논문 2006-43SD-4-7

효율적인 고장진단을 위한 딕셔너리 구조 개발

(A New Dictionary Mechanism for Efficient Fault Diagnosis)

김 상 욱*, 김 용 준**, 전 성 훈**, 강 성 호**

(Sangwook Kim, Yongjoon Kim, Sunghoon Chun, and Sungho Kang)

요 약

고장 진단은 고장이 빈번히 발생하는 위치를 파악하여 공정상의 문제점을 해결할 수 있도록 하는 매우 유용한 기법이다. 그러나 이 경우 일반적인 고장검출을 위한 것보다 훨씬 많은 고장에 대한 정보가 필요하며, 이는 딕셔너리라고 하는 형태로 저장된다. 이때 집적도가 높은 회로의 경우 고장에 대한 모든 정보를 포함한 딕셔너리를 구성하는 것은 매우 비효율적인 커다란 딕셔너리 크기를 요구하게 되어, 효과적인 딕셔너리 구조가 필요하다. 본 논문에서 제안하는 딕셔너리 구조는 고장에 대한 모든 정보를 포함하면서도 크기가 작은 딕셔너리이며, 이는 단일 고착 고장뿐 아니라 다중 고장의 경우에도 적용이 가능한 효과적인 딕셔너리 구조이다.

Abstract

In this paper, a fault dictionary for fault locations is considered. The foremost problem in fault diagnosis is the size of the data. As circuits are large, the data for fault diagnosis increase to the point where they are impossible to be stored. The increased information makes it impossible to store the dictionary into storage media. In order to generate the dictionary, i.e. pass-fail dictionary some dictionaries store a portion of the information. The deleted data makes it difficult to diagnose fault models except single stuck-at fault. This paper proposes a new dictionary format. A new format makes a dictionary small size without deleting any informations.

Keywords : fault diagnosis, dictionary, static diagnosis

Ⅰ. 서 론

반도체 제조 기술의 발전으로 작은 칩 안에 천 만개 이상의 게이트로 구성된 회로의 구현이 가능하게 되었다. 하나의 칩 내에 구현되는 게이트 개수의 증가에 의해 테스트뿐 아니라 고장진단은 점점 어려워지고 있다. 고장 진단의 경우, 회로의 크기가 점점 커지면서 고려되어야 할 고장의 개수가 증가에 의한 고장 진단에 사용되는 데이터의 증가와 회로 내에 발생하는 고장 원인이 다양해짐에 따라 고장진단의 새로운 방법론 및 자동화 방안이 요구되고 있다[1,2].

로직 고장진단(logic fault diagnosis)은 고장의 원인을 파악하기 위해서 회로의 테스트 결과를 분석하여 고장의 위치를 파악하는 칩의 생산에 필수적인 과정이다. 설계자에 의해 설계된 회로는 생산된 후 칩의 불량 여부를 확인하기 위해 테스트를 수행한다. 테스트에서 통과된 칩은 제 3자에 판매되고 불량으로 판별된 칩은 폐기되어 진다. 따라서 폐기되어 지는 칩의 개수가 적을수록 더 많은 이득을 얻을 수 있을 뿐 아니라 생산 단가가 줄어 제품의 가격을 줄일 수 있으므로 제품의 시장 경쟁력은 향상될 수 있다. 생산된 회로에 대한 정상 칩의 개수를 나타내는 지표인 수율을 증가시키기 위해서 고장을 내포한 칩에 대한 고장 진단이 수행된다. 즉, 고장 진단의 목적은 고장의 원인을 파악하여 수리를 통한 판매와 고장의 재 발생을 방지하여 수율을 증가시킴

* 정회원, LG 전자 SIC 사업팀 SoC 설계그룹.
(LG Electronics., System IC Team, SoC Design Group)
** 학생회원, *** 평생회원, 세대학교 전기전자공학과.
(School of Electrical and Electronic Engineering , Yonsei University)
접수일자: 2004년9월15일, 수정완료일: 2006년3월28일

에 있다. 따라서 고장진단은 수율을 증가시키기 위한 칩의 제조자들에게 필수적인 과정이다[3].

고장진단은 정적 고장진단(static diagnosis)과 동적 고장진단(dynamic diagnosis)의 두 가지 방법으로 분류될 수 있다[5]. 정적 고장진단은 고장 진단에 필요한 데이터를 고장진단 이전에 생성하여 저장한 후, 고장진단 동안에는 저장된 정보를 이용하는 방식이다. 동일 회로에 대한 다양한 고장 회로에 대한 정적 고장진단은 추가적인 정보의 생성이 아닌 저장되어 있는 정보를 이용하여 수행된다. 따라서 정적 고장진단은 고장진단의 수행 시간은 짧은 장점을 가지고 있으나 생성된 정보를 저장하기 위해 많은 저장 공간을 요구하는 단점을 가지고 있다. 동적 고장진단은 고장진단을 수행하는 동안 필요한 정보를 생성하여 고장진단을 수행하는 방식이다. 동일 회로에 대한 다양한 고장 회로에 대한 동적 고장진단은 각각의 고장 회로에 대한 선별된 고장에 대한 정보를 생성한 후 수행된다. 따라서 동적 고장진단은 많은 계산 시간을 요구는 단점을 가지고 있으나 데이터를 저장하기 위한 저장공간을 요구하지 않는 장점이 존재한다.

고장 딕셔너리(fault dictionary)는 정적 고장진단 방식에서 사용하는 정보이다. 즉 고장 딕셔너리는 회로의 고장 리스트 내에 존재하는 고장이 테스트동안 유발하는 에러들을 기록해 놓은 것이다. 반도체의 고집적화로 인해 회로의 크기가 증가되고 이에 따라 고장 리스트 내의 고장 개수가 증가하여 딕셔너리의 크기는 처리할 수 없을 정도로 커지고 있다. 실제 생산되는 칩에 대한 고장 딕셔너리는 현재의 저장매체로는 저장할 수 없을 정도의 크기를 가질 수 있다. 따라서 딕셔너리의 가장 큰 문제점인 딕셔너리의 크기를 줄이기 위한 연구가 진행되고 있다. 딕셔너리의 크기를 줄이기 위해서 딕셔너리를 저장하기 위한 여러 형태들이 제안되고 있다. 크기를 작게 하기 위해서 몇몇 정보를 저장하지 않거나 아니면 저장하는 데이터의 구조를 바꾸는 다양한 방식이 제안되어 왔다[1,2,4,5].

딕셔너리의 다른 문제점으로는 기존의 연구들이 딕셔너리의 크기를 줄이기 위해 단일 고착 고장의 구별 능력이 없는 정보를 제거하는 방식을 사용함에 따라 그로 인해 발생되는 다양한 고장 모델에 대한 고장 진단이 어려워지는 문제를 들 수 있다. 따라서 단일 고작 고장 모델만을 이용하여 생성된 딕셔너리를 이용하여 다양한 고장 모델에 대한 고장 진단을 가능하게 하고자 하는 연구들[6,7]에서 기존의 딕셔너리 방식들은 비효율적인 고장진단 결과를 보여준다.

따라서 본 논문에서는 다양한 고장 모델을 적용한 고장진단에서 필요한 정보의 손실없이 딕셔너리의 크기를 줄이기 위한 효율적인 딕셔너리를 제안한다. 제안하는 딕셔너리 기법은 테스트 동안에 고장에 의해서 유발되는 모든 에러를 기록하면서 크기를 줄일 수 있는 새로운 딕셔너리 저장 방식이다.

Ⅱ. 기존 연구

딕셔너리는 회로의 고장 리스트 내에 존재하는 고장에 의해 유발되는 오류를 기록하는 것이다. 오류라는 것은 테스트 패턴을 회로에 적용했을 경우 고장의 없을 때의 출력과 고장이 있을 경우의 출력에서 서로 다른 값을 갖는 부분이다. 즉 고장을 검출하는 패턴과 그 패턴에 의해 고장이 전파되는 출력을 의미한다.

고장 딕셔너리를 이용한 고장진단 방식의 가장 큰 문제점 중 하나가 바로 딕셔너리의 크기이다. 따라서 딕셔너리의 크기를 줄이기 위한 여러 형태의 딕셔너리가 연구되어 왔다.[1,2,4,5]

딕셔너리에 저장되는 데이터를 줄이기 위해 모든 딕셔너리 저장 방법은 딕셔너리의 구조에 대해 연구되어져 왔다. 딕셔너리의 구조는 딕셔너리의 내용과 순서에 따라 결정된다. 일반적으로 딕셔너리의 구조는 크게 무손실 압축 딕셔너리 (lossless compacted dictionary)와 손실 압축 딕셔너리 (loss compacted dictionary)로 나눌 수 있다. 여기서 손실 압축 딕셔너리의 경우는 고장 딕셔너리에 저장되지 않은 고장이 나타났을 경우 고장진단시 잘못된 정보로 인해 고장진단에 실패할 수 있으며 고장진단의 정확도 (resolution)을 떨어뜨리는 단점이 생긴다. 따라서 딕셔너리의 구조는 고장진단을 위해 필요한 메모리를 줄이기 위해서 가장 중요한 요소라고 할 수 있다. 이러한 이유로 이 장에서는 이전의 다양한 고장 딕셔너리 구조를 살펴보고자 한다.

가장 일반적으로 알려진 딕셔너리는 완전 딕셔너리 (full dictionary)와 pass/fail 딕셔너리다. 완전 딕셔너리는 모든 테스트 패턴에 대해서 모든 고장에 대한 전체 출력값을 저장하는 형식이다. 따라서 고장 시뮬레이션에서 생성된 모든 정보를 기록하기 때문에 많은 저장 공간을 요구하는 단점이 있다. pass/fail 딕셔너리는 출력의 정보를 없애고 패턴에 대한 정보만을 기록한 형태이다. 즉 각각의 고장에 대해서 그 고장을 검출할 수 있는 패턴에 대한 정보를 가지고 있지 않기 때문에 크기

면에서는 가장 작은 장점이 있으나 고장진단의 정확도가 많이 떨어진다는 단점이 있다.

위의 완전 딕셔너리와 pass/fail 딕셔너리의 단점을 보완하기 위해 여러 가지 압축 딕셔너리들이 연구되었다. Pomeranz와 Reddy는 greedy 알고리듬을 이용한 압축 딕셔너리를 제안하였다[4,5]. 이 압축 딕셔너리는 크기를 줄이기 위해 공간(space)과 시간(time) 압축을 하였다. 이 방법으로 고착 고장에 대한 고장 진단의 정확도를 떨어뜨리지 않으면서 딕셔너리의 크기를 줄였으나 완전 딕셔너리에 비해 다양한 고장 모델에 대한 고장 진단의 정확도가 떨어진다는 단점이 있다. 또한 Boppana et al, Chess 와 Larrabee는 트리(tree) 딕셔너리 구조를 제안하였다[2,10]. 이 트리 딕셔너리는 고장이 존재하는 회로에 대한 예상되는 응답값을 트리 구조로 저장하는 방식이다. 트리 구조의 노드는 각각의 테스트 패턴에 대한 고장의 응답으로 구별된다. 트리 구조는 벡터 기반의 트리(vector based tree)와 출력 기반의 트리(output based tree)로 나눌 수 있다. 벡터 기반의 트리는 노드의 수가 적은 장점을 가지고 있으나 트리의 구조가 규칙적이지 않아 트리의 구조를 저장해야 하는 단점이 존재하고 출력 기반의 트리는 트리 구조는 규칙적이나 노드의 개수가 너무 많아 노드를 인코딩하기 위해 많은 비트가 요구되는 단점이 존재한다.

최근에는 Lavo와 Larrabee가 출력 압축 시그너쳐(output compacted signature)를 가진 pass/fail 딕셔너리를 제안하였다. 이 방법은 기존의 pass/fail 딕셔너리에서 구별이 안 되는 고장들을 구별 할 수 있는 정보를 약간 더해줌으로써 고장 진단의 정확도를 높이는 방법이다. 하지만 이 방법은 손실 압축 딕셔너리 방법이기 때문에 손실 압축 딕셔너리가 갖는 고장진단 정확도가 떨어진다는 단점을 여전히 지니고 있다.

Ⅲ. 제안하는 딕셔너리 구조

기존의 딕셔너리 저장 방식은 단일 고착 고장에 대한 고장 구별 능력이 없는 정보를 삭제함으로써 딕셔너리 크기를 줄이는 방식이었다. 이러한 기존의 방식은 고장의 구별 능력이 없는 정보를 찾는 복잡한 과정으로 인해 딕셔너리를 생성하기 위한 계산 시간이 증가하는 문제점을 가지고 있다. 또한 단일 고착 고장만을 고려한 데이터의 삭제는 단일 고착 고장을 제외한 다양한 고장 모델에 대한 고장진단을 어렵게 하고 있다.

그림 1은 하나의 고장에 대한 완전한 정보를 기록한 완전 딕셔너리의 예이다. 각 열은 각각의 출력에서 테스트 벡터에 대한 pass(0)와 fail(1)의 응답값을 나타내고 있다. 제안하는 무손실 압축 방법을 쉽게 설명하기 위해 완전 딕셔너리에 포함된 정보들을 다음과 같이 정의한다.

정의 1: 시간 정보 (Time information)

시간 정보는 고장이 언제 발견되는지에 대한 정보이다. 테스터는 각 테스트 패턴들을 순서대로 가해주기 때문에 어떤 패턴에서 고장이 전파되는지를 쉽게 알 수 있다.

정의 2: 공간 정보 (Space information)

공간 정보는 고장이 어디서 발견되는지에 대한 정보이다. 각 고장은 특정 경로를 통해 출력단으로 전파되기 때문에 공간 정보는 고장이 전파되는 출력단에 대한 정보이다.

완전 딕셔너리에서 각 고장을 위한 고장 검출을 위한 데이터는 위에서 정의한 시간 정보와 공간 정보로 구성된다. 고장 리스트에 N_f의 고장과 N_v의 테스트 벡터와 N_o개의 출력이 있다면 완전 고장 딕셔너리를 저장하기 위한 총 비트수는 $N_f \times N_v \times N_o$이다. 따라서 회로의 복잡도가 증가할 수록 완전 딕셔너리의 크기는 기하급수적으로 증가하게 된다. 물론 정확한 고장 진단을 수행하

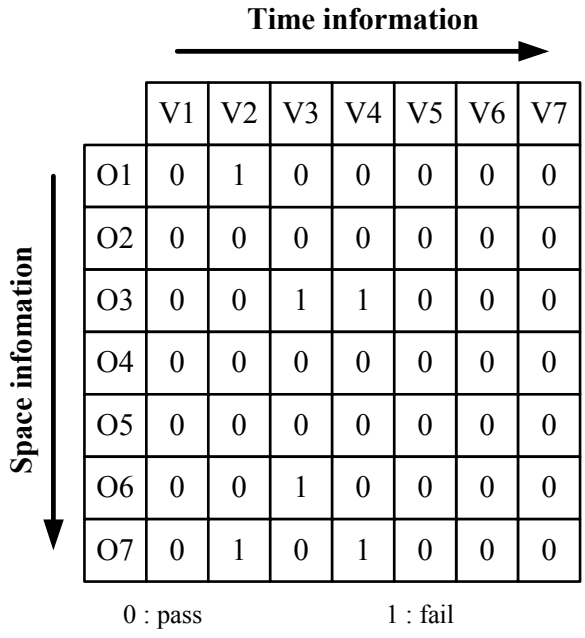

그림 1. 하나의 고장에 대한 완전 딕셔너리
Fig. 1. The full dictionary about a fault.

	V1	V2	V3	V4	V5	V6	V7	V8	V9	OC
O1	0	0	0	0	0	0	0	0	0	0
O2	0	0	0	0	0	0	0	0	0	0
O3	1	1	0	0	0	0	0	0	1	1
O4	1	1	0	0	0	0	1	0	1	1
O5	0	0	0	0	0	0	0	0	1	1
O6	0	0	0	0	0	0	0	0	0	0
O7	0	0	0	0	0	0	0	0	0	0
O8	0	0	0	0	0	0	0	0	0	0
O9	0	1	0	0	0	0	0	0	1	1
PF	1	1	0	0	0	0	1	0	1	

딕셔너리1 딕셔너리2

그림 2. 제안된 딕셔너리의 구조
Fig. 2. Proposed dictionary.

기 위해서 완전 딕셔너리의 고장 정보는 반드시 필요하지만, 이를 위하여 필요한 딕셔너리의 크기가 너무나 크다면 이는 감당할 수 없는 정도에 이를 수 있다. 따라서 고장에 대한 모든 데이터를 가지고 있으면서도 딕셔너리의 크기를 획기적으로 줄일 수 있는 새로운 무손실 압축 딕셔너리의 형식은 반드시 필요하며, 본 장에서는 이를 제안한다.

본 논문에서 제안하는 딕셔너리는 딕셔너리 1과 딕셔너리 2의 두 가지 딕셔너리로 구성되어 있다. 이는 고장증후가 발생되지 않는 패턴의 정보를 기록하지 않음으로써 최소한의 딕셔너리 크기를 통하여 정보를 저장할 수 있다. 우선 딕셔너리 1은 입력패턴과 출력에 대한 간단한 pass-fail 딕셔너리에 의해서 제공된다. 패턴과 출력에 대한 pass-fail 딕셔너리는 패턴을 저장하기 위한 비트의 개수와 출력을 저장하기 위한 비트의 개수의 합으로 2차원 정보를 저장하기 위해서 필요한 정보보다 적은 데이터를 요구한다.

이를 통해서 고장 검출은 충분히 가능하나, 고장 진단을 하기 위해서는 딕셔너리 1만으로는 불가능하다. 따라서 각 고장에 대한 시간 및 공간의 2차원 정보가 필요하며, 이는 딕셔너리 2를 통하여 저장한다. 딕셔너리 2의 2차원 정보는 딕셔너리 1 파일에서 1의 값을 가진 패턴과 출력의 해당 정보만 기록함으로써 만들어진다. 따라서 제안된 딕셔너리는 작은 크기의 딕셔너리를 통하여 고장 진단이 가능한 2차원 정보를 저장하는

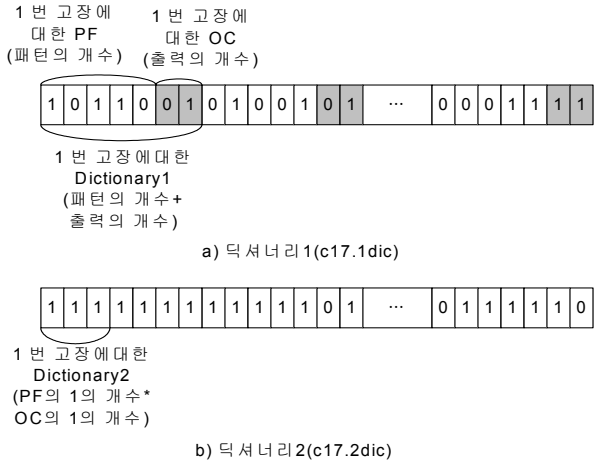

그림 3. 제안된 딕셔너리의 예
Fig. 3. An example of proposed dictionary.

방식이다. 그림 2는 제안하는 딕셔너리 구조의 예이다.

그림 2에서 O1에서 O9는 회로의 출력 핀을 나타낸다. 즉 9개의 출력을 가지는 회로의 경우를 나타내며 V1에서 V9는 각각 테스트 벡터를 나타낸다. 따라서 V1 벡터를 입력으로 가한 경우 이의 출력은 001100000인 것이다. 이러한 방식으로 구성된 전체 딕셔너리를 모두 저장하는 것은 매우 비효율적이므로 딕셔너리 1에서는 이에대한 pass-fail 여부만을 저장하여 필요한 공간을 최대한 줄인다. 그러나 pass-fail 정보는 고장의 존재여부를 확인할 수 있을 뿐 이에대한 진단을 수행하는 것은 불가능하므로, 완전 딕셔너리에서 실제로 필요한 부분만을 저장한다. 즉, 그림 2에서 보는것과 같이 pass-fail 정보중 공통적으로 1을 가지는 부분에 대한 부분만을 딕셔너리 2에 저장하는 것이다. 그림 2의 예는 하나의 고장을 삽입한 예를 나타낸 것으로써 실제로는 그림 2와 같은 정보를 고장의 개수만큼 가져야 하므로 상당한 저장 공간이 필요하게 된다.

그림 3은 C17 bench 회로에 대하여 적용된 제안된 딕셔너리의 예를 보여주고 있다. 딕셔너리 1은 하나의 고장에 대해서는, 패턴의 개수와 출력의 개수를 합한 수만큼의 데이터를 필요로 한다. 딕셔너리 1에서 회로와 회로에 대한 테스트 패턴이 결정된 경우 딕셔너리 1의 사이즈는 고정되며, 이는 고장의 개수×(패턴의 개수+출력의 개수) 와 같다. 딕셔너리 2의 크기는 딕셔너리 1에서 값이 1인 패턴과 출력에 해당하는 2 차원 정보를 기록하기 때문에 고장 시뮬레이션의 결과에 의해서 결정된다.

Ⅳ. 실험결과

제안된 딕셔너리의 성능 평가는 ISCAS 85 벤치마크 마크 회로에 대한 실험을 통하여 이루어졌으며, 실험에 사용된 테스트 패턴은 Synopsys사의 TetraMax ATPG를 이용하여 생성하였다[9,10]. 고장 딕셔너리의 크기를 결정하는 요소인 고장의 개수, 출력의 개수와 테스트 패턴의 개수를 표 1에 나타내었다.

C 언어로 구현된 고장 시뮬레이션을 이용하여 예상되는 고장에 대한 테스트 응답을 제안된 방식으로 저장된 딕셔너리의 크기를 표 2에 나타내었다. 제안된 딕셔너리는 고장에 대한 모든 정보를 가지고 있으므로, 그 크기에 대한 비교는 완전 딕셔너리의 크기와 비교하였다. 표 2를 통해서 제안된 딕셔너리는 회로의 특성에 따라 완전 딕셔너리의 크기 대비 2%에서 23%정도의 크기를 가지고 있음을 확인할 수 있다. 또한, C5315나 C2670과 같이 출력의 개수가 많고 테스트 패턴의 개수가 많은 회로의 경우 제안된 딕셔너리에 의한 압축률이 훨씬 향상된다. 이는 제안된 딕셔너리의 경우 고장이 전파되는 출력의 개수와 고장이 검출되는 테스트 패턴의 개수가 상대적으로 작기 때문이다.

따라서 회로에 대한 테스트 패턴 및 출력의 개수가 증가할수록 제안된 딕셔너리의 크기는 완전 딕셔너리에 비해 훨씬 작아진다는 결론을 얻을 수 있다. 이는 회로의 복잡도 및 크기가 증가하면서 생기는 현상으로 제안된 딕셔너리는 큰 회로에 대해 적용 가능한 형태의 딕셔너리가 될 수 있음을 보여준다.

표 3은 기존의 딕셔너리와의 크기 비교를 나타낸다. 제안된 딕셔너리는 데이터의 삭제를 통한 압축 방식인 공간 압축 방식보다 크기가 작은 것을 볼 수 있다. 공간 압축 방식은 고장간의 구별을 가능하게 하는 정보들만의 기록으로 크기를 줄인 방식이다. 고장간의 구별을 할 수 있는 정보를 알아내기 위해서는 구별할 수 없는 고장의 예상응답을 검사해야 하는 과정이 필요하다. 나무 구조의 딕셔너리는 고장 시뮬레이션에 의해서 생성된 예상되는 고장 응답을 나무 구조로 변경하여 완전 딕셔너리의 모든 정보를 나무 구조로 저장하는 방식이다. 제안된 딕셔너리는 완전 딕셔너리의 모든 정보를 가지고 있는 나무 구조의 딕셔너리보다 모든 회로에 대해서 작은 것을 볼 수 있다. 단일 고착 고장에 대한 고장 구별 능력이 없는 정보의 삭제 방식인 공간 압축 방식에 대해서도 일부 회로를 제외한 제안된 딕셔너리의 크기가 작은 것을 볼 수 있다. 따라서 제안된 딕셔너리는 회로의 크기가 큰 회로에 대해서 좋은 성능을 보이는 것을 볼 수 있다.

표 1. 실험 대상 회로
Table 1. Benchmak circuits.

회로	출력수	고장의 개수	테스트 패턴			
			랜덤	결정	합계	Coverage
C432	7	520	72	6	78	100%
C499	32	750	58	12	70	100%
C880	26	942	95	15	110	100%
C1355	32	1566	81	28	109	100%
C1908	25	1870	86	85	171	100%
C2670	140	2630	229	84	173	100%
C3540	22	3288	214	46	260	100%
C5315	123	5291	188	24	212	100%
C6288	32	7710	37	1	38	100%
C7552	108	7419	193	173	366	100%

표 2. 제안된 딕셔너리의 크기
Table 2. Proposed dictionary size.

회로	완전 딕셔너리 (bit)	제안된 딕셔너리(bit)			제안된 딕셔너리/ 완전 딕셔너리
		제안된 딕셔너리1	제안된 딕셔너리2	전체 크기	
C432	286,104	44,200	20,977	65,177	0.230
C499	1,680,000	76,500	180,175	256,675	0.153
C880	2,694,120	128,112	53,117	181,229	0.067
C1355	5,462,208	220,806	511,525	732,331	0.134
C1908	7,994,250	366,520	434,089	800,609	0.100
C2670	63,698,600	823,190	260,059	1,083,249	0.017
C3540	18,807,360	927,216	708,662	1,635,878	0.087
C5315	137,968,116	1,772,485	1,255,411	3,027,896	0.022
C6288	9,375,360	539,700	465,957	1,005,657	0.107
C7552	293,258,232	3,516,606	3,090,024	6,606,630	0.023

표 3. 기존의 딕셔너리와의 크기 비교
Table 3. Comparison of dictionary sizes.

회로	공간 압축 [5]	나무 구조 [2]	제안된 딕셔너리
C432	0.315	0.77	0.230
C499	0.050	0.21	0.153
C880	0.156	0.31	0.067
C1355	0.039	0.18	0.134
C1908	0.070	0.34	0.100
C2670	0.023	0.11	0.017
C3540	0.181	0.45	0.087
C5315	0.075	0.10	0.022
C6288	0.453	0.29	0.107
C7552	0.065	0.10	0.023
평균 압축률	0.143	0.286	0.094

Ⅴ. 결 론

반도체 산업의 발전으로 고장 진단의 문제점은 점점

증가하고 있다. 회로의 집적도에 따른 고장진단에 사용되는 데이터의 증가와 고장원인의 다양화가 고장진단의 문제점이다.

딕셔너리를 이용한 고장진단은 딕셔너리를 구성하는 많은 데이터에 의해서 고장진단에 많은 제한점을 가지고 있다. 딕셔너리의 저장 가능성에 의해서 고장진단을 수행해야 하는 회로의 크기가 제한될 것이다. 따라서 딕셔너리의 크기를 줄이기 위한 여러 방식이 제안되었다. 기존의 방식은 데이터의 삭제를 통한 딕셔너리의 크기를 줄이는 방식이다. 딕셔너리의 크기를 줄이기 위한 데이터의 삭제는 다양한 고장원인에 대한 고장진단을 어렵게 한다.

본 논문에서는 고장진단을 위한 효율적인 딕셔너리의 구조를 제안하였다. 논문에서 제안하는 딕셔너리는 데이터의 손실 없이 딕셔너리의 크기를 줄일 수 있는 장점을 가지고 있는 것을 실험을 통해 확인하였다. 제안된 딕셔너리는 완전 딕셔너리의 23%에서 2% 정도의 크기로 모든 정보를 기록할 수 있다. 제안된 딕셔너리는 회로의 크기가 켜질수록 딕셔너리의 압축률이 높은 것을 볼 수 있다. 즉 출력의 개수와 패턴의 개수의 증가에 대한 제안된 딕셔너리의 증가율은 높지가 않기 때문에 큰 회로에 대한 적용인 가능하다. 일부 회로에 대해서는 기존의 다른 딕셔너리보다 크기가 클 수 있으나 이것은 제안된 딕셔너리의 모든 정보를 가지고 있기 때문이다. 완전 딕셔너리의 모든 정보를 가지는 장점은 단일 고착 고장을 이외의 고장에 대한 고장진단의 정확도를 높을 수 있기 때문에 다양한 고장진단에 효율성을 높일 수 있다. 또한 딕셔너리를 줄이기 위한 알고리듬이 기존의 방식보다 간단하여 딕셔너리의 생성에 요구되는 계산 시간이 적은 장점을 가지고 있다.

참 고 문 헌

[1] V. Boppana, W. K. Fuchs, "Fault Dictionary Compaction By Output Sequence Removal", Proceedings of International Confernece on Computer-Aided Design, November 6-10, 1994, pp. 576-579.

[2] V. Boppana, I.Hartanto, W. K. Fuchs. "Full Fault Dictionary Storage Based on Labeled Tree Encoding", Proceedings of 14th VLSI test Symposium, May, 1996, pp. 174-179.

[3] J. A. Waicukauski and E. Lindbloom, "Failure Diagnosis of Structured VLSI", In IEEE Design & Test of Computers, August, 1989 pp 49 - 60.

[4] I. Pomeranz and S. M. Reddy, "On the Generation of Small Dictionaries for Fault Location", Proceedings of IEEE/ACM International conference on Computer-Aided Design, November 8-12, 1992, pp.272-279.

[5] I. Pomeranz, S. M. Reddy, "On dictionary-based fault location in digital logic circuits", IEEE Transactions on Computers, January, 1997, pp. 48-59.

[6] S. D. Millman, E. J. McCluskey, and J. M. Acken, "Diagnosing CMOS bridging faults with stuck-at fault dictionaries" Proceedings of International Test Conference, September 10-14, 1999, pp. 860-870.

[7] B. Chess, D. B. Lavo, F. J. Fergeson and T. Larrabee. "Diagnosis of realistic bridging faults with single stuck-at information", Proceedings of IEEE/ACM International Conference on Computer-Aided Design, November 5-9, 1995, pp. 185-192.

[8] B. Chess and T. Larrabeee, "Creating Small Fault Dictionaries", IEEE Transactions on Computer-Aided Design, March, 1999, pp. 346-356.

[9] F. Brglez and H. Fujiwara, “A neutral netlist of 10 combinatorial benchmark circuits and a target translator in FORTRAN,” In Int. Symposium on Circuits and Systems, Special Session on ATPG and Fault Simulation, 1985.

[10] TetraMaxReference Manual. Release 2004. 12, Synopsys Inc., Mountain View, CA, 2004.

저 자 소 개

김 상 욱(정회원)
2001년 8월 연세대학교 공과대학 전기공학과 학사졸업.
2003년 8월 연세대학교 공과대학 전기전자공학과 석사졸업.
현 재 LG 전자 SIC 사업팀 SoC 설계그룹.
<주관심분야 : SoC 설계, SoC 테스트>

전 성 훈(학생회원)
2002년 8월 연세대학교 공과대학 전기공학과 학사 졸업.
2005년 8월연세대학교 공과대학 전기전자공학과 석사졸업.
현 재 연세대학교 전기전자공학과 박사과정.
<주관심분야 : Logic BIST, Functional Test>

김 용 준(학생회원)
2002년 2월 연세대학교 공과대학 전기공학과 학사졸업.
2004년 2월 연세대학교 공과대학 전기전자공학과 석사졸업.
연세대학교 IT 사업단 연구원.
현재 연세대학교 전기전자공학과 박사과정.
<주관심분야 : SoC 설계, SoC 테스트>

강 성 호(평생회원)
1986년 2월 서울대학교 공대 제어계측공학과 학사졸업.
1988년 5월 The University of Texas at Austin 전기 및 컴퓨터 공학과 석사졸업.
1992년 5월 The University of Texas at Austin 전기 및 컴퓨터공학과 박사 졸업
미국 Schlumberger 연구원.
미국 Motorola 선임 연구원.
현재 연세대학교 전기전자공학과 교수.
<주관심분야 : SoC 설계, SoC 테스트>