)가 없고 회로의 다른 부분에서의 지연과 무관하게 경로 상의 지연 고장을 검출할 수 있는 두 패턴의 테스트이다. 무해저드 경성 테스트는 지연 고장 진단에 적합하고 시험중인 경로가 초과의 지연을 가지면 테스트 생성을 실패하게 된다. 경성 테스트는 회로의 다른 부분에서의 지연과 무관하게 경로 상의 지연 고장을 검출할 수 있는 두 패턴의 테스트이다. 경성 테스트는 시험중인 경로가 초과지연을 가지면 실패하며, 또 회로내의 다른 경로가 초과지연 갖는 경우에도 실패할 수 있다. 따라서 회로에 대한 테스트의 목적이 진단이 필요한 경우가 아니고 단지 고장검출 만이라면 경성 테스트를 구하는 것이 무해저드 경성 테스트를 구하는 것보다 더 쉬우므로 유리하다. 경성 테스트의 요건을 갖추지 못한 테스트를 연성 테스트라고 부르며 이는 2가지가 있다. 먼저 경성 테스트를 생성하는 접근법으로부터 나온 연성 테스트를 강연성 테스트라 부른다. 강연성 테스트는 모든 요건이 경성 테스트와 같지만 경성 테스트 중에서 경로의 입력에 정적 해저드(static hazard)가 없어야 하는데 비해 강연성 테스트에서는 정적 해저드가 있어도 무방한 두 패턴의 테스트이다. Schulz[5]는 두 번째 입력의 경우에 경성 테스트와 요구되어지는 값이 같고 첫 번째 입력의 경우 요구되어지는 것이 경로의 시작부분의 초기값으로만 결정되는 방법으로 연성 테스트를 구하였는데 이를 약연성 테스트라고 부른다. 약연성 테스트는 경로 상의 전이가 도달하기 전에 회로의 다른 부분이 최종값으로 안정된다고 가정한 테스트이다. 약연성 테스트를 이용할 경우는 경로를 제외한 회로의 나머지 부분에서는 지연 고장이 없어야 경로 상의 지연 고장을 검출할 수 있다

순차 회로에서 고착 고장을 위한 테스트에서는 테스트를 놓기 위해 주사(scan)를 이용하여 순차 회로를 조합 회로로 바꾼다. 그러나, 지연 고장 테스트에서는 주사를 쓰는 회로에 대해서도 두 개의 사이클을 가진 순차 회로로 여겨진다 이를 기존의 조합 회로에 대한 기법을 사용하려면 2개의 벡터를 동시에 저장할 수 있는 확장 주사(enhanced scan)를 사용하여야 한다. 그러나 확장 주사를 사용하면 각 플립플롭마다 벡터를 저장하는 플립플롭이 하나씩 더 필요해서 너무 많은 면적을 차지하기 때문에 실제로 많이 사용되지는 않는다. 따라서 표준 주사를 이용한 회로에 대해서 연구가 활발해지고 있다[7-9]. 앞에서 언급한 대로 순차회로에서 고착 고장을 검사하기 위해 플립플롭을 직렬로 연결하여 각 플립플롭의 초기값을 외부에서 입력할 수 있게 함으로써 조절 용이도와 관측 용이도를 높인 주사 플립플롭을 사용하여 효율적으로 테스트 입력을 생성해 낼 수 있으나, 경로 지연 고장은 테스트를 수행하기 위해 연속적으로 두 패턴을 필요로 하므로 주사 플립플롭의 사용이 테스트 패턴 생성 과정을 크게 향상시키지는 못한다 이러한 문제를 보완, 해결하기 위한 방법으로 기존의 주사(scan)를 개조하여 원래 회로의 각 플립플롭이 두 시간 프레임동안의 값을 모두 가지고 있도록 하는 확장 주사(enhanced scan)가 고안되었다[10, 11].

만약 확장 주사를 사용하면, 순차 회로에 대해서 조합 회로와 마찬가지로 지연 고장 검출 테스트 입력을 생성할 수 있게 된다. 따라서 표준 주사 환경에서는 테스트 입력 생성을 포기했던 경로들에 대해서 테스트 입력을 생성할 수도 있고, 전체 테스트 입력 생성 시간도 크게 줄일 수 있다. 확장 주사 방법으로 지연 고장 검출을 행하는데 소요되는 시간은 표준 주사를 사용할 때에 비해 약 2배 정도가 걸리게 된다. 이는 주사 사슬의 길이가 길어졌기 때문이다. 대신, 우리는 회로를 조



(그림 1) 부분 확장 주사를 적용한 순차회로

합 회로와 마찬가지로 취급하여 지연 고장 검사를 행할 수 있게 되며, 고장 검출율도 증가하게 된다.

일반적으로 CMOS회로에서 D-type 플립플롭은 트랜지스터 18개를 사용하여 구성되어진다. 이것을 확장 주사의 플립플롭으로 바꾸려면, 하나의 D 플립플롭과 멀티플렉서가 추가되어 플립플롭 부분의 하드웨어가 증가하게 된다. 이러한 하드웨어의 증가는 지연 고장이 용이해지는 것을 고려하더라도 실제 회로에 적용하기에는 실용적이지 못하다. 따라서 회로내의 모든 플립플롭들을 확장 주사의 플립플롭으로 바꾸지 않고 높은 지연 고장 검출율에 도달할 수 있는 부분 확장 주사의 사용이 제안된다. 다음의 (그림 1)은 부분 확장 주사를 적용한 순차회로를 보여주고 있다.

3. 부분 확장 주사 방법

부분 확장 주사 알고리즘은 고착 고장에서 사용하는 부분 주사 알고리즘을 이용하여 생각할 수 있다. 부분 주사는 회로 전체의 플립플롭을 모두 주사 플립플롭으로 바꾸지 않고, 일부만을 바꾸는 방법이다. 이 방법은 회로의 하드웨어 오버헤드를 줄이면서 완전 주사와 비슷한 수준의 고장 검출율을 얻기 위한 방법이다. 우리가 하려고 하는 부분 확장 주사 역시 이와 같은 목표와 방법을 고려하므로, 기존의 부분 주사 알고리즘을 응용하여 사용한다.

부분 확장 주사 알고리즘에는 회로의 구조상 테스트 입력 생성이 어려운 부분을 찾아서 이 부분의 플립플롭을 선택하는 구조적인 방법[12, 13]과 각 플립플롭의 조절용이도와 관측용이도를 계산하여 이를 바탕으로 플립플롭을 선택하는 방법[14, 15]이 있다. 이 때에는 주로 시뮬레이션과 같은 방법을 사용하여 용이도를 계산하기도 한다. 나머지 한가지 방법으로는 테스트 입력 자동 생성기를 이용하는 방법[16, 17]이 있다. 이 방법은 테스트 입력 자동 생성기로 테스트 입력을 생성하여 테스트 입력이 잘 생성되지 않는 고장에 대해서 검출을 쉽게 할 수 있는 플립플롭을 선택하는 방법이다. 테스트 입력 자동 생성기를 이용한 방법은 계산량이 많은 것이 단점이지만 고장 검출율을 높이는 데 매우 효율적인 방법이다. 그러나 이 방법은 사용되는 테스트 입력 자동 생성기의 영향을 많이 받기 때문에, 이러한 방법을 사용하여 부분 확장 주사 설계를 한 회로를 다른 테스트 입력 자동 생성기를 사용하여 테스

트 입력을 생성해 보면 고장 검출율이 낮게 나올 수도 있다.

이러한 부분 확장 주사의 알고리즘은 모두 부분 확장 주사에 적용될 수 있겠지만, 실제로 구현된 것은 많지 않다. 관련 연구로는 기능적 지정(functional justification)과 주사 이동(scan shifting)을 이용한 순차 회로의 지연 고장 검출 테스트 입력 자동 생성기를 기반으로 부분 확장 주사를 구현한 연구가 있다[18]. 이 방법은 우선적으로 기능적 지정과 주사 이동을 이용하여 순차회로에 대한 테스트 입력을 생성하는데, 테스트 입력이 제대로 생성되지 않는 부분에서는 테스트 생성기를 수행하는 동안 얻어진 정보를 이용하여 스캔 체인에서 플립플롭의 우선 순위를 정하게 된다. 그리고 목표한 고장 검출율에 도달할 수 있도록 플립플롭을 선택하여 부분 확장 주사를 구현하게 된다 그러나 이 연구에서는 부분 확장 주사에 사용될 주사 플립플롭을 선택하는데 있어서 BDD(Binary Decision Digram)를 적용하고 있는데 이 방법은 플립플롭을 많이 가지고 있는 회로에서는 BDD 과정을 수행하는데 있어서 많은 메모리를 필요로 하게 된다 따라서 실제로 사용되는 큰 회로에서는 효과적인지 못한 단점이 있다. 본 논문에서는 이런 점을 감안하여 표준 주사 환경과 확장 주사 환경에서의 각 주사 플립플롭이 테스트 입력 생성에 끼치는 기여도를 검사하는 방법으로 부분 확장 주사에 사용될 주사 플립플롭을 선택하게 된다. 테스트 입력 자동 생성기를 기반으로 하여 부분 확장 주사를 구현하였는데, 앞에서도 언급하였듯이 테스트 입력 자동 생성기를 기반으로 부분 확장 주사를 구현하는 것은 사용한 테스트 입력 자동 생성기의 영향을 많이 받게 된다. 그래서 본 논문에서는 가장 일반적으로 사용되는 유추(implication)와 할당(justification)을 적용한 테스트 입력 자동 생성기를 기반으로 하여 부분 확장 주사를 구현한다

확장 주사 환경에서는 테스트 입력이 생성되지만, 표준 주사 환경에서는 테스트 입력이 생성되지 않는 경로가 존재하게 된다. 이 경우 확장 주사 환경에서 테스트 입력 생성을 가능하게 하는 확장 주사 플립플롭들을 표시한다. 이러한 플립플롭들을 확장 주사 플립플롭으로 선택하여 부분 확장 주사를 구현하는 것이다. 확장 주사 환경에서도 테스트 입력이 생성되지 않는 경로에 대해서는 플립플롭을 추가하지 않은 표준 주사 환경으로 남겨 둔다.

확장 주사 플립플롭의 부분적 사용은 다음과 같은 과정을 거쳐 실행된다.

(1) 표준 주사 환경에서 함수적 지정을 사용하여 원하는 경로들에 대한 테스트 입력을 생성한다.
(2) 표준 주사 환경에서 테스트 입력 생성이 실패한 경로들에 대해서 확장 주사 환경을 가정하고 테스트 입력을 생성한다. 성공하면 이때 회로의 상태를 저장한다
(3) 확장 주사 환경에서 테스트 입력 생성이 성공한 경로들에 대해 플립플롭을 중심으로 회로의 상태를 확인하여, 확장 주사의 플립플롭의 공헌도를 계산한다.
(4) (3)에서 얻어진 정보를 분석하여 확장 주사의 플립플롭으로 바꾸어야 할 플립플롭들을 선택하여 우선 순위대로 주어진 하드웨어 오버헤드를 넘지 않는 개수만큼 확장 주사 플립플롭으로 바꾼다.
(5) 표준 주사 환경에서 테스트 입력 생성이 실패한 경로들에 대해서 바뀌어진 회로에서 다시 테스트 입력을 생성한다

위와 같은 방법을 사용하면 비교적 적은 계산량으로 부분 확장 주사가 가능하게 된다. (4)의 과정을 수행할 때에는 미리 하드웨어 오버헤드의 상한선을 정하여 이를 넘지 않는 만큼의 플립플롭을 확장 주사 플립플롭으로 선택한다. 위의 과정에 사용하기 위해서 기존의 주사 환경에서 동작하는 테스트 입력 자동 생성기를 확장 주사 환경과 부분 확장 주사 환경에서도 동작할 수 있도록 개선하였다. 개선된 테스트 입력 자동 생성기는 순차 회로를 입력받은 후 조건에 따라 표준 주사 환경이나 확장 주사 환경을 가정하여 테스트 입력 생성을 수행한다. 앞에서 언급한대로 표준 주사 환경에서 테스트 입력의 생성이 실패한 경로들에 대해서만 확장 주사 환경에서 테스트 입력을 생성한다. 이때, 확장 주사 환경에서도 테스트 입력이 실패한 경로는 확장 주사나 부분 확장 주사 기법을 사용해서는 검사할 수 없는 경로이므로 부분 확장 주사 플립플롭을 선택할 때의 고려 대상에서 제외한다 확장 주사에서 테스트 입력이 성공한 경로가 생기면, 이러한 경로는 표준 주사에서는 검사가 불가능하고 확장 주사에서만 테스트 입력 생성이 가능한 경우이므로, 부분 확장 주사 기법을 사용할 때 테스트 입력 생성의 목표가 된다.

따라서 이러한 경로가 발견되면, 각 확장 주사 플립플롭의 테스트 입력에 대한 기여도를 검사하여 저장하는데, 구체적인 알고리즘은 다음 (그림 2)와 같다.

```
check_flipflops( )
{
    for ( i = 0 ; i < Num_flipflops ; i++)
    {
        if ( flipflop[i].second_value == 0 or 1
            /* 함수적 지정으로 기능한 경우제외*/
            and flipflop[i].second_value != flipflop_input[i].first_
            value )
        {
            if ( flipflop_input[i].first_value == X )
                flipflop[i].edff_used++;   /* 사용된 횟수 기록 */
            else
                flipflop[i].edff_conf--;  /* 입력과의 충돌 기록 */
        }
    }
}
```

(그림 2) 확장 주사 플립플롭의 기여도 검사

확장 주사 환경에서 테스트 입력 생성이 성공하면, 이때의 플립플롭의 값들을 검사하게 된다. 이때 각각의 확장 주사의 플립플롭에서 두 번째 입력값이 X가 아닌 0이나 1의 값을 가졌다면, 이는 주사 사슬을 이용해 값을 할당해 준 것이므로, 확장 주사의 플립플롭이 역할을 한 것이다. 반면에 이 값이 X라면 두 번째 입력값이 테스트에 쓰이지 않은 것이므로, 그 확장 주사 플립플롭에 대해 주사 사슬을 이용해 값을 할당할 필요가 없는 경로임을 알 수 있다. 따라서 두 번째 입력값이 0이나 1인 플립플롭의 경우에는 확장 주사의 플립플롭이 사용되었음을 나타내는 edff_used의 값을 증가시켜 준다. 그리고, 플립플롭 입력단의 두 번째 논리값을 확인하여, 만약 플립플롭 입력의 첫 번째 논리값이 플립플롭 출력의 두 번째 논리값과 다르다면, 이러한 경우에는 함수적 지정을 사용해서는 확장 주사 환경에서 생성된 테스트 입력을 사용할 수 없으므로, 확장 주사 플립플롭의 필요성이 매우 높아진다. 이러한 경우는 edff_conf의 값을 1 만큼 증가시켜서 이러한 성질을 저장한다. 확장 주사 환경에서 테스트 입력 생성이 성공한 모든 경로에 대해 위와 같은 과정을 거치고 나면 각 플립플롭의 edff_used와 edff_conf의 값에는 각각의 플립플롭이 확장 주사 플립플롭으로서 테스트 입력 생성에 기여한 정도가 저장되어 있을 것이다.

위에서 얻어진 정보를 바탕으로 실제 확장 주사 플립플롭으로 바꾸어 줄 플립플롭을 선택할 때는 하드웨어 오버헤드를 기준으로 선택하는 방법과 원하는 고장 검출율을 기준으로 선택하는 방법의 두 가지가 있다. 다음 (그림 3)은 하드웨어 오버헤드를 기준으로 선택하는 알고리즘이다.

```
select_flipflop( )
{
   if ( conf_dff > max_edff )
    /* 입력과 충돌한 플립플롭이 선택할 플립플롭수 보다 클 때 */
      for ( i = 0 ; i < max_edff ; i++ )
      {
          dff_num = get_max_conf_edff( ),
          /* 충돌이 많은 플립플롭부터 선택 */
          change_to_edff( dff_num );
          /* 확장 주사 플립플롭으로 변환 */
          test_paths_per_10%( ),  /* 고장 검출율을 확인 */
      }
   else
   {
       for ( i = 0 ; i < conf_dff ; i++ )
       /* 입력과 충돌한 플립플롭부터 선택 */
       {
           dff_num = get_conf_dff( ),
           change_to_edff( dff_num );
           test_paths_per_10%( ),        /* 고장 검출율을 확인 */
       }
       for ( i = 0 ; i < max_edff-conf_dff ; i++ )
       {
           dff_num = get_max_used_edff( );
           /* 많이 사용된 플립플롭부터 선택 */
           change_to_edff( dff_num );
           test_paths_per_10%( );   /* 고장 검출율을 확인 */
       }
   }
}
```

(그림 3) 하드웨어 오버헤드에 의한 부분 확장 주사 플립플롭의 선택

단순히, 확장 주사 플립플롭이 사용된 경우보다는 입력값과 플립플롭의 값이 충돌하는 경우에 확장 주사 플립플롭의 필요성이 매우 크다고 생각하였다. 이것은 표준 주사 환경에서의 첫 번째 입력은 주사 입력으로 원하는 값을 입력할 수 있으나 두 번째 입력은 함수적 지정에 의해 결정되므로 그 값을 예상하기 힘들다. 따라서 원하는 입력값과 함수적 지정에 의해 생성된 입력값이 충돌하는 경우가 생기게 되는데 이런 경우에 확장 주사 플립플롭을 선택하여 원하는 값을 입력할

수 있게 된다. 따라서 입력값과 충돌한 플립플롭의 경우부터 먼저 확장 주사 플립플롭으로 선택하였다. 선택할 플립플롭의 수가 충돌한 플립플롭의 개수보다 작으면 입력과 충돌한 플립플롭들 중에서 충돌횟수가 큰 것부터 차례로 선택하여 확장 주사 플립플롭으로 바꾸어준다. 만약 입력과의 충돌횟수가 같으면, 사용횟수가 큰 것을 먼저 확장 주사의 플립플롭으로 선택하였다. 선택할 플립플롭의 수가 입력과 출력이 충돌한 플립플롭의 수보다 많으면, 입력과 충돌한 모든 플립플롭들을 확장 주사 플립플롭으로 선택하고, 나머지는 확장 주사의 사용횟수가 많은 플립플롭을 우선적으로 선택한다. 이때, 사용횟수가 같은 플립플롭의 경우에는 입력단이 주입력에서 먼 정도(level)를 기준으로 더 먼 것을 우선적으로 선택하였다.

선택과정 중, 일정 개수마다 고장 검출율을 검사하여, 100%가 넘으면, 자동으로 선택을 마치는 test_paths_per_10%()를 사용하였으며, 이때 고정 검출 테스트 입력에 성공하여 고장 목록에서 제거되는 경로들에 대해서는 그 경로에서 사용된 플립플롭의 기여도를 제거하도록 하여 보다 정확한 정보에 기반 하여 플립플롭을 선택하도록 하였다 또한, 더 이상 확장 주사의 플립플롭으로서 사용된 플립플롭이 없으면, 더 이상 확장 주사 플립플롭의 선택이 필요 없다고 판단하여 선택을 멈춘다.

반면에 고장 검출율을 고려하여 플립플롭을 선택하는 알고리즘은 다음 (그림 4)와 같다 앞의 select_flipflop()과 비슷한 방법으로 선택하지만 이번에는 기준이 하드웨이 오버헤드가 아니라 고장 검출율이라는 점이 다르다. 앞에 방법보다 더 자주 고장 검출율을 확인해야 하므로 일반적으로 시간이 더 오래 걸리는 단점이 있다.

```
select_flipflop2( )
{
    for ( i = 0 ; i < conf_dff ; i++ )
    /* 입력과 충돌한 플립플롭부터 선택 */
    {
        dff_num = get_conf_dff( );
        change_to_edff( dff_num );
        coverage = test_paths_per_10%( );
        /* 고정 검출율을 확인 */
        if (coverage >= target_coverage)
            break;
    }
    if (coverage < target_coverage)
    {
        for ( i = 0 ; i < max_edff-conf_dff ; i++ )
        {
            dff_num = get_max_used_edff( );
            /* 많이 사용된 플립플롭부터 선택 */
            change_to_edff( dff_num );
            coverage = test_paths_per_10%( );
            /* 고장 검출율을 확인 */
            if (coverage >= target_coverage)
                break;
        }
    }
}
```

(그림 4) 고장 검출율에 의한 부분 확장 주사 플립플롭의 선택

4. 결 과

위의 알고리즘을 Ultra Spark 워크스테이션(133*Mhz*)에서 C로 구현하여 실험해 보았다. 부분 확장 주사를 경성 테스트와 연성 테스트로 나누어 실험하였고 목표 고장 검출율은 100%로 설정하였다. 표준 주사 환경과 확장 주사 환경에서의 결과도 첨가하여 부분 확장 주사 환경에서의 결과와 비교할 수 있게 하였다. <표 1>은 연성 테스트에 대한 결과이고 <표 2>는 경성 테스트에 대한 결과이다.

실험에서 고장 검출율 계산 방법을 살펴보면 다음과 같다. 선택한 회로에 대하여 표준 주사를 사용하여 고장을 검출한다. 그리고 표준 주사에서 검출되지 않은 고장에 대하여 확장 주사를 사용하여 고장을 검출한다. 확장 주사에서도 검출되지 않은 고장은 부분 확장 주사에서도 검출되지 않으므로 제외하고 확장 주사에서 검출된 고장을 다시 부분 확장 주사를 사용하여 검출한다. 예를 들어 경성 테스트를 수행한 경우, s400 회로에 대하여 살펴보기로 하자. s400의 총 경로 수는 896이고 표준 주사 환경에서의 고장 검출율은 검출된 고장 수(150)를 전체 경로 수에서 확장 주사 환경에서 검출되지 않은 고장 수를 뺀 것으로 나눈 값으로 계산된다 따라서 표준 주사 고장 검출율은 (150 / (896 - 233)) * 100 = 22.62%가 된다. 그리고 확장 주사 고장 검출율은 당연히 100%로 나오고 부분 확장 주사 고장 검출율은 21개의 플립플롭 중 15개를 선택하면 100%에 이르게 된다. 따라서 확장 주사 환경에서 보다 작

은 하드웨어 오버 헤드로 같은 고장 검출율에 이름을 알 수 있다.

〈표 1〉 부분 확장 주사 플립플롭 선택 알고리즘 실험결과(연성 테스트)

회 로	경로수	전체 플립플롭수	표준주사 고장 검출율	확장주시 고장 검출율	선택한 FF	FF 선택율	부분확장 주시고장 검출율
s208	290	8	46.21	100	8	100.00	100
s298	462	14	64.84	100	6	42.86	100
s344	710	15	81.65	100	3	20.00	100
s382	800	21	55.04	100	12	57.14	100
s400	896	21	55.11	100	12	57.14	100
s420	738	16	47.43	100	16	100.00	100
s444	1070	21	53.63	100	13	61.90	100
s510	738	6	63.01	100	4	66.67	100
s526	820	21	41.67	100	13	61.90	100
s820	984	5	74.29	100	3	60.00	100
s832	1012	5	73.59	100	3	60.00	100
s953	2312	29	66.61	100	5	17.24	100
s1196	6196	18	98.30	100	7	38.89	100
s1238	7118	18	98.32	100	7	38.89	100
s1488	1924	6	82.78	100	3	50.00	100
s1494	1952	6	82.62	100	3	50.00	100
s5378	10000	179	93.74	100	64	35.75	100
s9234	10000	228	71.89	100	22	9.65	100
평 균			67.49	100		51.56	100

5. 결 론

본 논문에서는 지연 고장 테스트 용이화를 위한 설계 기법의 하나인 확장 주사 환경과 부분 확장 주사 환경, 그리고 기존의 표준 주사 환경에서의 지연 고장 테스트가 이떤 차이를 보이는지 실험하였다.

실험 결과를 분석해 본 결과, 표준 주사 환경보다 확장 주사 환경에서의 고장 검출율이 월등히 좋은 것을 확인할 수 있었다. 그러나 하드웨어의 추가가 너무 많이 요구되는 단점이 있었다. 이의 해결을 위해 본 논문에서는 일부분의 플립플롭만을 확장 주사 플립플롭으로 바꾸는 새로운 부분 확장 주사 방법을 제안하였고 이를 실험하였다. 모든 회로에서 표준 주사 환경

〈표 2〉 부분 확장 주사 플립플롭 선택 알고리즘 실험결과(경성 테스트)

회로	경로수	전체 플립플롭수	표준주사 고장 검출율	확장주사 고장 검출율	선택한 FF	FF 선택율	부분확장 주시고장 검출율
s208	290	8	27.50	100	8	100.00	100
s298	462	14	33.82	100	14	100.00	100
s344	710	15	41.11	100	15	100.00	100
s382	800	21	23.10	100	15	71.43	100
s400	896	21	22.62	100	15	71.73	100
s420	738	16	28.18	100	16	100.00	100
s444	1070	21	25.26	100	15	71.40	100
s510	738	6	22.50	100	6	100.00	100
s526	820	21	20.32	100	21	100.00	100
s820	984	5	34.80	100	5	100.00	100
s832	1012	5	34.76	100	5	100.00	100
s953	2312	29	41.43	100	6	20.69	100
s1196	6196	18	90.17	100	14	77.78	100
s1238	7118	18	90.10	100	14	77.78	100
s1488	1924	6	25.06	100	6	100.00	100
s1494	1952	6	25.13	100	6	100.00	100
s5378	10000	179	93.34	100	74	41.34	100
s9234	10000	228	56.50	100	176	77.20	100
평균			40.89	100		83.84	100

에서보다 월등한 100%의 고장 검출율을 확인할 수 있었고, FF 선택율에서 보여지듯이 확장 주사 환경에서보다 하드웨어 오버헤드가 작음을 확인할 수 있었다 이로써 제안한 부분 확장 주사 방법의 효율성을 입증하였다.

참 고 문 헌

[1] V. Iyengar, B. Rosen, and I. Spillinger, "Delay Test Generation Algebra and Algorithms," *Proc. of International Test Conference*, pp.867-876, 1988.

[2] J. Waiwkauski E. Lindbloom, B. Rosen and V. Iyengar, "Transition fault simulation," *IEEE Design and Test of Computer*, pp.32-38, April 1987.

[3] G. L. Smith, "Model for delay faults based upon paths," *Proc. of International Test Conference*, pp. 342-349, 1985

[4] C. Lin and S. Reddy, "On Delay Fault Testing in Logic Circuit," *IEEE Trans. on Computer*, pp.694-703, Step. 1987.

[5] M. Schulz et al., "Advanced Automatic Test Pattern Generation Techniques for Path Delay Faults," *Proc. of Fault Tolerant Computing Symposium*, pp.44-51, 1989.

[6] B. Underwood, S. Kang, and O. Law, "A Path-Delay Test Generator for Large VLSI Circuits," *Proc. of International Conference on VLSI and CAD*, pp. 368-371, 1993

[7] S. Kang, B. Underwood and W. Law, "Path Delay Fault Simulation for a Standard Scan Design Methodology," *Proc. of International Conference on Computer Design*, pp.359-362, 1994.

[8] Bill Underwood, Wai-On Law, Sungho Kang, Haluk Konuk, "Fastpath : A Path-Delay Test Generator for Standard Scan Design," *Proc. of International Test Conference*, pp.154-163, 1994.

[9] Prab Varma, "On Path Delay Testing in a Standard Scan Environments," *Proc. of International Test Conference*, pp.164-173, 1994.

[10] S. Devadas and K. Keutzer, "Design of Integrated Circuits Fully Testable for Delay Faults and Multi-faults," *Proc. of International Test Conference*, pp 284-293, 1990.

[11] S. Devadas and K. Keutzer, "Synthesis and Optimization Procedure for Robustly Delay-Fault Testable Logic Circuits," *Proc. of the 27th Design Automation Conference*, pp.221-227, June 1990

[12] V. Chickermane and J. H. Patel, "An optimization based approach to the partial scan design problem," *Proc. of International Test Conference*, pp.377-386, 1990.

[13] R. Gupta and M. A. Breuer, "Ballast : A methodology for partial scan design," *Proc. of the Fault Tolerant Computing Symposium*, 1989.

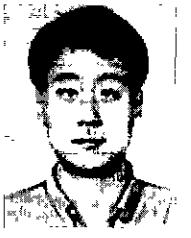
[14] V. Boppana and W. Kent Fuchs, "Partial scan design based on state transition modeling," *Proc. of International Test Conference*, pp.538-547, 1996.

[15] D. Xiang and J. H. Patel, "A Global algorithm for the partial scan design problem using circuit state information," *Proc. of International Test Conference*, pp.548-557, 1996.

[16] V. D. Agrawal, K. T. Cheng, D. D. Johnson, and T. Lin, "Designing circuits with partial scan," *IEEE Design and Test of Computers*, Vol.5, April 1988.

[17] I. Park, D. S. Ha, and G. Sim, "A new method for partial scan design based on propagation and justification requirements of fault," *Proc. of International Test Conference*, pp.413-422, 1995

[18] K. T. Cheng, S. Devadas, and K. Keutzer, "A Partial enhanced-scan approach to Robust delay fault test generation for sequential circuits," *Proc. of International Test Conference*, pp.403-410, 1991.



김 원 기

e-mail : kwongi0@samsung.co.kr

1997년 세대학교 전기공학과 졸업(공학사)

1999년 세대학교 전기공학과 대학원 졸업(공학석사)

1999년~현재 삼성 전기

관심분야 : VLSI CAD, 테스팅, VLSI 설계

김 명 균

e-mail : mgkim@dopey.yonsei.ac.kr

1999년 연세대학교 전기공학과 졸업(공학사)

1999년~현재 연세대학교 전기·전자공학과 학원 석사과정

관심분야 : VLSI CAD, 테스팅, VLSI 설계

강 성 호

e-mail : shkang@yonsei.ac.kr

1986년 서울대학교 제어계측공학과 졸업(공학사)

1988년 The University of Texas Austin 전기·컴퓨터공학과 (공학석사)

1992년 The University of Texas Austin 전기·컴퓨터공학과(공학박사)

1989년~1992년 Schlumberger Inc. Research Scientist

1992년~1992년 The Univ. of Texas at Austin Post Doctoral Fellow

1992년~1994년 Motorola Inc. Senior Staff Engineer

1994년~현재 연세대학교 전기·전자공학과 부교수

관심분야 : VLSI CAD, 테스팅, 설계검증, VLSI 설계

한 건 희

e-mail : gunhee@yonsei.ac.kr

1990년 연세대학교 전자공학과 졸업(공학사)

1992년~1997년 Texas A&M University 전기공학과 (공학박사)

1997년~1998년 Visiting Assistant Professor at Texas A&M University

1998년~현재 Rsearch Scientist at Texas A&M University

1998년~현재 연세대학교 전기·전자공학과 조교수

관심분야 : Analog/Digital Circuit Design Nonlinear Circuits and Systems Artificial Retina Communication VLSI Medical Telemetry