韓國事務自動化學會誌
第2卷 第1號. 1995. 6

역추적 결함 시뮬레이션을 이용한 새로운 테스트 생성 알고리즘

A New Test Generation Algorithm Using a Backtrace Fault Simulation

권 기 창*, 백 덕 화**, 권 기 룡***

요 약

결함 시뮬레이션은 테스트 생성의 중요한 과정이며 테스트가 올바른지 검증하거나 결함사전을 작성하는데에 쓰인다.

본 논문에서는 회로의 신뢰성을 검증하기 위해 사용되는 테스트 패턴을 효율적으로 생성하기 위하여 역추적 결함 시뮬레이션 알고리즘을 제안하였다. 제안한 알고리즘의 기본구성은 초기화 과정, 역추적 결함 시뮬레이션 과정 및 입력패턴의 변화가 생겼을 때 즉 리스트변화가 있을때의 재계산 과정 등 3부분으로 되어 있다. 기본 개념은 역추적 과정에서 출력선을 제어하지 못하는 입력을 커팅하므로서 최소의 결함리스트를 유지하는 것이며 리스트의 변화가 생겼을때 논리변화가 일어나는 신호선만 재계산한다. 제안한 알고리즘은 기억장소의 요구도를 O(n)으로 줄이고 수행시간을 향상시켜 효율적임을 보인다.

ABSTRACT

Fault simulation of logic circuits is an important part of the test-generation process. It is used for the propose of generation fault dictionaries or for the verification of the adequacy of tests.

In this paper, a backtrace fault simulation is proposed to test generation. This is consists of 3 part: initialization phase for given circuit, backtrace fault simulation phase to find fault list and reevaluation phase to list event. The main idea of this algorithm is to retain a minimum fault list by cutting uncontrollable lines of path when a logic event occurs in backward tracing phases. And the simulator is reevaluates a fault list associated with the output of an element only if logic event occurs at any of

* 안동전문대학 사무자동화과
** 창원전문대학 전자계산과
*** 창원전문대학 전자통신과

its inputs when a list event occurs at one of its primary inputs. It reguires a O(n) memory space complexity. where n is a number of signal lines for the given circuits. Several examples are given to illustrate the power of this algorithm.

Ⅰ. 서 론

반도체 제조 공정에서나 설계 과정 또는 제작된 회로를 사용하는 사용자에게는 제작된 논리회로가 의도한대로 정상적으로 동작 하는지를 검증하는 테스트 과정이 필요하다.[1]

논리회로에 대한 태스팅은 크게 두 단계로 수행된다. 테스트할 회로를 위한 테스트의 생성단계와 회로에 테스트을 적용하는 테스트 적용단계로 나누어진다. 테스트 셋(test set)의 생성은 회로의 집적도가 높아질수록 매우 어려워지며, 효율적이고 경제적인 테스트를 생성하는 연구가 되어지고 있다.[1,2]

테스트 적용은 대상회로에 테스트 입력을 인가하여 그 응답을 정상적일때의 값과 비교하여 대상회로가 정상인지 고장인지를 판단한다. 테스트 입력이 많을수록 태스트패턴을 인가하는 시간이 길어지며 많은 수의 테스트를 저장하기 위해 많은 기억장소가 요구되며 곧 테스트 비용의 상승을 초래하게 된다. 따라서 테스트 셋을 줄이는 방안이 연구되어지고 있다.[3,4]

테스트 셋을 생성하는 방법에는 테스트 패턴 생성(ATPG:automatic test pattern genera-tor) 알고리즘이나 결함 시뮬레이션을 행하여 구하는 방법이 있으며 보통은 각 방법을 단독으로 사용하며 두 방법을 결합하는 경우도 있다.

일반적으로 테스트 패턴 생성 알고리즘은 결함 시뮬레이션 알고리즘보다 시간이 많이 걸린다. 따라서 테스트 패턴 생성 알고리즘과 결함 시뮬레이션이 결합할 때에는 가능한한 테스트 패턴 생성 과정을 적게 사용하고 결함 포함율을 높일 수 있는 방법이 좋다.

결함 시뮬레이션은 테스트 셋을 구하거나 테스트 셋을 줄이는 것에 이용되며 그 외에도 다음의 목적으로 사용된다. 결함 시뮬레이션은 테스트 패턴의 타당성을 검증하고, ATPG에서 생성한 테스트 셋에 대한 결함 포함율을 계산하기 위해 사용된다. 또한 이 시뮬레이션은 결함진단의 능력을 향상시키기 위해 혹은 결함사전(fault dictionary)을 작성하여 테스트 엔지니어가 회로의 결함을 찾는데 응용이 될 수 있다.[2]

결함 시뮬레이션으로 테스트를 생성하는 과정은 난수 입력 패턴으로 주어진 회로에 적용하여 검출할 수 있는 결함을 모두 찾아내어 결함 리스트를 생성하며 회로내의 모든 신호선에 대한 결함을 모두 검출하거나 충분한 결함포함율이 될때까지 이 과정을 반복한다.[2-4]

본 논문에서는 회로의 신뢰성을 검증하기 위해 사용되는 테스트 패턴을 효율적으로 생성하기 위하여 역추적 결함 시뮬레이션을 이용한 새로운 테스트 생성 알고리즘을 제안한다. 역추적 과정에서 회로의 출력선을 제어하지 못하는 입력을 커팅하므로서 필요없는 계산을 하지 않으며 리스트의 변화가 생겼을때 논리변화가 일어나는 신호선만 재계산 한다. 제안한 알고리즘은 기억장소의 요구도를 줄이고 수행시간을 향상시켜 효율적임을 보인다.

Ⅱ. 테스트 생성

2.1 테스트 생성 과정

논리회로의 테스팅은 주어진 회로에 어떤 입력을 인가하여 그 출력이 정상일때와 비정상일때

를 관찰함으로써 이루어진다. 이때 회로에 사용되는 입력패턴을 테스트 패턴이라고 하며, 일반적으로 하나의 논리회로에 대한 테스트는 많은 테스트 패턴들로서 구성되어진다. 그 테스트 패턴들은 테스트 셋 혹은 테스트 시이퀀스(test sequence)라고 불리어진다. 테스트 시이퀀스는 테스트 패턴의 연속을 의미하는 것으로서 테스트 패턴이 특별한 순서로 적용되어질 때에 사용한다. 테스트 패턴은 그에 대한 출력응답과 함께 테스트 데이타라 한다.[2]

테스트의 생성 과정은 주어진 회로의 모든 결함을 검출할 수 있는 최소 집합의 테스트 패턴을 생성하는 것이 목표이다. 이의 생성과정은 결함 리스트에서 하나의 결함을 선택하여 테스트 패턴 생성 알고리즘을 가동하여 생성한 것이다. 구해진 테스트 패턴은 결함 시뮬레이션을 행하여 찾아진 결함을 결함 리스트에서 제거하고, 이 과정을 충분한 결함 포함율이 될때까지 반복하여 테스트 데이타와 함께 결함사전을 작성한다.[2,3]

테스트 패턴의 생성 알고리즘으로는 경로 감지식인 D-알고리즘, PODEM (path-oriented decision making) 알고리즘 및 FAN(fanout-oriented algorithm) 알고리즘이 있으며, 비교적 FAN 알고리즘이 효율이 좋은 것으로 평가되고 있다.[2]

논리회로의 결함(fault)에는 논리소자의 물리적인 요인에 의해 논리기능이 바뀌는 논리결함과 전파속도, 전압 및 전류 등 파라미터 크기의 변화로 인한 파라미터 결함이 있지만 본 연구에서는 논리결함만을 취급한다. 논리결함에는 회로내에 있는 신호값이 논리값 0 또는 1로 고정되는 결함으로서 0으로 고정될때 0-고착결함이라 하고 s-a-0(s/0,D)로 표현하며, 1로 고정될때 1-고착결함이라 하고 s-a-1(s/1,~D)로 표현한다. 논리결함중 단일고착결함모델(single stuck at fault model)은 회로내의 모든 신호선 중에서 한번에 한개만 0 또는 1로 고착한다고 가정하는 것으로서, N개의 신호선으로 이루어진 회로의 경우 전부 2N개의 결함으로 이루어진다.[2,3] 그러므로 테스트 생성 및 결함 시뮬레이션 알고리즘은 효율적인 테스트 생성을 위해 각 과정 모두가 계산 효율이 높게 개선 되어야 한다.

2.2 결함 시뮬레이션

결함 시뮬레이션은 주어진 회로에 임의의 테스트 패턴으로 구할 수 있는 모든 결함을 출력해주는 방법이다. 결함 시뮬레이션을 행하면 입력 테스트 패턴에 대한 주변 검출현상이 생겨서 생성된 테스트 패턴의 축소가 일어난다. 결함 시뮬레이션 알고리즘에는 PFS(parallel fault simulation) 알고리즘, DFS(deductive fault simulation) 알고리즘 및 CFS(concurrent fault simulation) 알고리즘이 있으며, CFS가 일반적으로 우수한 것으로 알려지고 있다.[5-9]

일반적으로 시뮬레이션의 입력패턴으로는 초단입력(PI:primary input)에 대한 이진난수(binary random number)를 사용한다. 회로의 입력선의 수가 n이라면 가능한 입력의 경우 수가 $2n$이 되어서 시간복잡도가 지수시간(exponential time)이 된다. 결국 결함 시뮬레이션으로 테스트 셋을 구하는 경우에는 NP-complete 문제가 되고 구해진 테스트 셋도 최적 테스트 셋이라 할 수 없다. 그러므로 가능한 한 많은 결함을 검증할 수 있는 초단입력을 선택할 수 있어야 하고, 결함 포함율을 적정선으로 유지하는 최적 테스트 셋를 찾는 것이 필요하다.[2,3]

기존의 결함 시뮬레이션을 하는 방법으로는 주어진 입력패턴으로 그 회로의 외부출력에까지 결함의 영향이 전파되는 모든 신호선을 구하는 것으로서 각 알고리즘의 연산방식에 따라 시간소비와 기억장소의 요구도 등 효율이 다르다.[9-13] 그러므로 기존의 방법은 수행전에 요구되는 기억장소의 크기를 알 수 없을 뿐만 아니라 많은 기억

장소를 요구한다. 따라서 수행전에 요구되는 기억장소를 예상할 수 있거나 작은 기억장소가 요구되는 알고리즘이 개발되어야 한다.

Ⅲ. 역추적 결함 시뮬레이션의 제안

본 논문에서는 기억장소의 요구도를 최대한 줄이고 수행시간을 향상시킨 역추적 결함 시뮬레이션 알고리즘을 제안하였다. 제안한 알고리즘의 과정은 초기화 과정, 역추적 결함 시뮬레이션 과정 및 테스트 패턴의 변화인 리스트변화(list event)가 생겼을 때의 재계산 과정등 3부분으로 구성되어 있다.

초기화 과정은 주어진 초단입력(PI)값에 대해 전방추적(forward tracing)을 하면서 모든 신호선에 전파되는 값을 세트한다. 역추적 결함 시뮬레이션 과정은 초기화 과정에서 세트된 값을 이용하여 종단출력(PO:primary output)으로부터 역추적(backward tracing)하면서 정상적인 값을 반대로 만드는 결함들을 검출한다. 재계산과정은 연속적인 결함 시뮬레이션을 행하기 위해 PI 값이 바뀌었을때, 즉 리스트변화가 생겼을때 재계산하는 과정으로서 기존의 CFS와는 달리 변경된 PI값으로 회로를 따라 논리변화(logcic event)가 일어나지 않을 때까지 전방추적하며, 이때 변화된 결함리스트는 새로운 결함리스트가 된다.

3.1 초기화 과정

초기화 과정은 주어진 대상회로에 대해 임의의 입력패턴을 선택하여 PI에서 PO로 전방추적하면서 각 신호선이 정상적일때 전파되는 논리값을 세트하는 과정이다. 초기화 과정에 대한 단계는 다음과 같다.

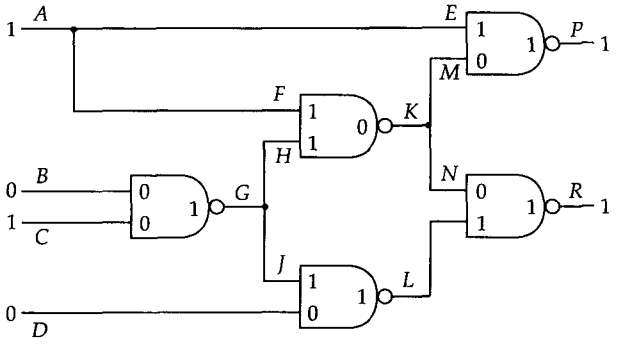
Initialize Net_List : (초기화 과정)

Step 1. 테스트 패턴 값은 flag = 0이면 대상 회로에 대한 난수입력패턴 (random input pattern)을 PI 값으로 선택하며, flag = 1이면 사용자가 주는 입력패턴을 PI 값으로 선택한다.

Step 2. PI에서 PO로 전방추적 하면서 넓이우선탐색(breadth-first)으로 각 신호선이 정상적일때 전파되는 값을 세트한다.

예제 회로에 대한 초기화 과정을 〈그림 1〉에 나타 내었다.

〈그림 1〉에 나타낸 바와 같이 초단입력이 ABCD = 1000 일때 각 입력으로 부터 전파되는 신호값으로 NAND게이트 G = 1로 결정되고 이후 각 게이트의 출력단은 K = 0, L = 1, P = 1 및 R = 1로 논리값이 결정되어 종단출력은 PR



〈그림 1〉 예제회로에 대한 초기화 과정

= 11로 결정되어진다. 각 게이트안의 숫자는 정상적일때 전파되는 논리 값이다.

3.2 역추적 결함 시뮬레이션 알고리즘

역추적 결함 시뮬레이션 과정은 초기화 과정에서 세트된 값을 이용하여 PO로부터 PI쪽으로 넓이우선탐색(breadth-first)으로 역추적하면서 각 게이트들의 출력이 제어하지 못하는 입력(uncontrollable input)을 커팅(cutting)하는 것이다.

역추적 결함 시뮬레이션 과정에 대한 단계는 다음과 같다.

Backtrace Fault Simulation(BFS) : (역추적 결함 시뮬레이션 과정)

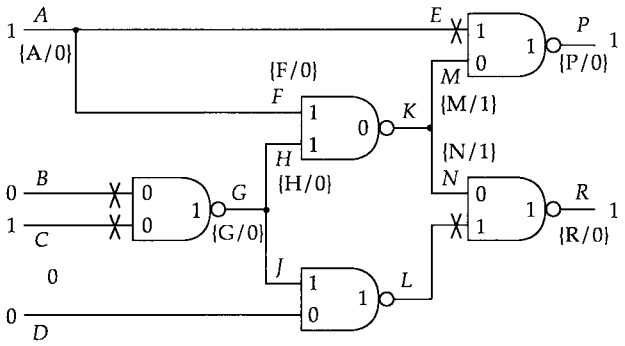
Step 1. 검출가능한 결함은 PO에 세트된 값의 반대 값을 결함으로 선택하고, PO로 출력으로 하는 게이트를 선택한다.

Step 2. 선택된 게이트의 입력들에 대해서 Step 3에서부터 Step 5를 반복하면서 PO에서 PI 쪽으로 역추적한다.

Step 3. 만일 선택된 입력값이 변할 때 그 게이트의 출력값이 변화하지 않으면(비제어성 입력이면 그 신호선으로는 더 이상 역추적을 하지 않고 커팅한다. 출력값을 변화시키면(제어성 입력이면 그 신호선을 선택한 후 다음 단계로 간다.

Step 4. 선택된 신호선에 세트된 값의 반대값을 검출 가능한 결함으로 택한다.

Step 5. 그 신호선을 출력으로 하는 게이트가 없으면 역추적을 멈추고, 게이트가 존재하면 Step 2에서부터 Step 5를 다시 반복한다.



〈그림 2〉 예제회로에 대한 역추적 결함 시뮬레이션 과정

예제회로에 대한 역추적 결함 시뮬레이션 과정은 〈그림 2〉에 나타내었다. 여기에서 중괄호안의 표시는 구해지는 결함을 의미하며 알고리즘의 각 단계별로의 설명은 다음과 같다. 단계 1에서 종단 출력 P=1 및 R=1의 반대 신호값은 검출가능한 결함으로서 결함리스트에 각각 P/0 및 R/0으로 추가 되어진다. 단계 3 및 4의 과정으로 출력이 P인 게이트의 입력선 E와 M에서 E=0인 값은 출력 P의 정상 값을 변화하지 못하는 비제어성 입력이므로 커트되어지고 M=1로 변하면 출력 P가 0으로 변하는 제어성 입력이므로 결함 M/1은 검출가능한 결함으로 결함리스트에 포함된다. 한편 단계 2에서 단계 5의 순환과정으로 R의 출력을 변화시키는 결함 N/1은 구해지지만 R값을 제어하지 못하는 신호선 L은 커트되어져 하위의 경로로 탐색이 봉쇄된다. M과 N은 분기점 K의

변화에 따라 제어되므로 결함 K/1도 구해지며 이후 같은 방법으로 결함 F/0, A/0, H/0 및 G/0이 구해진다. 더 이상의 논리변화가 일어나는 신호선이 없으지므로 역추적 과정을 중지하고 결함리스트를 출력한다. 이상의 과정으로 초단입력 패턴을 ABCD=1000으로 하여 검출할 수 있는 결함의 리스트는 A/0, F/0, G/0, H/0, K/1, M/1, N/1, P/0 및 R/0이다.

초기화과정과 역추적 결함 시뮬레이션 과정만으로도 주어진 회로에 대한 입력 패턴으로 검출할 수 있는 결함을 모두 리스트하는 완전한 알고리즘이 되지만 연속적인 테스트 입력의 변화에 대해서는 실행하기 곤란하다.

하나의 입력패턴으로 구해지는 결함의 종류가 f개 이고 회로내의 신호선의 수가 n이라면, 한번의 역추적 과정에 대한 계산 복잡도는 O(f)가 되며 $f \leq n$이므로 다른 알고리즘에 비해 결함시뮬레이션의 속도가 빠르다.

3.3 리스트변화가 생겼을 때 재계산 과정

역추적 결함 시뮬레이션 과정은 주어진 입력패턴에 대한 검출 가능한 결함을 모두 리스트하지만 테스트 패턴의 연속적인 변화에 대해서 결함시뮬레이션과정을 연속적으로 실행하기 위한 과정이 필요하다. 이 과정에서 변화된 입력 패턴은 새로운 검출가능한 결함리스트를 생성하며 이를 재계산과정 또는 재결함 시뮬레이션이라 한다. 본 논문에서는 최소한의 계산으로 리스트변화에 대한 재계산과정을 행한다.

리스트변화(list event)는 입력패턴의 값이 변하는 것이다. 리스트변화가 생겼을 때 재계산과정은 해당 PI에서의 모든 신호선에 대해서 논리변화가 일어나지 않을 때(즉, 세트된 값과 동일한 값을 갖게 되는 신호선에 도달할 때)까지 과정을 반복해서 새로운 검출가능한 결함을 리스트한다. 리스트변화가 생겼을 때의 재계산과정에 대한 단계는 다음과 같다.

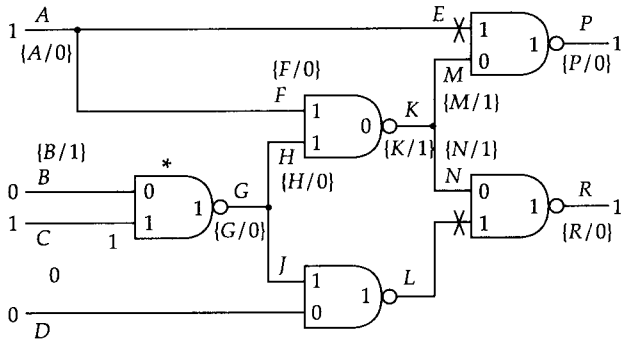
Reevaluation :
(리스트변화가 생겼을 때 재계산 과정)

Step 1 PI에서 PO로 입력패턴으로 부터 각 신호선(line)에 논리값을 정하면서 논리변화가 없을 때까지 다음 과정을 반복한다.

Step 2 PI에서 PO로 전방 추적을 하면서 만나는 게이트의 출력을 제어하는 제어성 입력에 대해 정상일때 전파되는 신호의 반대값을 결함 리스트에 넣는다.

Step 3 앞 과정에서 커트(cut)된 신호선을 만나면 이후의 결함을 별도 관리 하면서 계속 전방추적한다.

Step 4 도달 신호선이 PO이거나 앞 과정에



〈그림 3〉 예제회로에 대한 리스트변화가 생겼을 때의 재계산과정

서 형성된 값과 동일한 신호선을 만나면(논리변화가 없는 경우 전방추적을 끝낸다. 이때 만일 도달 신호선이 앞과정에서 커트된 신호선이면 Step 3에서 별도 관리한 결함 리스트를 무시한다.

예제회로에 대한 리스트변화가 생겼을때의 재계산과정은 〈그림 3〉에 나타내었다. 여기서 게이트의 *표는 그 입출력단에 논리변화가 일어나서 재계산과정이 수행됨을 표시한다.

〈그림 2〉의 역추적 결함 시뮬레이션 과정에서 입력패턴이 ABCD = 1000으로 구해진 결함리스트를 이용하여 〈그림 3〉의 입력패턴이 ABCD = 1010으로 변화되었을때 즉 리스트의 변화가 일어났을때의 계산과정은 다음과 같다. 단계 1에서 PI에서 PO 쪽으로 진행하면서, 변화된 입력값으로 각 신호선의 정상적인 논리신호값을 전파하게 된다. 게이트 G에서 더 이상 논리변화가 일어나지 않으므로 단계 2로 간다. 단계 2에서 신호선 B와 C에 대해 재계산한다. 결함 B/1만 게이트 G의 출력을 제어하므로 결함리스트에 추가되어진다. 이후 논리변화가 없으므로 단계 4로 가서 재계산 과정을 끝낸다. 최종으로 얻어진 〈그림 3〉의 결함리스트는 A/0, B/1, F/0, G/0, H/0, K/1, M/1, N/1, P/0 및 R/0이며 〈그림 2〉의 예에서 B/1만 추가되어진 결과이다. 따라서 제안한 알고리즘은 재계산과정에서 논리변화가 일어나는 부분만 계산하므로 시간 절약을 할 수 있다. 이것은 대규모 집적회로에 대해서 결함 시뮬레이션을 행할때 상당한 장점이 됨을 명백히 알 수 있다.

3.4 실험 결과 및 고찰

제안된 알고리즘의 효율성을 입증하기 위해 다수의 회로를 대상으로 실험하여 기존의 CFS와 비교하여 〈표 1〉에 나타내었다.

여기서 테스트 패턴은 난수 입력 패턴을 발생

〈표 1〉 실험 결과 비교

〈table 1〉 The experimental results

실험 회로				CFS		BFS	
NO	PI수	LN수	gate수	SR	C	SR	CC
C-1	4	14	5	28	5	9	2
C-2	5	15	6	32	6	14	3
C-3	6	19	7	45	7	11	4
C-4	7	24	9	55	9	12	5
C-5	7	49	11	89	11	27	5
C-6	8	32	15	75	15	15	6
C-7	11	30	18	60	18	14	12
C-8	12	45	18	101	18	29	11
C-9	14	50	22	170	22	30	10

(여기서, SR : 기억장소 요구도; CC : 계산 회수)

시켜 적용하였으며 각 대상회로에서 두 방법 모두 같은 결함리스트를 얻게 되었다.

계산 회수를 비교해 보면 기존의 CFS는 주어진 모든 게이트에 대해 결함 검출 가능성을 점검해야 하지만, 제안 알고리즘에서는 커팅되지 아니한 게이트에 대한 점검만 하면 되므로 훨씬 적어짐을 명백히 알 수 있다. 특히 기억장소 요구도는 검출 가능한 결함만이 기억 유지되므로 최

소의 기억 장소가 사용된다. 즉, 기억장소의 요구도는 최악의 경우 신호선의 수가 n일때 (2n)이다. 이것은 정상적인 값의 세트하기 위한 장소 O(n)과 각 신호선의 검출 가능한 결함을 보관하기 위한 장소 O(n)의 기억장소 요구도를 갖는다. 따라서 기억장소 요구도는 O(n)이 된다. 계산 회수는 하나의 테스트 패턴에 대해 논리변화가 일어나는 신호선에 따라 선형적으로 비례한다. 논리변화는 커트되는 신호선이 많을수록 CFS 보다 훨씬 줄어들게 됨을 알 수 있다.

Ⅳ. 결 론

본 논문에서는 회로의 신뢰성을 검증하기 위해 사용되는 테스트 패턴을 효율적으로 생성하기 위하여 역추적 결함 시뮬레이션을 이용한 새로운 테스트 생성 알고리즘을 제안하였다. 제안된 알고리즘은 주어진 회로에 대한 결함을 빠른 속도로 모두 검출한다. 제안한 알고리즘의 기본 구성은 초기화 과정, 역추적 결함 시뮬레이션 과정 및 리스트변화가 생겼을 때의 재계산 과정의 3부분으로 되어 있다. 역추적 결함시뮬레이션의 기본 개념은 회로의 출력단을 변화시키지 못하는 비제어성 입력을 커트하므로서 기존의 방법보다 기억장소의 낭비를 줄이고 계산의 고려 대상을 줄이고 있다. 이 알고리즘의 기억장소 요구도는 O(n)으로 대규모 집적회로에 대한 테스트 생성에 효율적이다.

앞으로의 연구 과제는 본 개발 시스템을 순서회로에 적용할 수 있게 확장하는 것과 테스트 패턴 생성시스템(ATPG)과 결합하는 방안이 요구된다.

참 고 문 헌

[1] G. Richard, "Special Report on Systems Testing," Computer Design, pp. 53, Oct. 1985.

[2] H. Fujiwara, Logic Testing and Design for Testability, MIT Press, pp. 1-129, 1985.

[3] A. Miczo, Digital Logic Testing and Simulation, Harper & Row, Publishers, Inc., pp. 118-183, 1986.

[4] M. A. Breuer and A. D. Friedman, Diagnosis & Reliable Design of Digital Systems, Computer Science Press, Inc., pp. 224-242, 1976.

[5] H. Y. Chang, S. G. Chappell, C. H. Elmensdorf, and L. D. Schmidt, "Comparison of parallel and deductive fault simulation methods," IEEE Trans.Comput., vol. C-23, pp. 1132-1138, Nov. 1974.

[6] P. R. Menon and S. G. Chappell, "Deductive fault simulation with functional blocks," IEEE Trans. Comput., vol. C-27, pp. 689-695, Aug. 1978.

[7] W. Ke, S. Seth, and B. B. Bhattacharya, "A fast fault simulation algorithm for combinational circuits," Proc. of IEEE ICCAD-88, pp. 166-169, 1988.

[8] J. P. Caisso and B. Courtois, "Fault simulation and test pattern generation at the multiple-valued switch level," IEEE ITC, pp. 94-101, 1988.

[9] W. Daehn and M. Geilert, "Fast fault simulation for combinational circuits by compiler driven single fault propagation," IEEE ITC, pp. 286-292, 1987.

[10] G. M. Silberman and I. Spillinger, "Test generation using functional fault simulation and the difference

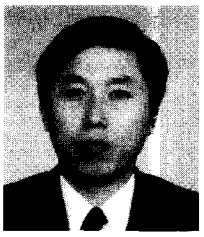
fault model," IEEE ITC, pp. 400-407, 1987.

[11] A. Goundan and J. P. Hayes, "Identification of Equivalent Fault in Logic Networks," IEEE Trans. Comput., vol. C-29, pp. 978-985, Nov. 1980.

[12] M. H. Schulz, E. Trischler, and T. M. Sarfert, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System," IEEE Trans. on Computer-Aided Design, vol. 7, no. 1, pp. 126-137, Jan. 1988.

[13] S. Patil and P. Banerjee, "Fault partitioning issues in an integrat-ed parallel test generation/fault simulation environment," IEEE ITC, pp. 718-726, 1989.

□ 著者紹介



권 기 창(Ki-Chang Kwon)

1962년 3월 27일생
1985년 2월 안동대학교 행정학과 행정학사
1985년 9월-1993년 2월 안동대학교 전자계산소 근무
1993년 2월 대구대학교 정보처리학과 경영학석사
1995년 현재 안동전문대학 사무자동화과 전임강사

※ 주관심분야 : 소프트웨어 개발 및 응용컴퓨터 그래픽스

백 덕 화(Deuk-Hwa Baek)

1962년 10월 14일생
1988년 2월 : 경북대학교 전자공학과 공학사
1990년 2월 : 경북대학교 컴퓨터공학과 공학석사
1995년 현재 : 경북대학교 컴퓨터공학과 박사과정 수료
1991년 4월 ~ 현재 : 창원전문대학 전자계산과 조교수

※ 주관심분야 : 컴퓨터 그래픽스 및 CAD, 테스팅 및 컴퓨터 구조 등

권 기 룡(Ki-Ryong Kwon)

1960년 2월 10일생
1986년 2월 : 경북대학교 전자공학과 공학사
1990년 2월 : 경북대학교 전자공학과 공학석사
1994년 8월 : 경북대학교 전자공학과 공학박사
1986년 3월 ~ 1988년 3월: 현대자동차 승용생산기술연구부 근무
1991년 9월 ~ 현재 : 창원전문대학 전자통신과 조교수

※ 주관심분야 : 적응 신호처리, 음향통신 및 소음제어 등