2000年度 大韓電子工學會 秋季綜合學術大會 論文集 第23卷 第2號 2000/11

카운팅 방법을 사용한 연역적 고장 시뮬레이션의 구현

강신영, 김규철
단국대학교 컴퓨터공학과
전화 : 02-709-2860

Implementation of deductive fault simulation using counting method

Shin-Young Kang , Kyu-Chull Kim

Dept. of Computer Engineering, Dankook University
E-mail : shinyoung72@hanmail.net

Abstract

Fault simulation is often necessary to determine the fault coverage of a given test, that is, to find all the faults detected by test. In this paper we implement a deductive fault simulation using counting method. Counting method uses f_i of fault table and Search list to compute set operation. f_i was counted by fault list of input gate. And we propagate fault list from primary inputs toward primary output by comparing with controling sum. It improved performance by reducing search of faults.

I. 서론

테스팅이란 제품의 결함을 검출하는 작업이다. 결함이 있는 제품이 소비자에게 전달된다면 곧 그 회사의 신뢰성은 떨어지게 되고 소비자에게 외면 당한 회사는 살아남기 힘들게 되므로 테스팅을 철저히 하는 것은 대단히 중요하다.

이렇게 테스팅은 제품의 제조 과정의 한 중요한 부분으로 그 시간과 비용도 많이 들어간다. 오늘날 설계 기술의 발달과 자동화 그리고 제조기술의 향상으로 회로의 집적도가 높아지고 있는 반면 회로의 테스팅은 점점 더 어려워져서 그에 따라 시간과 비용이 증가하고 있다. 따라서 앞으로는 테스팅 기술에 대한 연구도 병행하여 함께 이루어져야 할 것이다.[1]

본 논문은 테스팅 기술 중 고장 시뮬레이션에 대한 연구를 수행하는 것으로서, 기존의 연역적 고장 시뮬레이션(deductive fault simulation)[1,3] 방식에서 사용했던 집합 연산의 문제점을 살펴본 뒤 제어합과 fault table의 계수 f_i를 새로 정의하고 카운팅 방식을 사용하여 어떻게 문제점이 해결 될 수 있는지를 알아보고 이를 구현하고 실험을 통하여 제안된 방법의 효율성을 검증하였다.

II. 고장 시뮬레이션

2.1 고장 시뮬레이션의 개념

고장 시뮬레이션은 그림 1과 같이 무고장 회로(good circuit)와 고장이 주입된 회로(faulty circuit)에 동일한 테스트 패턴을 인가하여 그 회로의 주출력단에서의 반응을 비교하여 반응의 차이가 존재하면 인가된 테스트 패턴으로 가정된 고장이 검출됨을 알아내기 위한 과정이다.[1]

그림 1에서 알 수 있듯이 고장 시뮬레이션의 기본 구

성은 테스트 패턴, 고장 집합(fault set), 고장 검출 알고리즘으로 이루어져 있다.

고장 목록은 고장 모델 중에서 특정 모델을 가정하여 회로 내에서 존재할 수 있는 고장을 위치와 특성별로 기술한 것이다.[1]

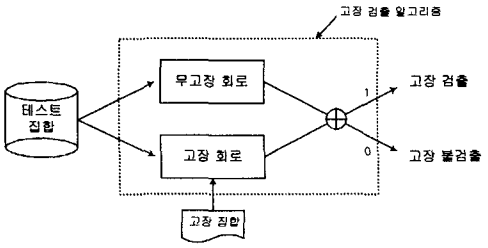

그림 1 고장 시뮬레이션

2.2 연역적 고장 시뮬레이션

연역적 고장 시뮬레이션은 주어진 테스트 패턴에 대하여 무고장 회로 시뮬레이션을 한 후 회로의 각 신호선에 전파되는 고장을 연역적으로 추론하는 방식을 사용한다. 따라서 무고장 회로 시뮬레이션 후, 주입력단에서부터 시작하여 각 신호선까지 전파되는 고장의 집합을 구하여 최종적으로 주출력단에서 검출 가능한 고장의 집합을 구한다.[1]

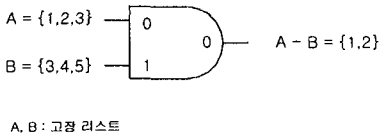

그림 2 고장 리스트 작성 예

그림 2에서 A, B는 고장 리스트를 나타낸다. 그리고 AND 게이트 안의 0, 1, 0은 각각 게이트의 입출력의 정상값을 나타내고 고장 리스트 내의 1, 2, 3 또는 3, 4, 5는 그 정상값을 반대로 만드는 고장을 나타낸다. 따라서 첫 번째 입력의 정상값인 0은 고장 1, 2, 3에 의해서 1이 된다. 그리고 두 번째 입력의 정상값인 1은 고장 3, 4, 5에 의해서 0이 된다.

출력의 정상값인 0도 이것을 1로 만드는 고장을 갖게 되는데 입력의 정상값들로부터 추론해보면 출력이 1이 되기 위해서는 첫 번째 입력은 1로 변해야 하고 두 번째 입력은 변하지 말아야 한다. 따라서 출력으로 전파된 고장은 A-B 연산으로 구할 수 있다. A-B 연산결과 고장 1, 2가 AND 게이트의 출력으로 전파됨을 알 수 있다.

이렇게 연역적 고장 시뮬레이션은 정상값과 그 값을 다르게 만드는 고장들만 유지하면서 고장 리스트에 대한 집합연산을 통해 고장을 전파시킨다.

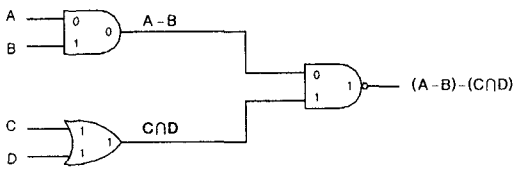

그림 3 고장 리스트의 전파

그림 3의 예로 보면 AND 게이트의 출력을 1로 만들기 위해서는 첫 번째 입력만 1로 되어야 하기 때문에 A집합에서 B집합을 빼야된다. OR 게이트의 출력을 0으로 만들기 위해서는 두 입력이 동시에 0으로 되어야 하기 때문에 C 집합과 D 집합의 교집합을 구해야 된다. 주출력단인 NAND 게이트의 출력을 0으로 만들기 위해서는 첫 번째 입력만 1로 되어야 하기 때문에 (A-B)에서 (C∩D)를 빼주어야 한다. 이러한 방식으로 고장을 최종 출력까지 전파되게 된다.

일반적인 형태로 입력 신호선이 n 개인 AND 게이트에서 논리값 0을 갖는 입력이 m 개일 때에 (n>m) 앞에서의 방법을 확장 적용할 수 있다.

그림 4의 n개의 입력을 갖는 AND 게이트에서 입력 신호 $a_1=a_2=....=a_m=0$, $a_{m+1}=a_{m+2}=....=a_n=1$일 때 출력단의 고장 리스트는 다음 식과 같이 추론된다.[1,2]

$$Fc = ((Fa_1 \cap Fa_2 \cap \cdots \cap Fa_m) - (Fa_{m+1} \cup \cdots \cup Fa_n)) \cup \{c_{s-a-1}\}$$

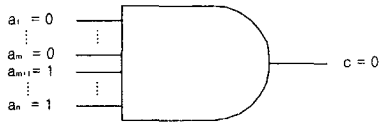

그림 4 고장 리스트 작성의 일반적인 형태

여기에서 Fa_i는 입력 a_i의 고장 리스트를 나타내고 Fc는 출력 c의 고장 리스트를 나타낸다.

하지만 고장 리스트에 대한 집합연산을 한다는 것은 많은 검색시간이 소요된다. A, B 두 집합의 교집합을 구할 때 A 집합의 한 원소에 대해 B 집합의 모든 원소들을 일일이 다 검색하여야 하기 때문에 많은 검색 횟수가 필요하다. 또한 고장 갯수와 그 게이트의 입력 수가 많아질수록 검색시간은 더 기하급수적으로 늘어난다. 하지만 다음 장에서 설명할 카운팅 방식을 사용하면 보다 효율적으로 연역적 고장 시뮬레이션에 필요한 연산을 수행할 수가 있다.

III. 카운팅 방법을 사용한 고장 시뮬레이션

3.1 카운팅 방법을 위한 자료구조

카운팅 방법을 살펴보기 전에 먼저 제어값, 제어합, Fault table, Search list를 정의하자.

첫째, 제어값이란 한 입력단의 값이 다른 입력단의 값에 관계없이 출력을 결정하는 값으로 AND 게이트에서는 한 입력이 0이면 바로 출력이 결정되기 때문에 제어값은 0이고 OR 게이트에서는 1이 된다.

둘째, 제어합이란 한 게이트에서 제어값을 가진 입력의 수를 말한다. 그림 5에서 AND 게이트는 2 개의 입력이 0을 가지고 있으므로 제어합은 2가 된다. 그리고 OR 게이트는 3 개의 입력이 1을 가지고 있으므로 제어합은 3이 된다.

셋째, Fault table은 회로 전체의 고장 목록으로 계수(f_i)를 포함한다. 계수에서는 +,-연산이 이루어지고 한 번 검색된 고장은 변수 Past에 1을 기록한다.

넷째, Search list는 검색된 고장들의 목록으로 고장의 중복 없이 만들어진다. 이때 Fault table의 변수 Past를 조사하여 Past가 0이면 Search list에 추가하고 1이면 이미 삽입되어 있으므로 중복을 피하기 위해 삽입하지 않는다.

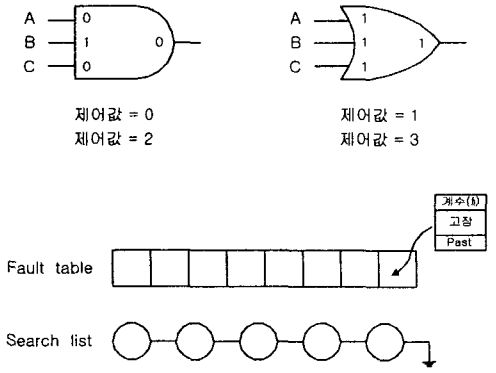

그림 5 제어값과 제어합

3.2 시뮬레이션 방법

카운팅 방법은 Fault table과 Search list를 통하여 연산을 수행한다.

먼저 Fault table의 모든 계수(f_i)는 0으로 초기화된다. 그 다음 입력의 고장 집합에 대한 연산이 수행되는데 입력이 제어값인 고장 집합에 대해서는 Fault table의 계수에 1을 더하고 입력이 제어값이 아닌 고장 집합에 대해서는 Fault table의 계수에 1을 뺀다.

이러한 입력에 대한 모든 연산이 끝나면 Search list를 이용하여 Fault table에서 제어합과 계수의 값이 같은 고장을 찾으면 된다.

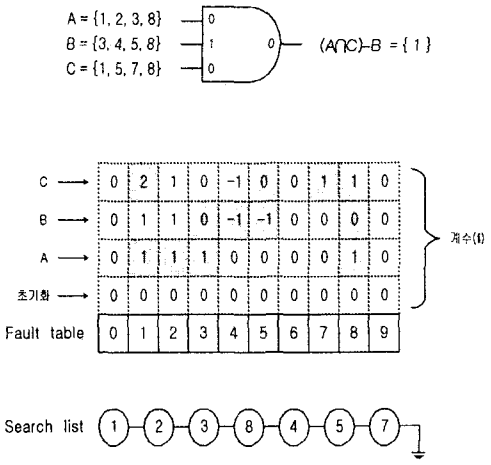

그림 6 고장 시뮬레이션 과정

그림 6은 카운팅 방법에 의한 3입력 AND 게이트의 고장 시뮬레이션을 설명하고 있다.

AND 게이트의 제어값은 0이고 제어합은 2가 된다. 각각의 입력이 0, 1, 0일 때 출력을 1로 만드는 집합을 구하기 위해서는 (A∩C)-B를 구하면 된다.

구하는 방식은 먼저 Fault table의 모든 계수는 0으로 초기화한다. 다음 각 입력의 고장집합에 대해 연산을 수행하는데 첫 번째 입력은 제어값이므로 A 집합의 고장들에 대해서는 그 Fault table의 계수에 1을 더한다. 두 번째 입력은 제어값이 아니므로 B 집합의 고장들에 대해서는 그 Fault table의 계수에서 1을 뺀다. 세 번째 입력은 제어값이므로 C 집합의 고장들에 대해서는 그 Fault table의 계수에 1을 더한다.

이렇게 입력에 대한 연산이 끝나면 Fault table의 계수의 값이 제어합과 같은 고장을 찾으면 된다. 이때 Search list를 이용하면 모든 Fault table를 다 검색할 필요 없이 연산이 수행된 고장만 검색하게 된다. Search list는 입력의 고장들 검색시 만들어진다.

제어합이 0일 경우에는 모든 입력의 합집합을 구하면 출력의 고장 리스트가 얻어진다. exclusive-OR의 경우에는 두 입력이 모두 변하면 출력이 변하지 않기 때문에 입력의 합집합에서 교집합을 빼주면 된다.

Ⅳ. 고장 시뮬레이션의 구현 및 비교

본 장에서는 앞서 설명한 카운팅 방법을 사용한 고장 시뮬레이션을 C++을 이용하여 구현하였다.

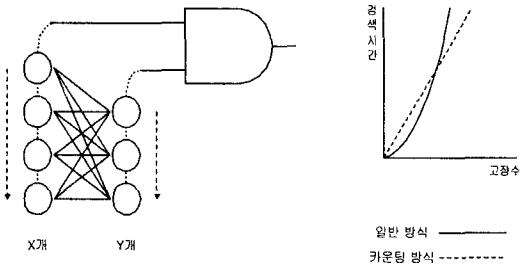

그림 7 카운팅 방식의 처리 및 비교

그림 7은 기존의 시뮬레이션 방식과 카운팅 방식을 비교한 것이다. 입력에 X개와 Y개의 고장이 존재할 때 교집합을 찾기 위해서는 최대 X*Y의 검색 횟수가 필요한데 카운팅 방식을 이용하면 최대 2(X+Y)만으로 구할 수 있으므로 더 빠르다.

여기서 주목할 점은 카운팅 방식은 그래프에서 볼 수 있듯이 고장의 수에 대해 검색 횟수가 선형성을 유지한다는 것이다. 그러므로 많은 고장을 가진 큰 회로에서 더 효율적인 연산을 수행할 수 있다.

표 1은 구현된 고장 시뮬레이션의 속도 측정 결과로 ISCAS85[4] 조합회로에 적용시켰고 입력값은 랜덤벡터를 사용하여 카운팅 방법을 사용하지 않은 일반적인 연역 고장 시뮬레이션과 속도를 비교하였다.

표 1 고장 시뮬레이션 수행속도 비교[초]

회로	패턴수	고장검출률	일반 방식	카운팅 방식
C432	816	96.6%	1.33	0.84
C499	848	96.3%	0.95	0.47
C1908	408	87.2%	2.12	1.48
C3540	408	89.1%	4.39	3.26

Ⅴ. 결론

본 논문은 집합 연산이 이루어지는 연역적 고장 시뮬레이션의 방법과 그 문제점을 살펴보았고 해결방법으로 카운팅 방법을 제안하고 C++를 이용해 구현, 검증하였다.

Fault table의 계수(fi)를 새로이 정의하여 집합 연산시 사용함으로써 그 동안 문제점이었던 중복검색을 해결할 수 있었고 개선된 속도를 얻을 수 있었다. 하지만 다중 논리값 시스템의 적용시 어려운 점이 있어 지속적인 연구가 필요하다.

참고문헌

[1] 홍성제, 박은세, 강성호, 최용호, 장훈 공저, "테스팅 및 테스팅을 고려한 설계", 홍릉과학출판사, pp.1-141, 1998. 3.

[2] Abramovici, Breuer, Friedman, "Digital systems testing and testable design", IEEE Press, 1990.

[3] D. B Armstrong, "A Deductive Method for Simulating Faults in Logic Circuit," IEEE Trans. on Computer, vol. C-21, pp.464-471, May 1972.

[4] F. Brglez, H. Fujiwara,"Neutral Netlist of Ten Combinationl Benchmark Circuits and a Target Translator in FORTRAN" in Secial Session on ATPG and Fault Simulation, Proc. 1985 IEEE International Symposium on Circuits and Systems, June 1985.