論文99-36C-9-5

조합회로에 대한 고장 진단 검사신호 생성

(Diagnostic Test Pattern Generation for Combinational Circuits)

朴 英 鎬 *, 閔 炯 福 **, 李 宰 勳 ***, 申 容 煥 **

(Young Ho Park, Hyoung Bok Min, Jae Hoon Lee, and Yong Whan Shin)

요 약

조합회로에 대하여 고장 진단 검사신호를 생성하는 것은 매우 어려운 문제로 남아있다. ISCAS85 c7552 회로의 경우, 약 1억 개의 고장짝(Fault pair)을 가지고 있기 때문에 적은 노력과 시간을 사용하여 좀 더 많은 정보를 얻을 필요가 있다.

본 연구에서는 새로운 고장 진단 검사신호 생성 알고리즘이 제안되었고 구현되었다. 또 새로운 알고리즘과의 비교를 위하여 PODEM을 기본으로 하여 DIATEST 알고리즘을 구현하였다. 구현된 두 개의 알고리즘을 ISCAS85 회로에 적용하여 실행하였으며, 그 결과 두 개 알고리즘 모두 99%의 고장 진단율을 보였으나, 새로운 알고리즘은 각 고장의 검출 및 구별을 동시에 고려하기 때문에 고장 진단 사전과, 고장 진단 결정 트리의 제작시 더 많은 정보를 줄 수 있다. 또한 검사신호의 수를 비교했을 때 새로운 고장 진단 검사신호 생성 알고리즘이 절반이하의 개수만이 필요했다. 다만, vector 생성 속도에 있어서는 DIATEST가 우수하였다.

Abstract

Generating diagnostic test patterns for combinational circuits remain to be a very difficult problem. For example, ISCAS85 c7552 benchmark circuit has 100 million fault pairs, Thus, we need more sophisticated algorithm to get more information.

A new diagnostic algorithm for test pattern generation is suggested and implemented in this paper. DIATEST algorithm based on PODEM is also implemented for comparison to the new algorithm. These two algorithms have been applied to ISCAS85 benchmark circuits. Experimental results show that (1) both algorithms achieve fault pair coverage over 99%, (2) total test length of the new algorithm is much shorter than that of DIATEST, and (3) the new algorithm gives much more information used for making diagnostic dictionary, diagnostic decision tree or diagnostic test system despite DIATEST is faster than the new algorithm.

I. 서 론

반도체 회로의 검사(test)란 설계된 회로에 대하여 정상적인 동작여부를 확인하는 과정으로서 주입력(primary inputs)에 검사 벡터를 인가한 후 정상 출력과 동일한 결과가 나오는지를 주출력(primary outputs)에서 확인하는 과정으로 이루어진다. 이때 주입력에 인가하는 검사 입력의 생성은 회로 내에 고장(fault)이 있다고 가정하고 이러한 일련의 작업을 특정 알고리즘에 의해 자동으로 생성하는데 이것을 검사신호 자동생성(ATPG :

* 正會員, 韓國電子通信研究院
(Electronics and Telecommunications Research Institute)
** 正會員, 成均館大學
(SungKyunKwan University)
*** 正會員, 麗州大學
(YeoJoo Institute of Technology)
※ 본 연구는 정보통신부의 1998년도 대학기초 연구지원 사업으로 연구되었음.
接受日字 : 1999年3月8日, 수정완료일 : 1999年8月19日

Automatic Test Pattern Generation)[1]이라 한다.

그러나 검사신호 자동생성에 의한 검사 입력은 회로 내에 존재하는 고장을 검출 할 수 있는지에 대한 여부는 가능하지만 문제가 되는 그 고장의 회로 내 위치와 원인은 알 수가 없다. 칩 면적(chip size)이 증가하고 공정기술의 복잡화로 고장 존재 시 고장의 위치와 원인을 찾는 일은 매우 어려워지고 있지만 칩의 빠른 제품화와 높은 수율(yield)을 위해서는 매우 중요하며 그 필요성이 증대되고 있다. 이와 같이 고장의 위치와 원인을 찾는 일련의 과정을 고장 진단(Fault diagnosis)[2]이라 하며 고장 진단 문제를 해결하기 위해 다음과 같은 방법이 사용되고 있다.

첫 번째 방법인 동적 고장 진단(Dynamic diagnosis)[3]은 효과-원인 고장 진단(Effect-cause diagnosis)[3]이라 불리기도 하는데 고장 투여(Fault dropping)와 고장 시뮬레이션(Fault simulation)을 통하여 관측된 출력(Effect)에 대한 정보를 이용하여 고장 원인(Cause)의 위치를 찾아내는 방법이다. 두 번째 방법은 원인-효과 고장 진단(Cause-effect diagnosis)[3]이라 불리며 원인에 따른 그 결과에 대한 정보를 가지는 고장 진단 사전(Diagnosis dictionary)[4]를 먼저 산출하고 고장 진단 결정 트리(Diagnosis decision tree)[5]를 만들게 된다. 이를 이용하여 고장의 위치에 대한 범위를 좁혀나감으로써 고장의 위치를 찾도록 한다.

위와 같은 회로 고장진단 해결책 중에서 본 연구는 조합회로에 대해 단일 고장 가정 하에 2개의 고착 고장(Stuck-at fault)을 대상으로 2개의 고장을 구별하는 검사신호를 생성하는 고장 진단 검사신호 생성을 위한 새로운 알고리즘의 개발하고자 한다.

Ⅱ. 고장진단 이론

1. 고장진단의 개념

고장 진단 검사신호는 기본적으로 주출력에 두 개의 고장으로 인하여 변형된 회로의 출력이 서로 다르게 나오게 하는 데에 목적이 있다. 이를 부울리안 미분법(Boolean difference method)을 이용하여 설명하면 조합회로 내에 단일 고착 고장 가정 하에서 두 개의 고착 고장 α와 β가 어떤 검사신호들로부터 찾아질 수 있지만 그 검사신호들로는 두 개의 고장이 구별이 되지 않는다고 하자.

n개의 입력을 갖는 j번째 출력 함수를 $F_j(p1, p_2, \cdots, p_1)$라 할 때, 단독 고착 고장 α는 그 출력 함수를 $F_{\alpha j}(P)$로 바꾸고 단독 고착 고장 β는 $F_{\beta j}(P)$로 바꾼다고 할 때, 검사 신호 P가 식(1)과 같은 부울리안 방정식을 만족시킬 경우 검사 신호 P는 두 개의 고장을 구별한다고 말한다.

$$F_{\alpha j}(P) \oplus F_{\beta j}(P) = 1 \tag{1}$$

이 방정식은 만일 고장 α와 고장 β가 j번째 출력에서 각각 다른 값을 가진다면 j번째 출력에서 두 개의 고장을 구별할 수 있다는 것을 가리킨다.

만일 고장 α를 도선 k의 μ-고착고장이라 하고 고장 β를 도선 l의 ν-고착고장이라 하자. $(\mu, \nu \in \{0, 1\})$ 도선 k와 l에 대한 부울리안 함수를 $k(P)$, $l(P)$라 할 때, 부울 대수를 이용하여 j번째 출력에서 두 개의 고장을 구별하는 모든 검사신호를 식(2)와 같이 부울리안 미분법을 사용하여 나타낼 수 있다.

$$k(P)_{-\mu}\frac{dF_j(P)}{dk} \oplus l(P)_{-\nu}\frac{dF_j(P)}{dl} = 1 \tag{2}$$

여기서 $k(P) - \mu[l(P) - \nu]$는 도선 $k[l]$의 고착고장 $\mu[\nu]$에 반대되는 논리를 인가하는 모든 검사신호 P에 대해서 논리 1을 가지게 되고 $dF_j(P)/dk[dF_j(P)/dl]$은 도선 $k[l]$의 논리 변화가 j번째 출력에 영향을 미치는 모든 검사신호 P에 대해서 논리 1을 가지게 된다.

2. Δ-알고리즘

Δ-알고리즘[2]은 자동 고장 검출 검사신호 생성 알고리즘의 하나인 PODEM (Path-Oriented Decision Mak-

표 1. Δ-알고리즘의 논리 모델

Table 1. Logic model of Δ-algorithm.

#	G	F_1	F_2		#	G	F_1	F_2		#	G	F_1	F_2	
1	0	0	0		10	1	0	0		19	X	0	0	
2	0	0	1	Δ	11	1	0	1	Δ	20	X	0	1	Δ
3	0	0	X	δ	12	1	0	X	δ	21	X	0	X	δ
4	0	1	0	Δ	13	1	1	0	Δ	22	X	1	0	Δ
5	0	1	1		14	1	1	1		23	X	1	1	
6	0	1	X	δ	15	1	1	X	δ	24	X	1	X	δ
7	0	X	0	δ	16	1	X	0	δ	25	X	X	0	δ
8	0	X	1	δ	17	1	X	1	δ	26	X	X	1	δ
9	0	X	X	δ	18	1	X	X	δ	27	X	X	X	δ

ing)[7]을 바탕으로 하여 정상 회로와 각각의 고장으로 인하여 변형된 회로를 동시에 고려하여 두 고장으로 인한 논리값의 차이로 정의된 Δ를 출력으로 전달하도록 고안된 알고리즘이다.

논리모델

만일 어떤 회로의 도선 l_1에 고장 F_1이, 도선 l_2에 고장 F_2가 있다고 가정할 경우 각 도선에 인가되는 논리값의 경우에 따라 표 1과 같은 논리 모델이 만들어지며, F_1로 인해 변형된 회로와 F_2로 인해 변형된 회로에서 나타나는 논리값의 차이에 대해 다음과 같이 정의한다.

Δ : 두 개의 고장 회로에 논리값의 차이가 존재하는 도선을 표시.

δ : 두 개의 고장 회로에 논리값의 차이가 발생할 가능성을 가지는 도선을 표시.

알고리즘

전체적 알고리즘은 2개의 고장 중 하나를 선택하여 그 고장을 찾을 수 있는 검사신호를 생성한 후, 생성된 검사신호에서 X값을 Δ가 출력으로 전달될 수 있도록 0 또는 1로 결정하는 구조를 가지고 있다. 즉, Δ-알고리즘은 다음과 같은 2단계를 거쳐 적용된다.

(1) 특정 도선에 Δ가 나타나도록 한다.

(2) δ-path를 따라 Δ를 출력으로 전파시킨다.

따라서 첫 번째 검사신호를 사용하여 성공했을 경우 최소 1개의 검사신호로써 1개의 고장을 찾고 단순히 구별만을 할 수 있으며 최대 2개의 검사신호로써 2개의 고장을 찾고 구별을 할 수 있다.

3. DIATEST

DIATEST[3]는 두 개의 고장으로 인한 회로의 변형을 표시하기 위하여 9가지 논리 모델을 사용하며 자동 고장 검출 검사신호의 하나인 CONTEST[8]에 기반을 둔 알고리즘이다.

DIATEST가 9개 논리 모델을 사용한 이유는 두 개의 고장으로 생성된 각각의 회로를 구별하기 위해서이다. Δ-알고리즘에서와 같이 정상 회로와 고장 1, 2로 인하여 변형된 회로의 모든 경우에 대해서 따로 논리값을 할당하는 것과 달리 정상 회로의 논리값을 제외한 각 경우에 대하여 하나의 논리값으로 정의하여 표 2와 같은 논리 모델을 가지며, 두 회로의 논리값의 경우에 따라 같은 상태, 다른 상태, 부분 정의된 상태, 정의되지 않은 상태의 값으로 분류한다. 9가지 논리 모델에 있어서의 논리 연산은 식(2.3)과 같이 이루어진다.

9개 논리 모델

표 2.DIATEST의 9개 논리 모델

Table 2. 9-valued logic model of DIATEST.

논리	분류	논리	분류
(0,0)	같은 상태 (Non-different value)	(X,0)	부분 정의된 상태 (Partly unspecified)
(1,1)	같은 상태 (Non-different value)	(X,1)	부분 정의된 상태 (Partly unspecified)
(0,1)	다른 상태 (Different value)	(1,X)	부분 정의된 상태 (Partly unspecified)
(1,0)	다른 상태 (Different value)	(0,X)	부분 정의된 상태 (Partly unspecified)
		(X,X)	정의되지 않은 상태 (Unspecified value)

$$(b1_\alpha, b1_\beta) \circledcirc (b2_\alpha, b2_\beta) = [(b1_\alpha \circledcirc b1_\beta), (b2_\alpha \circledcirc b2_\beta)] \quad (3)$$

(◎는 AND, OR, NAND, NOR..가 될 수 있다.)

알고리즘

먼저 비분기영역(Fanout free region)내 기능적 동일(Functionally equivalent) 고장들로 이루어진 고장 집단을 결정한다. 각 고장 집단에 대해 하나의 고장으로 생각하여 고장 집합(Fault list)을 생성한다. 구성된 고장 집합에 대해서 고장 검출 검사신호 생성 알고리즘인 CONTEST 알고리즘을 수행하여 완전한 고장 검출 검사신호를 구하고 만일 구하지 못할 경우 과잉 고장(Redundant fault)으로 간주하여 고려 대상에서 제외시킨다. 과잉 고장이 제거된 고장 집합 내의 한 쌍을 골라 각 고장을 찾는 검사신호를 차례로 인가하여 출력에 각기 다른 상태의 값이 보여지는가를 살펴본다. 만일 두 검사신호에 의해 다른 값이 보이지 않을 경우 DIATEST 알고리즘을 수행하여 찾도록 한다. 따라서 자동 고장 검출 검사신호의 생성과정까지 합칠 경우 DIATEST는 최소 2개의 검사신호로써 2개의 고장을 찾고 1개의 위치를 알 수가 있으며 최대 3개의 검사신호로써 2개의 고장을 찾고 1개의 위치를 알 수가 있다.

Ⅲ. 새로운 고장 진단 검사신호 생성 알고리즘

새로운 고장 진단 검사신호 생성 알고리즘을 설명하기에 앞서 타 알고리즘과의 비교를 위하여 본 연구에서는 각 고장짝의 구분을 다음 정의와 같이 완전 구분과 단순 구분으로 구별하여 차이를 두었다.

(1) 완전 구분(Fully distinguish) : 하나의 검사신호로 정상회로와 각 고장으로 인하여 변형된 값이 구별되는 두 개의 고장의 효과를 각기 다른 출력으로 전달하여 고장을 구분

(2) 부분 구분(Partly distinguish) : 둘 중 어느 하나의 고장 효과만을 출력으로 전달하여 고장을 구분

따라서 앞서 이야기되었던 기존 해결책인 DIATEST는 9개 논리 모델과 자동 고장 검출 검사신호 알고리즘의 기법들을 사용하여

(1) 최소2개에서 최대3개의 검사신호를 생성하여 2개의 고장을 찾고 부분 구분

(2) 고장 진단 검사신호 알고리즘을 수행하는 시간 외 기본적으로 고장 검출 검사신호 알고리즘을 수행하는데 필요한 추가적 시간이 필요

와 같이 분석될 수 있다. 그리고, DIATEST의 9개 논리 모델은 단순하게 2개의 고장으로 인해 변형된 회로의 논리 차이를 단순하게 나누어 사용하기 때문에 다른 상태의 논리값이 출력으로 전달되었다 하더라도 정상회로의 출력과 구별되지 않기 때문에 고장 진단 사전 제작시 정상회로의 출력에 대한 정보를 얻기 위한 추가적 계산이 필요하다.

Δ-알고리즘의 경우 27개의 값을 가지는 논리 모델을 제시하고 있다. 이로 인하여 게이트 출력 논리값을 구하는데 많은 수행 시간이 필요할 것이며 기존의 고장 검출 검사신호 자동 생성 알고리즘을 적용하는데 있어서도 제약점이 따른다. 그리고, DIATEST와 마찬가지로 부분 구분만을 목적으로 하기 때문에 DIATEST에서 보이는 단점을 또한 함께 가지고 있다.

1. 새로운 9개 논리 모델

앞서 언급된 기존에 연구되어진 알고리즘의 문제점을 개선하기 위해 본 연구에서는 표 3과 같은 새로운 9개 논리 모델을 사용하였다. 표 3에서의 설명 중 첫 번째 항(G)은 정상 회로의 논리값이고 두 번째 항(F1)과 세 번째 항(F2)은 각각 첫 번째, 두 번째 고장에 의해 변형된 회로의 논리값이다. 여기서 D와 $\overline{D}$는 각 고장으로 인해 생성된 두 개의 회로에 검사신호를 인가했을 때에 각 고장으로 인한 효과로 나타나는 논리값이 동일하게 나타나는 경우를 나타내며 이 값이 출력으로 전달되었을 경우 구분을 할 수 없으므로 고장 검출의 D와 $\overline{D}$와는 구별되어야 한다. 새로운 9개 논리 모델의 장점은 정상 회로의 논리도 함께 표현된다는 점이다. 정상 회로의 논리와 이를 이용하여 각 고장에 대하여 $D_1(\overline{D_1})$ 또는 $D_2(\overline{D_2})$를 발생시켜 출력으로 전달시키는 방법을 사용하여 완전 구분하도록 한다.

표 3. 새로운 9개 논리 모델
Table 3. New 9-valued logic model.

논리값	설명(G/F1/F2)	논리값	설명(G/F1/F2)
0	(0/0/0)	D_1	(1/0/1)
1	(1/1/1)	$\overline{D_1}$	(0/1/0)
D	(1/0/0)	D_2	(1/1/0)
$\overline{D}$	(0/1/1)	$\overline{D_2}$	(0/0/1)
X	(X/X/X)		

2. 알고리즘

기존 고장 진단 알고리즘에서는 단일 출력에 대해서만 고려를 한 반면 새로 제안하는 알고리즘(NEON : NEO-Nine valued logic model diagnostic test pattern generation)은 복수 출력 회로에 대해서도 고려를 하고 있으며, 오히려 기본적으로 2개 이상의 출력에 대해서만 알고리즘의 주목적을 달성할 수 있다. 그 이유는 두 개의 고장으로 인한 출력의 변화에 대한 효과를 각기 다른 출력으로 전달시킴으로써 가장 좋은 경우 하나의 검사신호로써 두 개의 고장을 찾고 두 개의 고장을 각각 구별할 수 있기 때문이다.

본 연구의 새로운 알고리즘은 다음 그림 1과 같이 3개의 단계를 거치게 된다. 먼저 1단계인 2개의 고장 검출 및 구별 단계에서는 본 알고리즘의 목표인 2개의 고장에 대해서 각 효과를 각기 다른 출력으로 내보내는 단계를 가지게 된다. 2단계는 첫 번째 단계가 실패

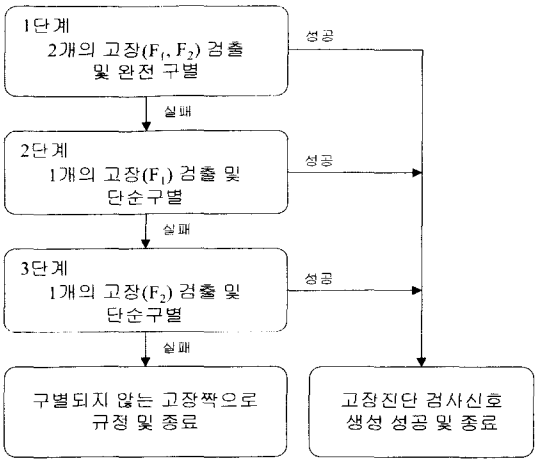


그림 1. NEON 전체 흐름도
Fig. 1. Flow chart of NEON.

했을 경우에 실행되며 두 개의 고장 중 하나의 고장의 효과만을 출력으로 전달하도록 한다. 3단계는 2단계가 실패했을 경우에 실행되며 두 번째 단계에서 선택되어지지 않은 고장의 효과만을 출력으로 전달하도록 한다. 이 3단계 또한 실패했을 경우 구별되지 않는 고장짝(Undistinguish fault pair)으로 규정하며 알고리즘을 마치도록 한다.

1단계 : 2개의 고장 검출 및 완전 구분

1단계는 크게 두 개의 과정으로 나누어진다. 첫 번째 과정은 두 개의 고장 효과를 발생시키며 두 번째 과정은 각 효과를 출력으로 전달한다.

첫 번째 과정은 그림 2와 같이 고장 1의 고장 효과를 발생시키고 나머지 고장 2의 고장 효과를 발생시키도록 한다. 이를 실패했을 경우 회로 내 할당되어 있는 논리값을 리셋(Reset)하고 앞의 경우와 반대로 고장2의 고장 효과를 먼저 발생시키고 나머지 고장1의 고장 효과를 발생시킨다. 이 또한 실패했을 경우 다시 회로에 할당되어 있는 논리값을 리셋하고 2단계로 넘어가게 된다.

두 번째 과정인 고장 효과 전달은 그림 3에서 보이는 것과 같이 첫 번째 과정에서 발생한 고장 효과로 인하여 발생한 각 D 프론티어를 구하고 이로부터 X-경로 체크(X-path check)를 수행하여 두 개의 고장 효과 모두가 출력으로 전달될 수 있는가를 검사한다. 여기서 두 개의 고장 효과중 하나라도 출력으로 전달될 수 없으면 백트랙을 수행하여 다른 선택을 하도록 한 후에 새로운 입력을 정하도록 하고 반대의 경우 두 개

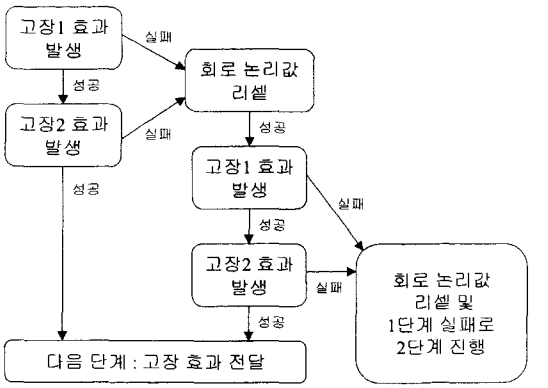


그림 2. NEON 고장 효과 발생 흐름도
Fig. 2. Flow chart of fault effect excitation.

의 고장 중 임의의 고장 D 프론티어를 잡아 역추적(Backtrace)을 하여 새로운 입력을 정한 후 인가하여 그 효과가 출력으로 전달되었는가를 검사한다. 출력이 전달되지 않았다면 다른 고장의 효과로 인한 D 프론티어와 X-경로가 존재하는가를 매 입력 결정 및 인가 후에 검사하면서 다른 입력을 선택하도록 한다. 만일 처음 고장의 효과가 하나의 출력으로 전달되었을 경우, 앞의 경우와 마찬가지로 역추적과 입력 인가의 방법을 사용하여 검사신호를 정해 남은 하나의 고장 효과를 전달하도록 한다.

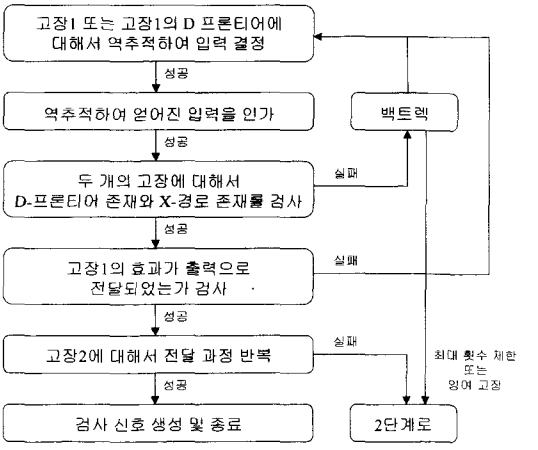


그림 3. NEON 고장 효과 전달 흐름도
Fig. 3. Flow chart of fault effect propagation.

2, 3단계 : 1개의 고장 검출 및 단순 구분

2 또는 3단계는 먼저 그림 4와 같이 1단계 수행 중 하나의 고장 효과만을 출력으로 전달하는 검사 신호의

존재 유무를 검사하여 존재할 경우에는 그것을 검사신호로 삼아 알고리즘 수행을 종료한다. 만일 존재하지 않을 경우에는 임의의 하나의 고장에 대하여 PODEM 알고리즘을 새로운 9개 논리 모델에 맞추어 수행하여 하나의 고장 효과만을 출력으로 전달하는 검사신호를 생성하도록 한다. 이것이 만일 실패했을 경우, 2단계 수행 중이었을 경우에는 3단계로 이동하여 다른 고장에 대하여 같은 작업을 반복하도록 한다.

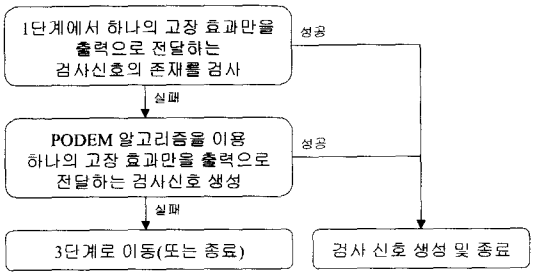


그림 4. NEON 2,3단계 흐름도
Fig. 4. Flow chart of second and third level of NEON.

3. 새로운 고장 진단 검사신호 알고리즘 NEON의 예

그림 5의 회로에 도선 1과 도선 5의 0-고착고장을 본 연구에서 제안하는 알고리즘을 사용하여 구분하여 보자. 먼저 1단계의 첫 번째 과정으로 고장 효과 발생 과정을 수행하면 도선 5에 발생한 고착고장의 효과를 발생시키기 위해서 주입력단 PI_0와 PI_2에 1을 할당하고 함축 과정을 수행하면,

(1) $PI_0 : 1 \rightarrow 0 : 1$

(2) $PI_2 : 1 \rightarrow 2 : 1 \rightarrow 11-1 : 1 \rightarrow 5 : \overline{D_1}$

와 같이 도선 5에서 고장으로 인한 효과가 $\overline{D_1}$와 같이 나타난다. 여기서 일반 PODEM 알고리즘의 경우 이 고장의 효과를 출력으로 전달하는 과정으로 진행되지만 본 알고리즘에서는 1단계의 두 번째 과정으로 도선 1에 발생한 고착고장의 효과를 발생시키기 위해서 주입력단 PI_1 0을 할당하고 함축 과정을 수행한면,

(1) $PI_1 : 0 \rightarrow 1 : \overline{D_2}$

와 같이 도선 1에 $\overline{D_2}$이 할당됨을 알 수 있다.

1단계의 첫 번째 과정인 고장 효과 발생 과정이 성공적으로 완료되었으므로 다음 과정인 고장 효과 전달 과정을 수행하면 먼저 도선 5의 고장의 효과인 $\overline{D_1}$을

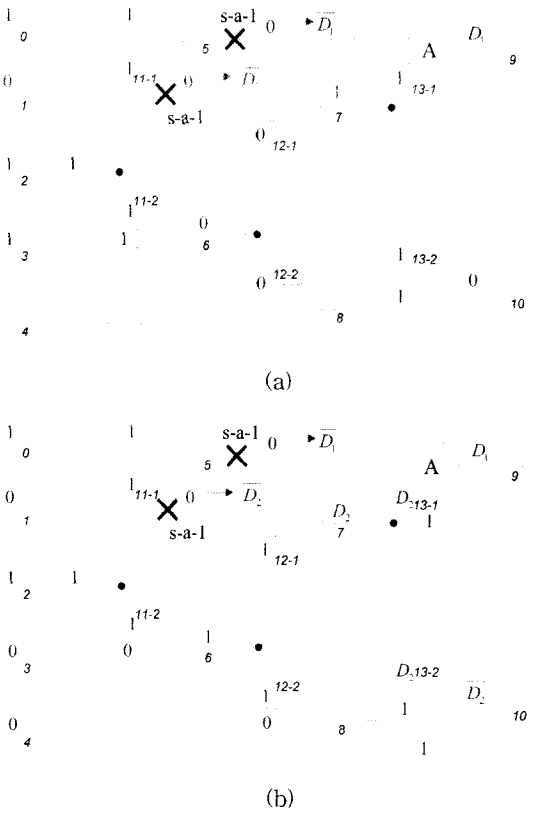


그림 5. NEON 예제
Fig. 5. Example of NEON.

출력으로 전달하기 위하여 도선 13-1에 부-제어(non-controlling) 논리값인 1을 할당하여야 한다. 이를 위하여 역추적의 결과로 주입력단 PI_3에 1을 할당하면 그 결과는 그림 5 (a)와 같이

(1) $PI_3 : 1(3:1) \rightarrow 6 : 0 \rightarrow 12-1 : 0 \rightarrow 7 : 1 \rightarrow 13-1 : 1 \rightarrow 9 : D_1(PO_3 : D_1)$

(2) $6 : 0 \rightarrow 12-2 : 0 \rightarrow 8 : 1$

(3) $13-2 : 1 \rightarrow 10 : 0(PO_{10} : 0)$

와 같이 되어 도선 5의 고장의 효과가 출력 PO_9로 전달되었으나 도선 1의 고장의 효과가 다른 출력으로 전달되지 못하게 되었다. 여기서 PODEM 알고리즘의 경우 고장의 효과를 찾았으므로 알고리즘을 종료하고 DIATEST의 경우 부분구별을 했으므로 알고리즘을 종료한다. 그러나 본 알고리즘 NEON에서는 다른 하나의 고장 효과를 전달하지 못했으므로 종료하지 않고 백트렉을 수행하게 된다. 따라서 주입력단 PI_3에 0을 할당

하게 되고 그 결과는 그림 5(b)와 같이

(1) $PI_3 : 0(3:0) \rightarrow 6:1 \rightarrow 12-1:1$
$\rightarrow 7 : D_2 \rightarrow 13-1 : D_2 \rightarrow 9 : D_1 (PO_3 : D_1)$

(2) $6:0 \rightarrow 12-2:0 \rightarrow 8:1$

(3) $13-2:1 \rightarrow 10:0 (PO_{10}:0)$

와 같이 되어 도선 5의 고장 효과가 출력 PO_9로 전달되었고 도선 1의 고장 효과도 전달 출력으로 전달될 수 있는 X-경로를 가짐을 알 수 있다.

이제 마지막 과정으로 남은 도선 1의 고장으로 인한 효과를 아직 논리값이 정해지지 않은 출력으로 보내는 과정이 남았다. 도선 1의 고장 효과를 출력 PO_{10}으로 전달하기 위해서는 도선 8에 1이 와야한다. 이를 위하여 역추적을 통해 주입력 PI_4에 0이 할당되어야 함을 알 수 있으며 함축 과정을 수행하게 되면 그림 5(b)에서 보이는 것과 같이

(1) $PI_4 : 0 \rightarrow 8:1 \rightarrow PO_2 : \overline{D_2}$

가 되어 $\overline{D_2}$가 출력 PO_{10}에 할당되어 도선 1의 고장을 찾으며 도선 5의 고장과 완전 구별할 수 있음을 알 수 있다.

Ⅳ. 구현 및 결과

본 연구의 새로운 고장 진단 검사신호 생성 알고리즘 NEON과 DIATEST는 C언어로 구현되었으며 128MB의 주메모리를 갖는 Sun Ultra Spac-1 workstation에서 ISCAS85 측정기준 회로(Benchmark circuit)를 대상으로 수행되었다.

전체 고장 집합의 수가 n인 회로에 대해 ${}_nC_2$ 개의 고장짝에 대해서 수행을 해야만 한다. 그러나 c7552 회로의 경우 전체 고장 집합의 수가 15106개이므로 전체 고려해야할 고장짝의 수는 114088065개나 된다. 따라서 이것을 계산하는데는 무리가 있으므로 99%의 신뢰성을 갖도록 무작위로 표본을 추출하여 수행하도록 하였다. 99%의 신뢰성을 갖기 위해 필요한 표본의 크기 n는

$$n = \frac{(z_{\alpha/2})^2 p(1-p)/d^2}{1 + (z_{\alpha/2})^2 p(1-p)/d^2 N} \tag{4}$$

와 같이 구할 수 있다. 여기에서 $Z_{\alpha/2}$는 정규분포지수이며 p는 표본평균을, d는 오차한계, N은 모집단의 수이다. 따라서 c7552의 경우 고려해야할 고장짝은 16631개가 된다.

DIATEST은 원래 CONTEST 알고리즘을 기반으로 하여 구현되었으나 본 연구에서 제시한 알고리즘과 동등한 상태에서의 평가를 위해 PODEM을 기반으로 하여 구현되었다.

표 7은 비교대상으로 삼은 DIATEST와 연구의 알고리즘이 실행된 전체 CPU time의 비교표이다. 보는 바와 같이 대부분의 회로에서 실행시간이 더 필요하다는 것을 알 수 있다. 표 가장 밑에 보이는 가중평균이란 각 회로의 네트수에 실행시간을 곱한 것을 모두 더하여 각 회로의 네트수의 합으로 나눈 평균을 말이며, 전체적인 비교에 사용된다.

표 7. 실행시간
Table 7. Total CPU time.

회로	NEON(sec)	DIATEST(sec)
c17	0.08	0.08
c95	5.47	4.00
c432	494.78	120.92
c499	460.48	346.48
c880	167.97	97.63
c1355	1312.60	1097.70
c1908	733.68	361.92
c2670	373.03	351.33
c3540	1155.00	697.03
c5315	528.30	532.33
c6288	2835.93	1816.90
c7552	869.00	1191.48
가중 평균	1177.00	952.35

이를 식 (5)를 이용하여 하나의 고장짝당 소요된 계산 시간을 산출하여 보면 표 8에서 보는 바와 같이 c7552를 제외하고는 많은 계산 시간이 필요함을 알 수 있다.

$$\frac{\text{CPU Time}}{\text{Fault Pair}} = \frac{\text{CPU Time used for ATPG and DTPG}}{\text{Number of Total Fault Pair}} \tag{5}$$

표 8. 한 개의 고장짝에 대한 실행시간
Table 8. Average CPU time for one fault pair.

회로	NEON(ms)	DIATEST(ms)
c17	5.14	0.37
c95	1.42	1.05
c432	33.50	8.48
c499	29.45	2.46
c880	10.76	6.58
c1355	80.09	67.35
c1908	44.69	22.30
c2670	23.39	23.79
c3540	69.88	42.42
c5315	32.59	33.62
c6288	170.77	19.57
c7552	53.22	73.42
가중 평균	71.53	58.50

식 (6)는 고장짝 진단율(Fault pair coverage)를 구하는 식이고 표 9는 이를 이용하여 본 연구의 알고리즘과 DIATEST를 수행하여 얻은 결과를 비교한 것이다. 보는 바와 같이 c499와 c6288을 제외하고는 두 개의 알고리즘 모두 99%에 이르는 진단율을 보이고 있음을 알 수 있다.

표 9. 진단율
Table 9. Fault pair coverage.

회로	NEON(%)	DIATEST(%)
c17	100.00	100.00
c95	99.83	99.38
c432	99.53	99.06
c499	99.54	97.54
c880	99.86	99.91
c1355	99.07	99.23
c1908	99.50	99.61
c2670	99.39	99.31
c3540	98.84	98.24
c5315	99.84	99.83
c6288	97.29	98.65
c7552	99.52	99.46

$$\text{Fault Pair Coverage} = \frac{\text{Number of Distinguished Fault Pair}}{\text{Number of Total Tested Fault Pair}} \quad (6)$$

표 10. 평균 검사신호
Table 10. Average number of test pattern.

회로	NEON	DIATEST
c17	1.00	2.01
c95	1.00	2.04
c432	1.00	2.04
c499	1.00	2.11
c880	1.00	2.06
c1355	1.00	2.13
c1908	1.00	2.12
c2670	1.00	2.11
c3540	1.00	2.11
c5315	1.00	2.11
c6288	1.00	2.11
c7552	1.00	2.12
가중 평균	1.00	2.11

$$\frac{\text{Number of Average Test Vectors}}{\text{Fault Pair}} = \frac{\text{Number of Total Test Vectors Used in Distinguishing Fault Pair}}{\text{Number of Distinguished Fault Pair}} \quad (7)$$

식 (7)는 두 개의 고장을 구별하는데 필요한 평균 검사신호를 구하는 식이며 표 10은 이를 이용하여 본 연구의 알고리즘과 DIATEST를 수행하여 얻은 결과를 비교한 것이다. 결과에서 볼 수 있듯이 새로운 고장진단 알고리즘에서는 평균 단 1개의 검사신호를 사용하여 고장진단을 할 수 있는데 반해 DIATEST는 기본적으로 2개의 검사신호가 필요하기 때문에 새로운 고장진단 알고리즘에 비해 2배가 넘는 수치를 보이고 있다.

표 11은 새로운 고장진단 알고리즘 수행 결과 두 개 고장을 어떻게 구분했는가를 나타내는 것이다. 결과와 같이 새롭게 제안된 알고리즘을 이용할 경우 c880회로 이후로 c1355와 c3540을 제외하고는 출력의 수가 많고 게이트의 수가 큰 회로에 대해서 전체 고장짝 중 약 60~80% 정도의 고장짝에 대해서 완전 구분함을 알 수 있다.

표 12는 DIATEST 수행 중 검사신호 생성 시기에 대한 분포를 나타내고 있다. 결과에서 볼 수 있듯이 60% 이상이 첫 번째 PODEM 검사신호로 구별이 되며 90% 가까운 고장이 PODEM으로 생성된 검사신호로 구별이 되고 있음을 알 수 있다.

표 11. NEON 구별 분포
Table 11. Distribution of NEON's distinguish.

회로	완전 구분	%	부분 구분	%
c17	136	59.91	91	40.09
c95	2274	56.23	1770	43.77
c432	7518	50.90	7251	49.10
c499	9776	62.44	5880	37.56
c880	15028	93.84	987	6.16
c1355	8212	50.84	8055	49.52
c1908	13275	80.97	3126	19.06
c2670	12699	77.12	3767	22.88
c3540	12863	78.43	3537	21.57
c5315	14671	88.41	1923	11.59
c6288	15589	96.34	592	3.66
c7552	14683	88.71	1868	11.29

표 12. DLATEST 검사신호 생성 시기 분포
Table 12. Distribution of test pattern generation time of DLATEST.

회로	1번째 PODEM 검사신호	%	2번째 PODEM 검사신호	%	DIATEST Vector	%
c17	153	67.40	72	31.72	2	0.88
c95	3266	81.12	579	14.38	181	4.50
c432	9640	65.58	4518	30.74	541	3.68
c499	10812	70.48	2892	18.85	1637	10.67
c880	10729	66.96	4273	26.67	1021	6.37
c1355	10418	63.94	3799	23.32	2077	12.75
c1908	10241	62.37	4164	25.36	2016	12.28
c2670	10461	63.58	4127	25.08	1865	11.34
c3540	9859	60.48	4676	28.69	1766	10.83
c5315	10320	62.19	4481	27.01	1792	10.80
c6288	10540	64.24	4131	25.18	1735	10.58
c7552	10088	60.98	4495	27.17	1959	11.84

V. 결 론

본 연구의 결과로 DIATEST와 본 연구에서 새롭게 제안하는 알고리즘은 대부분의 회로에 대해서 99%가 넘는 고장짝 진단율을 가지므로 조합회로에 대해서 성공적으로 동작하는 고장 진단 검사신호 생성 알고리즘임이 검증되었다. 새로운 고장진단 알고리즘은 전체 수행시간과 1개 고장짝에 대한 수행시간에 있어서 DIATEST보다 느렸다. 이러한 결과의 원인은 DIATEST의 검사신호 생성 시기를 분석한 결과 DIATEST는 2번의 고장 검출 검사신호 자동 생성 후 90%에 달하는 고장짝들이 구분되기 때문이라는 결론을 얻게 되었다. 그렇지만 c7552와 같이 출력단의 수가 많고 회로의 깊이가 깊지 않은 회로의 경우에는 본 알고리즘이 속도 면에서 더 빠른 결과를 보인다는 점은 특기할만한 점이었다.

이러한 속도 면의 단점에 비하여 검사신호의 수를 살펴보았을 때 새롭게 제안된 알고리즘(NEON)에 필요한 검사신호의 수가 가중 평균을 구하였을 때 1개로서 DIATEST가 2.109861개에 비하여 검사 신호 길이가 짧다는 이점을 가지고 있다는 사실을 알 수 있다. 이는 검사 길이(test length)를 단축시킴으로써 전체 테스트 과정에 필요한 비용을 절감할 수 있는 결과를 낳을 수 있다. 본 연구의 알고리즘에서 가장 특기할만한 점은 c1908이후의 비교적 많은 출력을 가지고 있는 회로에 대해서 80%에 가깝게 이르는 수치를 기록하는 완전 구분율이다. 이와 같이 높은 수치의 완전 구분율은 고장 진단 검사신호 생성의 목표인 고장 사전과 고장 진단 판단 트리 제작에 있어서 더욱 많은 정보를 주고 있다는 점에서 중요한 의미를 가지고 있다고 할 수 있다. 즉, 추가적인 정보를 얻기 위한 테스트 비용의 절감을 가져올 수 있다는 장점을 가져온다. 특히 현재 VLSI 칩들이 점점 더 더욱 많은 출력을 가지고 있다는 점을 생각할 때 이 수치는 더욱 더 높아질 것으로 기대가 되며, 그만큼 더 많은 정보를 줄 수 있을 것으로 기대된다.

이상과 같은 결론 하에서 앞으로 본 연구의 알고리즘에서 얻어진 완전 구분된 고장짝의 정보를 이용하여 보다 효율적인 고장사전의 제작 및 전체 고장 진단 시스템의 구성과 수행속도를 증진시킬 수 있는 방법들이 모색되어야 할 것이다.

참 고 문 헌

[1] Miron Abramovici, Melvin A. Breuer, Arthur D. Friedman., "*Digital System Testing and Testable Design*", IEEE Press, 1990.

[2] P. Camurati, D. Medina, P. Prinetto, and M. Sonza Reorda, "A diagnostic test pattern genera-

tion algorithm," *Proceedings of International Test Conference*, pp. 52-58, 1990.

[3] Torsten Gruning, Udo Mahlstedt, and Hartmut Koopmeiners, "DIATEST: A Fast Diagnostic Test Pattern Generator for Combinational Circuits," *Proceedings of the International Conference on Computer-Aided Design*, pp. 194-197, November, 1991.

[4] Irith Pomeranz and Sudhakar M. Reddy, "On the Generation of Small Dictionary for Fault Location", *Proceedings of the International Conference on Computer-Aided Design*, pp. 272-279, Nov. 1992.

[5] Vamsi Boppana, W. Kent Fuchs, "Fault Dictionary Compaction by Output Sequence Removal", *Proceedings of the International Conference on Computer-Aided Design*, pp. 576-579, 1994.

[6] R.E. Tulloss, "Fault Dictionary Compression: Recognizing When a Fault May Be Unambiguously Represented by a Single Failure Detection," *Digest of Papers 1980 Test Conference*, pp. 368-370, Nov., 1980.

[7] Goel, P., "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic", *IEEE Transaction on Computer*, Volume C-30, Number 3, pp. 215-222, 1981.

[8] Udo, M., Torsten, G., Cengiz, O., Wilfried, D., "CONTEST: A Past ATPG Tool for Very Large Combinational Circuits", *Proceedings of the International Conference on Computer Aided Design*, pp. 194-197, November, 1990.

[9] Fujiwara, H., Shimono, T., "On the Acceleration of Test Generation Algorithms", *Proceedings of the International Symposium on Fault Tolerant Computing*, pp. 98-105, 1983.

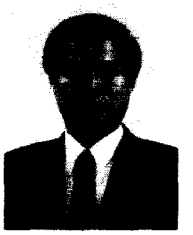
[10] Schulz, M.H., Auth, E., "SOCRATES:A highly efficient automatic test pattern generation system", *IEEE Transaction Computer-Aided Design*, volume 7, pp. 126-137, Jan. 1988.

[11] Schulz, M.H., Auth, E., "Advanced Automatic Test Pattern Generation and Redundancy Identification Techniques", Proceedings of 18th International Symposium on Fault Tolerant Computing Society, pp.30-35, 1988.

저 자 소 개

朴 英 鎬(正會員)

1956년 8월 10일생 1985년 대전산업대학교 전자계산학과 졸업 1983년~현재 한국전자통신연구원 시스템종합팀 선임기술원 주관심분야는 CAD, 컴퓨터네트워크

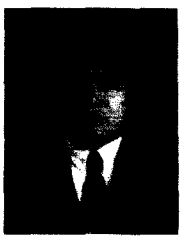


閔 炯 福(正會員)

1980년 2월 서울대학교 공과대학 전자공학과(공학사) 1982년 2월 한국과학기술원 전기 및 전자공학과(공학석사) 1990년 12월 The University of Texas at Austin 전기 및 컴퓨터 공학과 공학박사 1982년 3월~1985년 4월 금성통신(주)연구소 주임연구원 1985년 8월~1986년 7월 미국 Columbia 대학교 연구원 1991년 3월~현재 성균관대학교 전기공학과 부교수 주관심분야는 피냐 Testing, CAD 시스템

李 宰 勳(正會員) 第 35卷 C編 第 4號 參照

1991년 성균관대학교 공과대학 전기공학과 졸업(공학사) 1993년 성균관대학교 공과대학 전기공학과 졸업(공학석사) 1993년~1995년 LG전자기술원 주임연구원 1996년~1998년 성균관대학교 전기공학과 박사수료 1999년 3월~현재 여주대학 전자과 전임강사 주관심분야는 VLSI CAD / Testing, Low power design



申 容 煥(正會員)

1973년 7월 서울태생 1996년 2월 성균관대학교 공과대학 전기공학과(학사) 1996년 1월~1997년 2월 엘렉스테크(주) 1999년 2월 성균관대학교 대학원 전기전자 및 컴퓨터공학과(석사) 1999년 3월~현재 삼성전자(주)