論文96-33A-2-23

테스트 용이도를 이용한 조합 회로의 효율적인 로보스트 경로 지연 고장 테스트 생성

(Efficient Robust Path Delay Fault Test Generation for Combinational Circuits Using the Testability Measure)

許 龍 珉 *, 林 寅 七 *

(Yong-Min Hur and In-Chil Lim)

요 약

본 논문에서는 조합 논리 회로의 경로 지연 고장을 검출하기 위한 효율적인 로보스트 테스트 생성 알고리듬을 제안한다. 제안하는 로보스트 테스트 생성 방식은 회로 내의 각 게이트들에 대한 테스트 용이도를 계산하고 이 수치를 이용하여 로보스트 경로 지연 고장을 검출하기 위한 가중화된 임의 지연 테스트 벡터를 생성한다. 생성된 로보스트 테스트 벡터를 ISCAS '85 벤치마크 회로에 인가하여 병렬 패턴 기법을 이용한 고장 시뮬레이션을 수행한다. 실험결과 제안하는 테스트 생성 방식은 로보스트 경로 지연 고장 검출수를 증가시키고, 또한 로보스트 테스트 생성 시간을 단축시킬 수 있다.

Abstract

In this paper we propose an efficient robust path delay fault test generation algorithm for detection of path delay faluts in combinational logic circuits. In the proposed robust test generation approach, the testability measure is computed for all gates in the circuit under test and these computed values are used to generate weighted random delay test vectors for detection of path delay faults. For generated robust test vectors, we perform fault simulation on ISCAS '85 benchmark circuits using parallel pattern techniques. The results indicate that the proposed test generation method not only increases the number of detected robust path delay faults but also reduces the time taken to generate robust tests.

I. 서 론

반도체 기술의 발달로 수십-수백만개이상의 트랜지스터들을 단일 칩에 집적시킬 수 있게 됨으로써 속도와 기능이 크게 향상되어 보다 뛰어난 회로 설계가 가능하게 되었다. 이러한 대규모 집적회로가 실용화됨에 따라 광범위한 각종 응용 분야에서 사용되는 디지탈 시스템의 구성과 설계가 용이하여 그 사용이 급증하고 있다. 그러나 대규모 집적회로의 경우에는 높은 회로 복잡도로 인하여 회로가 정상적으로 동작되는지의 여부를 검증하는 테스트 과정이 매우 어렵고 많은 시간을 필요로 한다.

논리회로의 크기와 복잡도가 증가하면서 보다 안정되고 고속의 동작 속도를 보장하는 회로 테스트는 중

* 正會員, 漢陽大學校 工科大學校 電子工學科

(Dept. of Elec. Eng., Hanyang Univ.)

接受日字:1995年11月25日, 수정완료일:1996年1月29日

요한 문제로 대두되고 있는데 집적회로의 정확한 동작을 위해서는 정적(steady)동작 뿐만 아니라 동적(dynamic)동작을 검증하는 지연 고장 테스트 방식이 필요하다.

지연 고장을 일으키는 원인으로는 제조 공정 상의 도핑(doping), 확산(diffusion)단계에서의 파라메타 변동, 물리적 결함과 통계적 타이밍 분석(statistical timing analysis)에 의한 회로 설계 방식에도 기인하며 이러한 지연 고장의 존재는 시스템의 성능과 신뢰도를 저하시키는 요인이 된다.[1][2]

지연 고장의 모델링 방법으로는 두가지가 있는데, 게이트 지연 고장(gate delay fault)모델과 경로 지연 고장(path delay fault)모델로 구분할 수 있다.[3][4][5][6] 게이트 지연 고장은 대상회로 내의 한 개의 게이트가 규정된 최악(worst-case)지연 값을 초과하였을 경우를 말하며, 이때의 지연 고장은 게이트의 입력 또는 출력선에 집중되어 발생했다고 가정한다. 경로 지연 고장은 게이트 지연 고장과는 달리 주입력선(또는 래치의 출력)에서 주출력선(래치의 입력)까지의 한 경로 상에서 작은 지연의 누적된 고장 효과를 모델링 할 수 있을 뿐만 아니라 대상 경로를 통한 전파 지연이 시스템의 클럭 주기안에 있는가만을 평가한다. 게이트 지연 고장의 경우 회로 전체 또는 활성화된 경로 전반에 걸쳐 지연 고장이 발생할 수 있음에도 불구하고 단지 한 개의 단일 게이트에서 고장이 일어난다고 가정하므로써 분산되어 있는 작은 지연 고장의 효과를 모델링 하지 못하는 단점을 가지고 있다. 또한 규정된 지연을 초과하는 고장이 경로를 전파하면서 빠른 게이트(fast gate) 나 짧은 경로(short path)로 인해 보상되어 출력 관측시 시스템의 클럭 주기안에 있게 되는 경우가 있을 수 있으며, 고장이 발생된 신호선의 지연 고장 크기가 대상회로의 시스템 클럭주기 를 벗어날 정도로 매우 크다고 가정하는 모델인 천이 고장(transition fault or gross delay fault)이 아닐 경우 각 게이트의 신호선의 지연 고장으로 인한 출력측에서의 검출 임계값(detection thresholds) 계산이 복잡하게 된다.[7][15][16] 이 검출 임계값은 어느정도의 크기 이상으로 고장이 발생되어야 출력측에서 그 고장 효과가 관측가능한지를 알 수 있는데 그때의 고장 크기를 말하며, 각 신호선마다 서로다른 값을 갖는다. 반면에 경로 지연 고장의 경우 이러한 게이트 지연 고장을 포함하는 포괄적인 모델링 방법이지만 회로의 크기가 증가함에 따라 회로 내에 존재하는 구조적(structual), 기능적(functional) 경로 수가 크게 증가하므로 이에 대한 적절한 대책이 요구되고 있다.[8][9]

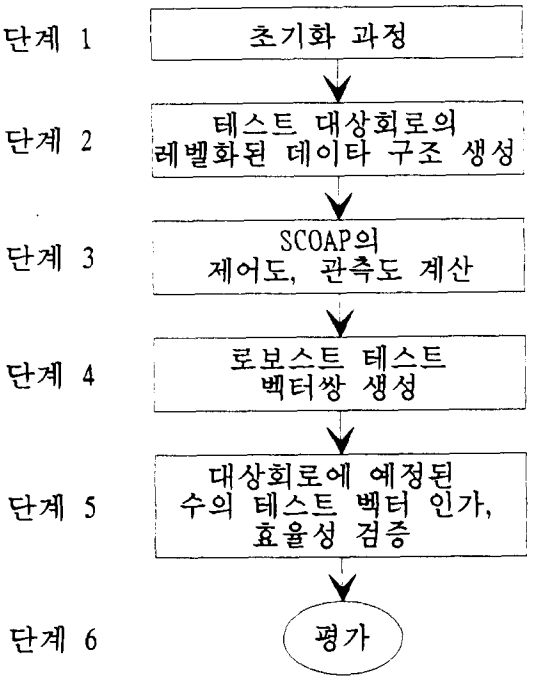

그림 1. 경로 지연 고장 테스트 벡터 생성 과정의 흐름도

Fig. 1. Flowchart of the path delay fault test vector generation process.

경로 지연 고장을 검출하기 위한 테스트 생성 방식으로는 기존의 stuck-at 테스트에서 사용된 결정론적인(deterministic)방식과 임의 테스트(random test) 방식이 모두 적용 가능하다. 기존의 결정론적인 테스트 방식에 기초한 연구로서 PODEM방식에 근거한 Lin & Reddy[10]와 Fuch[11], Niermann[12] 등의 연구가 있으며, 다중 경로 전파[13]에 의한 방법, 또한 에너지 함수[14]에 의한 방식등이 제안되어 왔다. 그러나 이들 논문의 결정론적인 방식을 사용하였을 경우는 대상회로의 크기가 증가하면서 고려하여야 할 대상 경로 고장 수가 크게 증가하여 많은 테스트 생성 시간이 필요하게 되므로 테스트 대상 경로수를 제한하여 테스트 벡터를 생성하게 한다. 또한 임의 테스트 방식을 사용하였을 경우는 테스트 생성 시간은 줄어들지만 테스트 벡터의 질(quality)은 떨어지게 되므로 역시 원하

는 고장 검출율을 얻기 위해서는 많은 수의 임의 테스트 벡터가 인가되어야만 한다.

따라서 본 논문에서는 대상회로의 테스트 용이도의 정보를 활용하여 적은 시간내에 대상회로의 많은 로보스트 경로 지연 고장들을 검출할 수 있는 테스트 벡터를 생성시키도록 한다. 또한, 생성된 테스트 벡터의 평가를 위해 병렬 패턴 경로 지연 고장 시뮬레이션을 수행하여 테스트 벡터의 성능을 확인한다. 그림 1은 제안된 경로 지연 고장 테스트 벡터 생성과정의 흐름도로 IV장에서 설명한다.

Ⅱ. 테스트 용이도와 지연 고장 테스트

1. 지연 고장 테스트 하드웨어 모델

기존의 stuck-at 고장 모델은 단일 패턴을 대상회로에 인가하여 시스템 클럭주기에 상관없이 주출력의 신호 값이 안정하게 되면 이를 관측하여 정상상태(fault-free)의 출력값과 비교하여 고장의 유무를 판단하는 비교적 간단한 테스트 방식이다. 그러나 회로의 기능적인 면만을 검증하는 관계로 시간과 관련된 회로의 타이밍 고장은 검출할 수 없는 단점을 가지고 있다. 그러므로 회로의 타이밍과 관련된 지연 고장을 검출하기 위해서는 천이 신호를 발생시켜 그 지연 고장 효과를 주출력까지 전파시키는 두 개의 서로 다른 테스트 패턴이 필요하다.

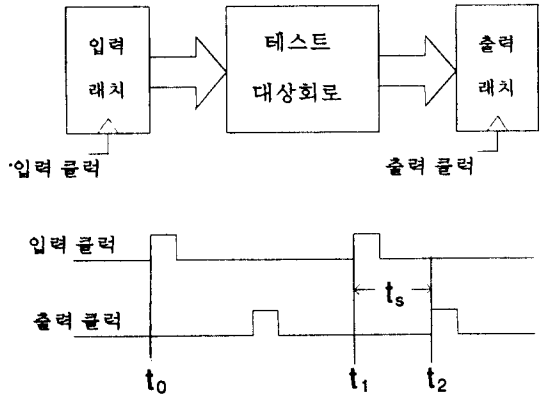

그림 2. 지연 고장 테스트인가를 위한 하드웨어 모델

Fig. 2. Hardware model to apply delay fault test.

지연 고장 모델에는 앞절에서 언급한바와 같이 게이트 지연 고장 모델과 경로 지연 고장 모델로 구분된다. 게이트 지연 고장 모델은 회로 내의 고장이 발생한 게이트 입출력 신호선에 상승(slow-to-rise) 및 하강(slow-to-fall) 지연 고장을 모델링하며, 경로 지연 고장 모델에서는 주입력에서 주출력까지의 경로를 대상으로 역시 상승 및 하강 지연을 발생시킨다. 따라서 주입력에서 천이를 발생시켜야 한다. 이러한 천이를 주출력까지 전파시키기 위해서는 회로의 초기화 벡터(initialization vector) V1 과 전파 벡터(propagation vector) V2의 한 쌍의 벡터가 필요하다. 따라서 이 한 쌍의 벡터가 순서적으로 인가되어야 하며 그림 2에서 나타난바와 같이 시간 t_0에서 V1벡터가 인가되어 안정하게 된후 시간 t_1에서 V2벡터를 인가한다. 그리고 회로의 시스템 클럭주기 $t_s(t_2 - t_1)$에서 출력을 샘플링한다.

〈정의 1〉 활성화된 경로 P상에 존재하는 모든 게이트의 경로외 입력 (off-path)의 최종값을 비제어값(non-controlling value)으로 할당하여 출력에서 검출 가능하면 이를 넌로보스트(nonrobust) 테스트라 하고 이 넌로보스트 테스트중에서 다중 고장 또는 경로외 입력의 지연에도 영향을 받지 않고 항상 고장 검출 가능하다면 이를 로보스트 테스트라 한다. 따라서 로보스트 테스트로부터 얻게 되는 고장 집합은 넌로보스트 테스트로 얻은 고장 집합의 부분집합이 된다.

정의 1에서의 제어값이란 입력단에서 출력값을 반드시 제어할 수 있는 논리값을 말하는 것으로 AND게이트의 경우 0, OR게이트의 경우는 1이 된다. 비제어값의 경우는 AND게이트의 경우 1, OR게이트의 경우 0이다.

2. 테스트 용이도

대상회로의 구조로부터 정적으로(static) 계산되거나 또는 과거의 기록으로부터 동적으로(dynamic) 계산되는 테스트 용이도는 회로 내부의 각 신호선들이 외부 입출력선을 통하여 테스트하기가 어느 정도 용이한지를 수치로 표시한 것이다. 고장 신호 전파(propagation)과정에서 PO(Primary Output)까지 여러 개의 전파 경로가 존재하여 그 중 하나의 경로를 선택해야 할 경우나, 일치(justification)과정에서 게이트의 한 입력을 선택해야 할 경우에 각 신호선의 테스트 용이도를 참조하여 판단하게 된다.

디지탈 논리회로의 테스트 용이도를 계산하는 알고리듬은 대표적으로 SCOAP[17], CAMELOT[18],COP[19],VICTOR[27], 등 여러가지가 제안되었으며, 테스트 용이도는 제어도(controllability)와 관측도(observability)로 구성된다.

제어도는 대상회로의 내부 신호선의 값을 PI (Primary Input)에 의해 용이하게 제어할 수 있는 정도를 나타내고, 관측도는 내부 신호선의 값을 PO에서 용이하게 관측할 수 있는 정도를 나타내는 수치이다. 일반적으로 테스트 용이도는 PI에서 PO측으로 제어도를 먼저 계산하고 다시 PO에서 PI측으로 관측도를 계산하는 과정으로 이뤄진다. 본 논문에서 사용하는 테스트 용이도는 SCOAP을 이용하며, 이 SCOAP은 제어도와 관측도를 비용 개념으로 생각하여 신호선의 0에 대한 제어도와 1에 대한 제어도를 구분하여 사용한다. 한 게이트 출력을 제어하기 위한 비용은 게이트 입력을 제어하기 위한 비용으로 계산된다. 각 기본 게이트 논리에 따른 제어도와 관측도의 계산식은 다음과 같다.

표 1. SCOAP의 제어도와 관측도
Table 1. Controllability and observability of SCOAP.

게이트형태	제어도	관측도
AND	$CY0(z) = \underset{i=1}{\overset{n}{MIN}}\, CY0(x_i) + 1$ $CY1(z) = \sum_{i=1}^{n} CY1(x_i) + 1$	$OY(x_i) = OY(z) + \sum_{i=1,\, i\neq j}^{n} CY1(x_i) + 1$
OR	$CY0(z) = \sum_{i=1}^{n} CY0(x_i) + 1$ $CY1(z) = \underset{i=1}{\overset{n}{MIN}}\, CY1(x_i) + 1$	$OY(x_j) = OY(z) + \sum_{i=1,\, i\neq j}^{n} CY0(x_i) + 1$
NAND	$CY0(z) = \sum_{i=1}^{n} CY1(x_i) + 1$ $CY1(z) = \underset{i=1}{\overset{n}{MIN}}\, CY0(x_i) + 1$	$OY(x_j) = OY(z) + \sum_{i=1,\, i\neq j}^{n} CY1(x_i) + 1$
NOR	$CY0(z) = \underset{i=1}{\overset{n}{MIN}}\, CY1(x_i) + 1$ $CY1(z) = \sum_{i=1}^{n} CY0(x_i) + 1$	$OY(x_j) = OY(z) + \sum_{i=1,\, i\neq j}^{n} CY0(x_i) + 1$
XOR	$CY0(z) = MIN(s0) + 1$ $CY1(z) = MIN(s1) + 1$	$OY(x_j) = OY(z) + \sum_{i=1,\, i\neq j}^{n} MIN(CY0(x_i), CY1(x_i)) + 1$
INV	$CY0(z) = CY1(x) + 1$ $CY1(z) = CY0(x) + 1$	$OY(x) = OY(z)$
BUF	$CY0(z) = CY0(x) + 1$ $CY1(z) = CY1(x) + 1$	$OY(x) = OY(z)$

초기화 : 제어도 : CY0(PI) = CY1(PO) = 1, 관측도 : OY(PO) = 0

표1에서 x_i 는 입력을 나타내고, z는 출력, n은 입력신호선 수, XOR게이트의 s0, s1은 각각 출력을 0과 1로 만드는 입력 집합이다.

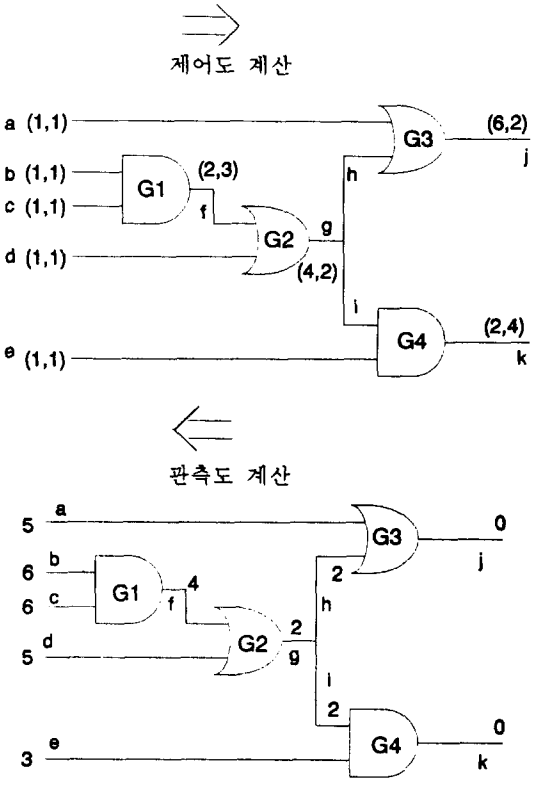

그림 3. 제어도와 관측도의 계산
Fig. 3. Computation of controllability and observability.

위의 공식에 의해 PI의 제어도 값이 CY0,CY1 = (1,1)에서 시작하여 PO측으로 계산을 수행하기 때문에 제어도 값이 커질 수록 신호선에 대한 제어가 어려워진다. 관측도는 PO의 값이 0에서 시작하여 이미 계산된 제어도와 위의 관측도 수식에 따라 계산되며 그 값이 0에서 부터 커질수록 신호선에 대한 관측이 어려워진다. 2-입력(x1,x2) XOR의 경우 출력을 1로 하는 입력 집합 s1 = {10, 01} 이므로 CY1(z) = MIN (CY1(x1)+CY0(x2), CY1(x2)+CY0(x1)) 이다.

또한 팬아웃 지점에서의 관측도 계산은 팬아웃가지중 가장 낮은 값을 취한다.

다음은 SCOAP을 이용하여 고장을 검출하기 위한 제어도와 관측도의 수치를 이용한 예이다.

그림 3은 표1을 사용하여 제어도와 관측도순으로 계산한 예이며, 그림 4는 s-a-0 고장이 발생하였을 경우 d 신호선의 값의 1제어도가 1이고 f 신호선의 1제어도가 3이므로 g 신호선의 논리 1을 만들기에 제어값이 낮은 d신호선이 용이하다는 것을 알 수 있다. 또한 이 고장 효과를 주출력에 전파시키기 위해서 관측도값을 참조하는데 이경우는 h, i의 관측도가 동일하므로 j, k 출력신호선에 고장 효과를 전파시키는 용이함은 두 경로가 같다는 것을 알 수 있다.

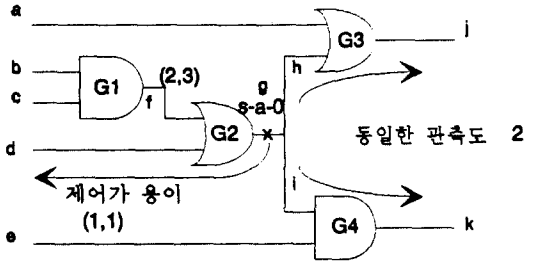

그림 4. SCOAP테스트 용이도 사용 예
Fig. 4. An example of SCOAP testability measure.

Ⅲ. 테스트 용이도를 이용한 경로 지연 고장 테스트 생성

경로 지연 고장 수는 회로의 크기가 증가함에 따라 지수 함수적으로 증가한다. 따라서, 이러한 모든 경로 지연 고장을 검출하기 위한 테스트 벡터쌍의 생성은 지연 테스트를 고려한 회로 설계(delay testable design)가 아니면 매우 많은 시간을 필요로 한다. 또한, 전체 경로 지연 고장 중에서 많은 수의 경로 지연 고장이 구조적으로는 존재하나 기능적으로 활성화될 수 없는 경로들이다.[21] 따라서 적은 시간내에 많은 수의 경로 지연 고장을 검출하고 다중 고장에 영향을 받지 않는 로보스트 테스트 생성이 특히 중요하다.

1. 초기화 벡터 생성

임의 테스트 방식을 이용한 stuck-at 테스트는 간단한 하드웨어를 이용하여 테스트 벡터를 생성할 수 있다. 그러나 이를 두 개의 테스트 벡터가 필요한 stuck-open 고장 및 지연 고장 테스트에 적용하였을 경우에는 대상회로의 특성에 상관없이 천이를 회로 전체의 입력에 임의적으로 분포시켜 줌으로써 효율적인 로보스트 테스트 생성 방식이 되지 못한다. 제안된 경로 지연 고장의 로보스트 테스트 방식은 주입력측의 천이가 주출력까지 로보스트하게 도달 가능하도록 하는 테스트 벡터의 확률 값을 대상회로의 관측도 수치를 이용하여 구하도록 한다. 이것은 지연 고장 전파시 게이트의 경로외 입력의 비제어값이 높은 확률로 인가된다면 해당 고장이 주출력쪽으로 전파될 가능성이 높아지게 된다. 예를 들어 2-입력 AND게이트의 상승 지연 고장을 출력측으로 전파시키기 위해선 고장이 전파될 경로상의 입력에 초기화 벡터와 전파 벡터를 각각 0과 1을 인가하여야 한다. 그리고 다른 한쪽의 입력벡터쌍으로는 상승이 유발되는 시점에서 비제어값인 논리1이 인가된다면 고장이 없는 경우 출력측에 최종적으로 논리 1값이 관측되고 고장이 발생하여 천이가 제대로 발생되지 못한 경우라면 출력측에 논리 0값이 관측될 것이다.

SCOAP의 관측도 계산은 해당 신호선의 논리값을 출력쪽으로 전파시키기 위해 해당 게이트의 다른 입력 신호선들의 비제어값의 비용으로 계산되어진다. 그러므로 주출력에서 시작하여 주입력으로 다음과 같은 식에 의하여 계산하고 최종적으로 구한 주입력의 논리0과 1의 계산 값을 각각 P^0, P^1이라면 $P^1/(P^0 + P^1)$이 지연 테스트 초기화 테스트 벡터의 인가 확률 값이다.

다음의 식들은 각 게이트들의 초기화 테스트 벡터 확률 계산 식이다.

$$\text{AND}: \quad (P^0, P^1)_{i+1} \rightarrow (P^0, P^1 \frac{O_i}{O_{i+1}})_i \tag{1}$$

$$\text{OR}: \quad (P^0, P^1)_{i+1} \rightarrow (P^0 \frac{O_i}{O_{i+1}}, P^1)_i \tag{2}$$

$$\text{NAND}: \quad (P^0, P^1)_{i+1} \rightarrow (P^1, P^0 \frac{O_i}{O_{i+1}})_i \tag{3}$$

$$\text{NOR}: \quad (P^0, P^1)_{i+1} \rightarrow (P^1 \frac{O_i}{O_{i+1}}, P^0)_i \tag{4}$$

$$\text{XOR}: \quad (P^0, P^1)_{i+1} \rightarrow (P^0 \frac{O_i}{O_{i+1}}, P^1 \frac{O_i}{O_{i+1}})_i \tag{5}$$

$$\text{INV}: \quad (P^0, P^1)_{i+1} \rightarrow (P^1, P^0)_i \tag{6}$$

$$\text{BUF}: \quad (P^0, P^1)_{i+1} \rightarrow (P^0, P^1)_i \tag{7}$$

위 식에서 P_i^0 는 i(1<= i <= n ,1:주입력, n:주출력)레벨에서의 0의 확률 전파 값이고, P_i^1 는 i레벨에서의 1의 확률 전파 값이다. O_i 는 i레벨에서의 관측도를 의미한다. 따라서 주출력에서 주입력으로 진행하면서 각 게이트들의 해당하는 수식으로 각각 0과 1의 논리 신호전파값을 계산하고, 최종적으로 구한 주입력의 P_i^0 와 P_i^1로 인가확률을 구한다. 단, 주출력의 관측도는 1로 계산하고 팬아웃 지점의 계산은 팬아웃가지에 할당된 값중에서 가장 큰 값을 취하도록 한다. 주출력 n에 대해선 $(P^0, P^1)_n$ = (1/2, 1/2)로 초기화 한다.

테스트 용이도중에서 주출력측에서 바라보는 관측도를 각 게이트의 비제어확률값을 생성하는 인자로 사용하였다. 또한, 위 예제에서 보는 바와 같이 SCOAP의 관측도는 주출력에서 멀어질수록 또는 주입력에 인가되는 연결 신호선의 수가 증가할 수 록 관측도가 떨어지므로 상대적으로 관측도의 수치가 작은 주입력선의 신호선에 높은 비제어값의 확률값이 인가된다면 경로 길이가 긴 고장들에 대해서 로보스트하게 경로 지연 고장을 검출할 수 있는 확률이 증가하게 된다.

확률 값의 계산은 대상회로를 레벨화한 순서로 한 번의 과정으로 계산이 가능하며 테스트 생성 시간은 전체 시뮬레이션에 비하면 아주 작은 부분을 차지하게 된다.

2. 전파 벡터의 생성

전파 벡터는 임의 테스트 생성기에서 발생되는 출력을 그대로 사용한다. 임의 테스트 발생기는 대상회로안에 자체 테스트 발생기를 내장하였을 경우 보통 LFSR(Linear Feedback Shift Register)를 사용하는데 n 개의 출력 단자가 있다면 출력 단자에 모두 0이 오는 경우를 제외한 총 2^n - 1의 테스트 벡터를 생성할 수 있다. 본 논문에서 제안한 방식은 BIST (Built-In Self-Test)방식에 적용 가능하며 초기화 벡터의 생성을 위해 부가적인 가중화 하드웨어가 필요하다. 따라서, 2^{n-1} 테스트 벡터를 인가할 수 있는 LFSR을 입력원으로 이용하였을 경우 0.5의 확률을 갖는 논리 1값이 한 번은 대상회로의 주입력단자에 인가되고 그 다음은 부가 하드웨어를 거쳐 수정된 확률값을 갖는 논리값이 다음 테스트 클럭에서 주입력에 교대로 계속 인가된다. 부가 하드웨어로 2 입력 AND 게이트를 LFSR 출력단 2개에 연결하였을 경우 AND 출력단의 1 확률 값은 1/4, 0은 3/4이 된다. 이는 부가 하드웨어 AND 입력단에 균일한 0.5의 논리 1값을 발생시키는 하드웨어가 있다고 가정하였기 때문이다. 전파 벡터의 확률 값을 0.5로 인가시킨 이유는 동일하게 초기화 벡터의 확률을 인가하게 되면 오히려 상승 및 하강의 천이 기회가 줄어 들며, 그 반대로 (1 - 초기화 벡터 확률)을 인가하게 되면 주입력 모두에서 같은 형태의 천이가 지속적으로 발생되기 때문에 효과적이지 못하다. 따라서 부가 하드웨어적인 면과 벤치 마크 회로를 통한 실험에서 알아본 바와 같이 어느 특정한 경로만이 아닌 전체 경로들의 고른 천이 기회를 부여하기 위해서이다.

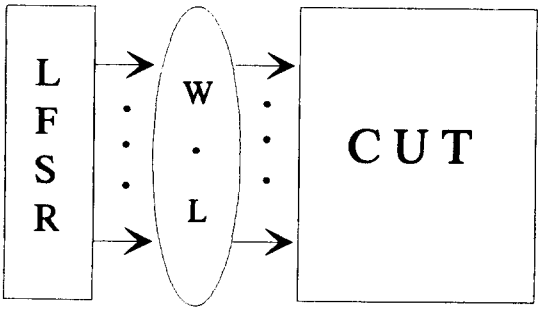

그림 5. LFSR과 가중화 논리를 사용한 테스트 인가.

Fig. 5. Test application using LFSR and weighted logic.

3. 경로 지연 고장 테스트 생성 과정의 예

그림 3의 예제회로에서 주출력 j,k 에 초기값 $(P^0,P^1)=(1/2,1/2)$을 할당하고 G3의 게이트 형태가 OR이므로 a 신호선의 (P^0,P^1)의 계산은 식 (2)에 따라 P^0 = j신호선의 $P^0(1/2)$ * (a 신호선의 관측도 5 / j 신호선의 관측도 1) = 5/2 로 계산되고 P^1은 j 신호선의 $P^1(1/2)$ 이 된다. 또한 h신호선의 경우도 OR게이트의 입력 신호선이므로 같은 방법으로 P^0 = j 신호선의 P^0 (1/2) * (h 신호선의 관측도 2 / j 신호선의 관측도 1) = 1, P^1 = j 신호선의 $P^1(1/2)$ 로 된다. 따라서 G4의 입력선에 대해서도 식 (1)과 같은 방법으로 계산되어지며, g신호선의 팬어웃지점에 대해서는 앞서 정의 한대로 팬아웃가지의 (P^0,P^1)값들 중에서 가장 큰 값들로 결정한다. 이와같이 나머지 게이트들에 대해서도 주출력에서 주입력으로 진행하면서 식(1)-(7)에 기초하여 계산한다.

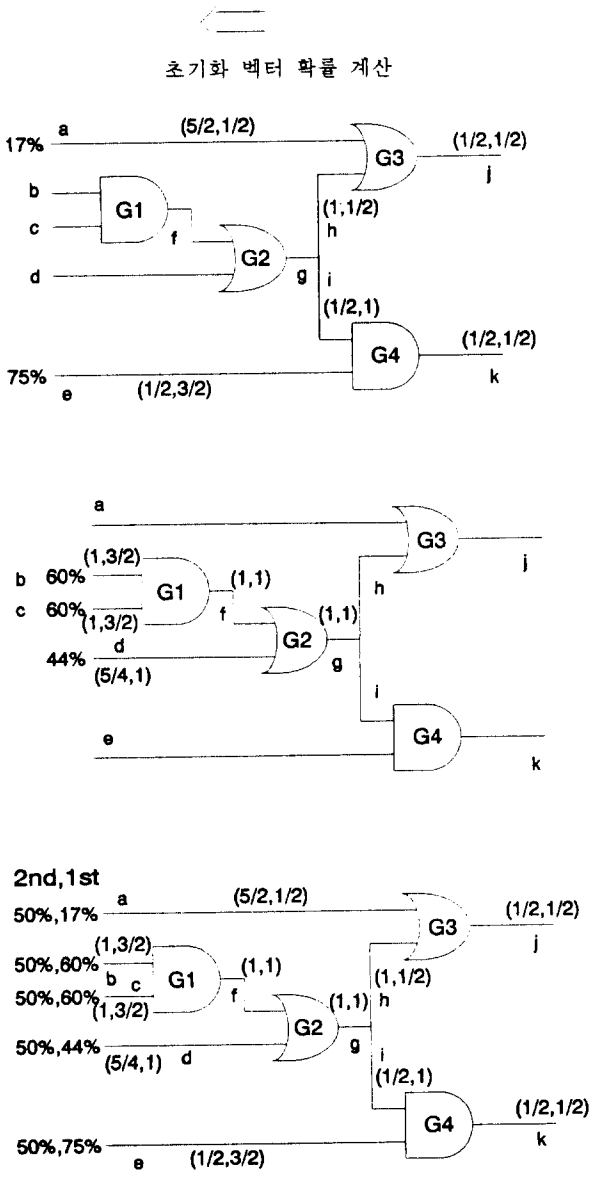

그림 6. 그림 3에 대한 테스트 생성 예
Fig 6. An test generation example of Fig.3.

주입력선 a,b,c,d,e의 (P^0,P^1)의 값이 각각 구해지면 최종적으로 $P^1/(P^1+P^0)*100(\%)$ 로 17%,60%, 60%,44%,75% 의 논리1값에 대한 초기화 벡터 인가 확률을 구할 수 있다. 다음으로 두번째 전파 벡터는 LFSR의 생성기에서 출력으로 나오는 0.5의 논리 1값에 대한 확률을 인가하면 된다. 예제에서 알 수 있듯이 주출력 j,k에서 상대적으로 가까운 신호선인 a,e에 대해서는 b,c,d의 비제어값보다 높게 인가되므로 상대적으로 보다 긴 경로들에 대한 로보스트 고장 검출의 가능성을 높여준다. 즉, b,c,d ->f,g->h,i->j,k로 이어지는 경로 지연 고장의 상승 및 하강천이의 고장을 로보스트하게 검출할 수 있다. (표 5참조)

Ⅳ. 경로 지연 고장 테스트 벡터 생성 절차

제안된 그림 1의 절차에 의해 경로 지연 고장의 테스트 벡터를 생성하고 , 생성된 테스트 벡터쌍에 고장 시뮬레이션을 수행하여 그 효율성을 입증한다. 그림 1에서 보는바와 같이 테스트 생성과 그 검증이 크게 6단계로 나누어 각 단계에서 수행되어지는 절차를 보면 다음과 같다.

(단계 1)

초기화 과정으로 우선 대상회로를 결정하고(본 논문의 실험에서는 ISCAS'85벤치마크[28] 회로를 대상으로 함), 대상회로에 인가되는 테스트 벡터의 수를 결정한다. 또한, 인가되는 초기화벡터의 확률값을 정확하게 생성하기 위해선 많은 H/W(HardWare)가 소비되므로 일반적으로 다음과 같은 가중값 단계를 두어 시뮬레이션을 수행한다. 즉, 내정치(default)로 4-가중값 단계(0.2, 0.4, 0.6, 0.8)를 두고 사용자가 특별히 정할 수 있게 한다. 4-가중값 단계의 테스트 생성인 경우 주입력의 어느 한 신호선의 비제어 신호선값의 확률값이 0.25라면 이에 가장 가까운 0.2를 선택하여 테스트 입력이 만들어 지게 한다.

그리고 32비트 LFSR을 사용하여 각 대상회로에 테스트 벡터를 인가하여 고장 시뮬레이션을 수행한다. 32 비트 LFSR 은 2^{32} - 1 개의 테스트 벡터를 생성할 수 있도록 하였으며 이렇게 함으로써 출력에서 0과 1의 확률이 1/2이 될 수 있어 그림 5 에서와 같이 가중화 논리를 통해서 인가되는 확률값이 의도한대로 인가될 수 있다. 실험에서는 $X^{32} + X^{28} + X^{27} + X^1 + 1$ 의 원시다항식(primitive polynomial) 함수를 사용하였다.

(단계 2)

대상회로를 레벨화된(levelized) 연결 구조로 데이타 구조를 작성하다.

(단계 3)

SCOAP의 제어도를 먼저 계산하고, 계산된 제어도 값을 이용하여 각 신호선의 관측도값을 계산한다.

(단계 4)

전 장에서 설명한 바와 같이 (1)에서 (7)까지의 식을 이용하여 각 신호선의 관측도의 값을 인자로 하여 주출력에서 주입력으로 진행하면서 주입력의 초기화 벡터의 확률값을 계산하고, 단계 1에서 정한 가중값 단계에 기초하여 설계된 논리를 통해 초기화 벡터가 생성된다. 그리고 임의 벡터기에서 생성된 출력 벡터를 전파 벡터로 사용한다.

(단계 5)

단계 1에서 정한 테스트 벡터의 수 만큼 테스트 대상회로에 인가되어 로보스트와 넌로보스트의 경로 지연 고장 검출 수를 확인하기 위해 고장 시뮬레이션을 수행한다.

(단계 6)

모든 테스트 벡터가 인가 되었다면 종료한다.

본 논문에서는 고장 시뮬레이션시 병렬 패턴 기법을 적용하여 경로 지연 고장 시뮬레이션을 수행한다. 병렬 패턴 처리 기법에 의한 시뮬레이션 기법은 컴퓨터의 n 비트 워드 길이에 해당하는 만큼의 테스트 벡터를 동시에 처리하는 것으로 이 병렬 패턴에 활성화될 수 있는 다수의 경로 지연 고장을 동시에 시뮬레이션 한다. 그리고 경로 지연 고장 시뮬레이션시 신호값으로는 일반적으로 가장 널리 사용되는 S0, S1, T0, T1, U0, U1으로 표시되는 6개의 논리값[6,22]을 기본으로 하여 시뮬레이션 한다.

S는 테스트 벡터가 인가되는 동안 일정하게 논리 값을 유지하는 신호선에 할당되며, T는 하강 천이나 상승 천이 신호를 로보스트하게 전파하는 신호선에, 그리고 U는 초기값이 0인지 또는1 인지는 알 수 없지만 최종값만 알 수 있으며 S 또는 T로 할당될 수 없는 신호선의 논리값을 표현한다.

본 논문에서 제안된 경로 지연 고장 테스트 시스템은 대상회로에서 시스템 클럭에 영향을 주는 최장 경로(longest path)뿐만 아니라 다양한 경로 길이에 대해서도 생성된 테스트 벡터의 효율성을 검증 할 수 있다. 즉 모든 게이트 소자가 동일한 상승 및 하강 지연 크기를 갖는다고 가정하고 각 게이트의 팬아웃 가지 수에 선형적으로 비례하여 지연 크기를 설정한다. 그리고 주입력에서 주출력으로 회로를 레벨화시켜 진행하면서 가장 긴 경로를 검출하여 그 경로의 크기를 1로 정규화(normalization)시킨다. 사용자가 원하는 경로 길이를 0과 1사이에서 지정할 경우는 지정된 경로 길이 이상 되는 모든 경로 고장에 대해서 고장 시뮬레이션을 수행하여 로보스트 및 넌로보스트의 고장 수를 검출한다.

따라서 사용자는 설계된 회로의 시스템 클럭을 결정하는 임계경로(critical path)에 대해 고장 시뮬레이션을 할 수 있을 뿐만 아니라 각 경로 길이에 따른 타이밍 분석도 할 수 있게 된다. 또한 정교한 ATE (Automatic Test Equipment)장비를 사용하여 테스트할 경우 시스템 클럭보다 작은 주기에서 관측이 가능하므로 설계된 회로의 타이밍 분석이 용이하다.

V. 실험 및 결과

제안된 경로 지연 고장 테스트 시스템은 ISCAS'85 벤치마크 회로를 대상으로 임의 테스트 기법과 기존의 경로 지연 고장 테스트 방법과 성능을 비교 평가하였으며, 입력원으로써 32비트의 원시 다항식 함수의 LFSR로서 생성된 패턴을 사용하였고 전파 패턴은 다음 패턴쌍의 초기화 패턴이 되도록 시뮬레이션 하였다. 본 논문의 실험 환경으로는 SUN SPARC-10 시스템을 사용하였다.

표 3은 제안된 테스트 생성 방식과 임의 테스트 패턴 생성 방식과의 실험 결과로서 4-가중값 단계에 기초한 테스트 벡터로 각 1000개와 10,000개의 입력 벡터를 인가하였을 경우의 고장 검출수를 나타낸다. c880의 경우 처음 1000개의 입력 벡터로 고장 시뮬레이션을 수행한 결과 임의 테스트의 경우 240개와 2692개의 로보스트 및 넌로보스트의 고장을 각각 검출할 수 있었고, 테스트 용이도를 이용하여 테스트 벡터를 발생하였을 경우는 380개와 1567개의 로보스트와 넌로보스트의 고장을 검출하였다. 또한, 입력벡터수를 10,000개로 증가하였을 경우에도 440(6028)에서 778(3413)의 로보스트 테스트 가능한 고장을 검출하였으며, c432 회로의 경우를 제외하고는 제안된 테스트의 벡터의 생성 방식이 로보스트 테스트 생성에 있어 보다 좋은 결과를 얻을 수 있었음을 보여주고 있다.

표 4는 10,000개의 입력 벡터를 인가하였을 때 고장 시뮬레이션의 수행 시간, 그리고 괄호안의 시간은 테스트벡터를 생성하기 위해 사용된 시간 즉, 제어도와 관측도계산 그리고 III장의 식 (1) - (7)을 이용해 초기화벡터 확률을 계산하는데 드는 시간을 나타낸다.

표 3. 경로 지연 고장 테스트 생성 결과
Table 3. Path delay fault test generation results.

Circuit	LFSR*		this paper*		LFSR		this paper	
	Robust	Nonrobust	Robust	Nonrobust	Robust	Nonrobust	Robust	Nonrobust
c432	109	4093	131	2556	302	8326	224	6672
c499	17	16840	30	14041	30	113879	154	73142
c880	240	2692	380	1567	440	6028	778	3413
c1908	161	7116	300	2938	625	37545	965	8092
c2670	912	19257	899	9192	1186	44989	1626	21159
c3540	285	87717	746	69685	567	294947	2014	238517
c5315	1796	37090	2044	24551	2988	149560	5036	103858
c7552	1126	70860	1477	50786	2105	14209	3575	133063

표 4. 전체 테스트 시간
Table 4. Total test time.

Circuit	Robust/Nonrobust		CPU time
c432	224	/6672	3.70s(0.05s)
c499	145	/73142	1.42s(0.03s)
c880	778	/3413	3.13s(0.12s)
c1908	965	/8092	4.63s(0.3s)
c2670	1626	/21159	10.32s(0.37s)
c3540	2014	/238517	44.45s(0.46s)
c5315	5036	/103858	31.42s(0.75s)
c7552	3575	/133063	47.63s(1.22s)

표 5는 c880의 회로를 대상으로 10만개의 벡터를 인가하여 각 경로 길이 크기에 따라 실험한 결과이다. 로보스트 테스트 생성수에 있어서 순수 임의 테스트 벡터로 실험하여 얻은 결과보다 경로 길이가 큰 고장에 대해서 보다 효과적인 것으로 나타났다. 경로 길이 0.9에서 1.0사이의 로보스트고장 수를 보면 제안된 방식에 의해서는 4개이며 임의 테스트의 경우에는 한 개도 검출되지 않은 것을 알 수 있다.

전체적인 실험 결과 로보스트 테스트 생성수가 크게 향상된 것을 알 수 있었으며, 상대적으로 넌로보스트의 테스트수가 감소된 것은 로보스트한 전파를 위해 고정된 확률로 지연 테스트의 초기화 입력 벡터를 인가한 결과라 할 수 있다. 표 5의 실험에서와 같이 제안된 테스트 방식의 로보스트 전파 가능한 1326개의 검출수를 얻기 위해서는 임의 테스트 벡터로 50만개 이상의 벡터를 인가하여도 1000개 이상의 로보스트 전파 고장을 검출할 수 없었다.

표 5. 경로 길이에 따른 경로 지연 고장 분포
Table 5. Path delay fault distribution to the path length.

Size of path length	LFSR		this paper	
	Robust	Nonrobust	Robust	Nonrobust
0.9-1.0	0	476	4	157
0.7-0.9	25	3923	205	2247
0.5-0.7	239	4104	590	2656
0.3-0.5	240	1233	329	962
0.1-0.3	179	244	202	253

표 6은 5-가중값(0.03,0.25,0.5,0.75,0.97) 단계에 기초하여 생성된 테스트벡터로 테스트한 결과로 PI는 주입력선의 수이고 Gate는 총 게이트수이다.

위 표에서 알 수 있듯이 제안된 방법이 전체 회로에 대해서 고른 성능 향상을 나타내고 있다. 논문 [29]의 방법은 본 논문과 유사한 BIST 스킴(scheme)을 제안하고 있으며, 이 논문의 테스트벡터의 인가방식에 있어 근본적인 접근 방식은 논문[6], [31] 등에서 언급한 단일 입력비트 변화(single input change)를 사용하고 있다. 즉 확률값이 고려된 초기화 벡터를 처음 인가하고 그 다음 전파벡터를 인가할 때 초기화 벡

터와 단지 1비트만 다르게 첫번째 주입력 신호선의 비트값을 반전시켜 전파 벡터를 인가한다. 그리고 3번째 테스트 벡터로 다시 처음에 인가 되었던 초기화 벡터를 기억시켜 놓았다가 그대로 회로에 인가하고 다시 전파 벡터를 인가할 때는 이제는 두번째 주입력선의 비트값만을 초기화벡터와 다르게 반전시켜 전파 벡터로 인가 한다. 그리고 이러한 과정을 주입력선의 마지막 신호선의 값을 반전시켜 인가하고 난 후 (2n + 1 테스트클럭 후 (n은 주입력선의 신호선 수)), LFSR을 통해 나온 출력값이 두번째 초기화벡터를 만들고 위와 같은 일련의 과정을 반복하여 로보스트 고장 검출의 수를 늘리려는 시도이다. 그러나 비교적 작은 회로인 c432, c499, c1908의 경우에 있어서는 제안된 방법보다 높은 로보스트 고장 검출율을 보이지만 c3540의 회로를 제외한 c2670회로에서 부터는 회로 크기가 커지고 주입력선의 핀수가 증가함에 따라 로보스트와 넌로보스트의 고장 검출율이 급격하게 떨어지는 것을 알 수 있다. 실험 결과 이러한 양상을 보이는 이유로는 소규모 회로일 때는 단일 비트만 변화하는 테스트 벡터쌍을 인가하더라도 쉽게 경로 지연 고장 효과가 출력측에 전파되지만 회로가 커지고 복잡하여 지면 다중 입력비트의 변화가 로보스트 경로 지연 고장 효과를 보다 용이하게 전파시킨다는 것을 의미한다.

표 6. 다른 방법과의 비교 (로보스트 / 넌로보스트)

Table 6. Comparison of other methods (robust / nonrobust)

Circuit	PI	Gate	[29]	[30]	this paper
c432	36	153	901/1590	930/7659	518/7655
c499	41	170	2498/7757	853/75567	221/73347
c880	60	357	1063/1181	1107/5915	1110/6101
c1908	33	880	3420/3685	1208/80256	1198/63852
c2670	233	1193	382/999	2261/70179	1949/25380
c3540	50	1647	4370/36592	1743/192286	1949/263852
c5315	178	2184	487/595	5415/236972	5287/112388
c7552	207	3513	454/807	3735/13062	4063/143919

또한 이러한 기능을 실현시키기 위한 부가 하드웨어로 제어 논리와 가중화논리를 포함하여 n단의 LFSR과 2n+1단의 SSR(scan shift register) 를 사용하고 있으므로 제어논리와 가중화 논리 그리고 한 개의 32비트 LFSR을 사용하는 본 논문의 방법보다 많은 H/W 오버헤드가 있다는 것을 알 수 있다. 따라서 일반적으로 규모가 작은 회로보다 대규모회로에 적용되는 BIST논리에 본 논문의 방법을 적용시킨다면 높은 고장검출수와 상대적으로 적은 H/W 오버헤드로 인하여 보다 우수한 성능을 나타내리라 사료된다.

VI. 결 론

본 논문에서는 조합 회로의 효율적인 경로 지연 고장 테스트 생성 알고리듬을 제안하였다. 제안된 경로 지연 고장 테스트 생성 방식은 테스트 용이도의 개념을 이용하여 주입력의 지연 테스트 벡터의 확률값을 생성하도록 하였으며, 테스트 용이도의 관측도 값을 이용하여 주출력에서 주입력으로 진행하면서 각 게이트의 비제어값의 초기화 벡터의 확률값을 생성하도록 하였다. 따라서 생성된 지연 테스트 벡터는 임의 테스트 벡터의 실험 결과보다 로보스트하고 보다 긴 경로를 검출할 수 있는 테스트임이 실험을 통해 알 수 있었다. 또한, 생성된 지연 테스트 벡터를 입력으로 병렬 패턴 경로 지연 고장 시뮬레이션을 수행하였으며 다른 방법들과의 실험 비교를 통하여 적용한 벤치마크의 회로들에 대해 전반적으로 향상된 로보스트 고장 검출율을 나타내었다. 제안된 경로 지연 고장 테스트 시스템은 yacc 및 C 언어를 사용하여 구현하였으며, ISCAS'85 벤치마크 회로를 대상으로 그 타당성과 효율성을 입증하였다.

참 고 문 헌

[1] Jacob Savior and William H. Mcanney, "Random pattern testability of delay faults," *IEEE Trans. Computers*, vol.37, pp.291-300, March, 1988.

[2] Patrick C. McGeer, "Robust path delay-fault testability on dynamic CMOS circuits," *ICCD 91*, pp.206-211.

[3] S. Koeppe, "Modeling & simulation of delay faults in CMOS circuits," *Proc. Int. Test Conf.*, pp.530-536, 1986.

[4] T. Storey et al., "Delay test simulation," *Proc. 14th Design Automation Conf.*,

pp.492-494., 1977.

[5] Lesser, J., et al, "An experimental delay test generator for LSI logic," *IEEE Trans. Computers*, pp.235-248, 1980.

[6] Gordon L. Smith, "Model for Delay Faults Based upon Path," *Proc. Int.Test Conf.*, pp.342-349, 1985.

[7] Vijay S. Iyengar, Barry K. Rosen and Ilan Spillinger, "Delay test generation -- concepts and coverage metrics," *Proc. Int. Test Conf.*, pp.857-866, 1988.

[8] Irith Pomeranz and Sudhaker M. Reddy, "An Efficient Non-Enumerative Method to Estimate Path Delay Fault Coverage," *Proc. IEEE Int. Test Conf.*, pp.560-567, 1992.

[9] Manfred Henftlig, Hannes C. W. and Kurt J. A., "Path Hashing to Accelerate Delay Fault Simulation," *Proc. IEEE Design Automation Conf.*, pp.522-526, 1994.

[10] C. J. Lin and S. M. Reddy, "On delay fault testing logig in circuits," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 277-284, Apr. 1987.

[11] K. Fuchs, F. Fink, and M. H. Schulz, "DYNAMITE: An efficient automatic test pattern generation system for path delay faults," *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 1323-1335, Oct. 1991.

[12] Tom Michael Niermann, "Concurrent automatic test generation for delay fault test," Technical report, CSG-102, Univ. of Illinois at urbana-champaign, 1989.

[13] Ankan K. Pramanick, and Sudhakar M. Reddy, "On Multiple Path Propagating Tests for Path Delay Faults," *IEEE International Test Conference*, pp. 393-402, 1991.

[14] Srimat T. Chakradhar, Mahesh A. Iyer, and Vishwani D. Agrawal, "Energy Models for delay Testing," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, No. 6, pp. 728-739, June. 1995.

[15] Eun Sei park and M. Ray Mercer, "An Efficient Delay Test Generation System for Combinational Logic Circuits," *IEEE Trans. Computer-Aided Design*, vol. 11, No 7, pp. 926-938, July. 1992.

[16] V. S. Iyengar, B. K. Rosen, and J. A. Waicukauski, "On computing the sizes of detected delay faults," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 299-312, Mar. 1990.

[17] Goldenstein, L. H and E.L.Thigpen, " SCOAP:Sandia Controllability/Observability Analysis program," *Proc., 17th DAC., pp.190-196, June, 1980.*

[18] Benetts R.G., and C.M.Maunder and G.D.robinson, "CAMELOT:A Computer-Aided Measure for Logic Testability," *Proc., ICCC, pp.1162-1165, Oct.,1980.*

[19] Brglez,F. P.Pownall and R.Hum, "Application of testability analsis: From ATPG to critical delay path tracing," *Proc., ITC, pp705-712, Oct, 1984.*

[20] C. J. Lin and S. M. Reddy, "On Delay Fault Testing in Logic Circuits," *IEEE Trans. CAD*, 1987, pp.694-703, Sept 1987.

[20] Paul U. Bardell, et al *Built-In Test for VLSI:Pseudorandom techniques* pp.52-59, 1987 John wiley & Sons.

[21] U. sparmann et al, "Fast Identification of robust dependant path delay faults," *Proc. DAC 95, pp119-125.*

[22] Franz Fink, Karl Fuchs and Michael H. Schulz, "Robust and Nonrobust Path Delay Fault Simulation by Parallel Processing of Patterns," *IEEE Trans. Computers*, Vol 41, 12, Dec.,

pp.1527-1536, 1992.

[23] Yuejian Wu and Ander Ivanov, "Accelerated Path Delay Fault Simulation," *Proc. IEEE Int. Test Symposium.*, pp.1-12., 1992.

[24] M. H Schulz and F. Brglez, "Accelerated Transition Fault Simulation," *Proc. 24th DAC*, pp.237- 243, June, 1987.

[25] E. S. Park, M. R. Mercer and T. W. Williams, "The Delay Fault Model and Statistical Delay Fault Coverage," *IEEE Trans. Computers*, Vol. 41, No. 6, pp.688-698, June 1992.

[26] Susheel J. Chandra and Janak H. patel, "Experimental evaluation of Testability for Test Generation," *IEEE Trans. Computer-Aided Design*, vol. 8, No 1. pp. 93-97, January, 1989.

[27] Paul U.Bardell, et al. *Built-In Test for VLSI:Pseudorandom techniques*, pp.52-59, 1987, John wiley & Sons.

[28] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuit and a Target Translator in Fortran," *Proc. IEEE Symp. and Circuit and Systems Special Session on ATPG and Faults Simulation*, 1985.

[29] Weili Wang and Sandeep K. Gupta, "Weighted Random Robust Path Delay Testing of Synthesized Multilevel Circuits," *12th VLSI Test Symposium*, pp.291-297, 1994.

[30] 허용민 외 ,"조합논리회로의 경로지연고장 검출을 위한 가중화 임의 패턴 테스트 기법," 대한전자공학회 논문지 제32권, A편, 제12호

[31] C. Thomas Glover and M. Ray Mercer, "A deterministic approach to adjacency testing for delay faults," *26th DAC*, pp.351-356, 1989.

저 자 소 개

許 龍 珉(正會員) 第 31卷 A編 11號 參照
현재 한양대학교 전자공학과 대학원 박사과정

林 寅 七(正會員) 第 31卷 A編 11號 參照
현재 한양대학교 전자공학과 교수