논문 2004-41SD-5-10

학습 정보를 이용한 테스트 용이도 척도의 계산

(New Testability Measure Based on Learning)

김 지 호*, 배 두 현*, 송 오 영**

(Jiho Kim, Duhyun Bae, and Ohyoung Song)

요 약

본 연구는 테스트 패턴 생성 알고리즘에서 결정 과정을 안내하는 데 이용되는 새로운 테스트 용이도 척도 계산법을 제안한다. 이 테스트 용이도 척도는 학습에 의해 얻어지는 회로의 구조적 정보를 이용한다. 제안된 테스트 용이도 척도는 오직 하나의 해결책이 존재할 경우 모순조건을 조기에 찾아내는 패턴을 유도하며, 반면에 다수의 해결책들이 존재할 경우 최소한의 모순이 발생토록 유도한다. 제안된 테스트 용이도 척도는 기존의 방법과 동일한 고장 검출율을 얻는 패턴을 얻는데 소요되는 CPU 시간을 상당히 감소시킨다.

Abstract

This paper presents new testability measure based on learning, which can be useful in the deterministic process of test pattern generation algorithms. This testability measure uses the structural information that are obtained by learning. The proposed testability measure searches for test pattern that can early detect the conflict in case of the hardest decision problems. On the other hand, in case of the easiest decision problem, it searches for test pattern that likely results in the least conflict. The proposed testability measure reduces CPU time to generate test pattern that accomplishes the same fault coverage as that of the distance-based measure.

Keywords: testability, ATPG, VLSI, DFT, learning

I. 서 론

휴대폰, PDA, 노트북 등과 같은 디지털 시스템의 소형화, 경량화와 더불어 주문형 반도체나 범용 마이크로프로세서와 같은 고집적 회로의 이용이 기하 급수적으로 증가해왔다. 그리고 설계 및 집적 기술이 급속도로 발달하여 단일 칩에 집적되는 논리 게이트의 수는 이미 수백만 단위에 이르게 되었다. 하지만 반도체 집적도의 증가는 상대적으로 회로에 대한 테스트를 더욱 어렵게 만들었고, 시장 경쟁력 있는 칩을 만들기 위해서 설계 단계에서부터 테스트를 고려한 설계 기법이 필수적이 되었다. 실제로 대부분의 상용칩들이 설계 단계에서부터 테스트를 고려하여 설계를 하고 있다. 집적회로의 검사는 주요 입력단(Primary Input : PI)의 값을 제어하여 검사 대상이 회로 부위를 원하는 값으로 만들고 이에 따른 회로값의 변화를 주요 출력단(Primary Output : PO)으로 보내어 측정하는 것으로 이루어진다. 그러나 집적회로의 집적도의 증가에 비해 입출력 핀의 수는 제한적으로 증가하고 있기 때문에 적은 수의 입출력 핀으로 상대적으로 더 많은 회로 내부의 값들을 제어하고 측정해야 하는 부담을 안게 되었다.

현재까지 순차회로 및 조합회로의 testability[1][2] 및 자동 테스트 패턴 생성[3][4]에 관한 많은 연구가 진행되었다. 본 논문에서는 조합 회로에 대해서 기존의 자동 검사 패턴 생성 알고리즘의 효율성을 향상시킬 수 있는 새로운 테스트 용이도 척도를 제안한다. 이를 위하여 기존의 알고리즘들 중에서 우수한 성능을 갖는 부분을 선택적으로 이용하고, 학습 기법, 테스트 용이도 척도 기법의 조합을 통한 새로운 방법을 제안하였다.

* 학생회원, ** 정회원, 중앙대학교 전자전기공학부
(School of Electrical and Electronic Engineering, Chung-Ang University)

※ 이 논문은 2003년도 중앙대학교 학술연구비 지원에 의한 것임
접수일자: 2004년1월20일, 수정완료일: 2004년4월22일

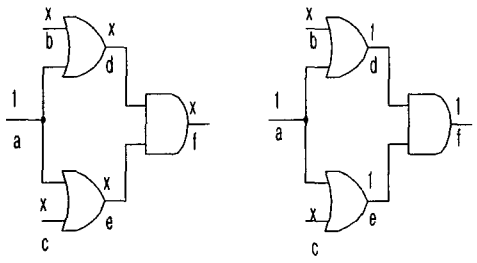


그림 1. a=1로부터의 Implications
Fig. 1. Implications from a = 1.

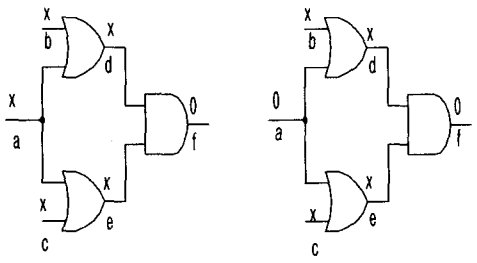


그림 2. f=0으로부터의 Implications
Fig. 2. Implications from f = 0.

Ⅱ. 기반 알고리즘

1. 학습(Learning) 기법

학습(learning) 기법은 SOCRATES 시스템[5]에서 처음 제안된 방식으로 패턴 생성 이전에 학습을 통하여 회로의 구조상 필수적으로 인가되는 값과 신호선 사이의 관계를 정리하는 방법에서 패턴생성 탐색 과정 중에도 학습이 이루어지는 동적 학습[6]으로 개선되었다.

그림 1에서 신호 a에 논리값 '1'되고, 현재 이 부분에는 회로의 고장이 없다면, 논리값은 '0', '1', 'X'로만 구성된다. 그러면 FAN 알고리즘[7]의 Implication 과정은 d=1, e=1, f=1이 된다. 그림 2는 신호 f 가 논리값 '0'이 된다면, 신호 f는 AND 게이트의 출력신호이고 게이트 입력인 신호 d와 e가 'X'값을 가지므로 어떠한 Implica-tion 과정도 수행되지 않는다.

여기서, 논리 명제(logical identity)중에서 대우 명제(contrapositive)의 원리를 적용할 수 있다. 즉,

$$(P \Rightarrow Q) \Leftrightarrow (\neg Q \Rightarrow \neg P) \qquad (1)$$

의 관계를 이용하여 P 대신에 "a=1"을 Q 대신에 "f=1"을 대입하면,

$$\{(a=1) \Rightarrow (f=1)\} \Leftrightarrow \{\ \neg(f=1) \Rightarrow \neg(a=1)\ \} \qquad (2)$$

의 관계가 성립한다. 이 명제를 다시 표시하면

$$\{(a=1) \Rightarrow (f=1)\} \Leftrightarrow \{\ (f=0) \Rightarrow (a=0)\ \} \qquad (3)$$

이 된다. 그러므로 신호선 f가 '0가 된다면 신호선 a는 항상 '0'이 되는 것이다. 이러한 부가적인 Implication은 결정적 검사 패턴 생성(DTPG: Deterministic Test Pattern Generation) 수행동안에 탐색 공간의 향상된 가지치기와 모순(conflict) 및 논리 과잉(redundancy)의 초기 인식과 되돌아감의 횟수의 감소를 가져온다.

학습 과정은 DTPG 수행동안에 Implication에서 이득을 얻기 위해서 DTPG 수행에 앞서서 수행된다. 학습 과정은 회로의 특정 신호에 값을 인가하고, 이로부터 모든 Implication을 수행하여 Implication 결과로부터 학습하는 것이다. 이것은 회로의 모든 신호의 논리값 '0'과 '1'에 대해 수행된다.

위에서 설명된 학습의 수행이 가치 있는 Implication 정보로 이용되기 위해선 아래의 학습 기준에 따라야 한다.

2. 학습 기준(Learning Criterion)

신호 i 를 현재의 학습 단계에서 신호값 v_i 인가에 의해서 초기화되는 신호라 하고, j 를 Implication 과정동안에 고정된 논리값 v_j ('0' 또는 '1')인 신호라 하자. 예를 들어,

$$(i = v_i) \Rightarrow (j = v_j) \qquad (4)$$

j를 게이트 g의 출력 신호라 할 때, 만약

(1) v_j 가 게이트 g의 모든 입력에 출력 무영향 논리값(noncontrolling value)을 갖도록 요구하고,

(2) 전방 Implication이 $j = v_j$가 되었을 때 대우명제를 적용하면,

$$(j = \overline{v_j}) \Rightarrow (i = \overline{v_i}) \qquad (5)$$

은 가치 있는 학습정보로 판단된다.

학습의 개념은 DTPG의 전처리 과정에서 성공적으로 적용되고 있으며 순차 회로의 자동 검사 패턴 생성에도 시간 비용 이득을 가져온다.

3. 테스트 용이도 척도

자동 검사 패턴 생성 시에 탐색 과정을 고속으로 수행하기 위해 테스트 용이도(testability)를 측정한다.[8]

이러한 테스트 용이도 측정은 다음과 같은 원리를 기반으로 한다.

(1) 해결책이 오직 한 개뿐인 별개의 문제 중에서는, 가장 어려운 것을 우선 공략한다. 이렇게 하는 이유는 어려운 것이 해결되지 않은 상태에서 상대적으로 해결 가능한 것을 해결하는데 쓸데없이 소비되는 시간비용을 피하기 위해서이다.
(2) 해결책이 여러 개가 존재하는 별개의 문제 중에서는 가장 쉬운 것을 우선 공략한다.

테스트 용이도 척도(testability measure)는 주로 비용함수(cost function)로 불려지며, 다음 두 가지 형태를 갖는다.

- 제어 용이도 척도(controllability measure): 라인에 값을 설정하는데 드는 상대적 어려움
- 관측 용이도 척도(observability measure): 고장을 어떤 라인에서부터 주출력단(PO)까지 전파시키는데 드는 상대적인 어려움

제어 용이도 척도(controllability measure)는 해결이 힘든 논리값 정당화 문제를 선택하거나(즉, 임의의 게이트 G에 논리값 v를 인가하는 것), 게이트의 제어값(controlling value)을 인가해주는 신호를 선택하는 데 이용된다. 관측 용이도 척도(observability measure)는 D-frontier들 중에서 입력단의 고장이 가장 쉽게 관측될 수 있는 게이트를 선택하는 데 이용된다. 이들은 모두 상대적인 척도만을 제공한다. 그리고 테스트 용이도 척도는 검사 패턴 생성의 전처리 단계에서 미리 계산되어진다.

본 논문에서는 테스트 용이도 척도 중에서 제어 용이도를 이용하며, 이것은 FAN 알고리즘에서 후방 추적(backtrace) 과정의 안내를 위한 척도로서 역할을 담당한다. 위에서 언급된 두 가지 원리를 바탕으로 게이트 입력단에 임의의 논리값 v가 필요할 때, 논리값 v가 해당 게이트의 출력 제어값(controlling value)이라면, 논리값 v를 설정하기에 가장 용이한 입력을 선택하고, 아니면 가장 어려운 입력을 선택한다.

4. 거리 기반 테스트 용이도 척도

일반적으로 테스트 용이도 척도로 거리 기반의 테스트 용이도 척도(Distance-based cost function)를 많이 사용한다. PI는 값을 제어하기에 가장 쉽고, PO는 값을 관측하기에 가장 쉽다. 이런 관점에서 보면, PI로부터 멀어지면 제어하기가 어려워지고, PO에서 멀어지면 관측하기가 어려워진다. 그러므로 라인의 제어 용이도 척도(controllability)는 그 라인의 단계(level)에 의해서 결정되고, 라인의 관측 용이도 척도(observability)는 PO로부터의 그 라인의 최소 거리에 의해 결정된다. 이 방식은 회로의 논리 게이트 등의 정보는 고려 대상에 두지 않으며 오직 PI와 PO와의 거리 정보만을 고려한다.

본 논문에서는 거리기반 테스트 용이도 척도의 개념과 학습정보 개념을 이용하여 새로운 테스트 용이도 척도를 제안하며 다음절에 자세히 설명한다.

III. 학습정보를 이용한 테스트 용이도

본 논문에서는 기존의 DTPG 알고리즘 중의 하나인 FAN 알고리즘에 기존의 휴리스틱인 학습과 테스트 용이도 척도를 조합한 새로운 방식의 휴리스틱을 적용하여 검사 패턴 생성 알고리즘의 효율성을 향상시키고자 하였다.

본 논문에서 사용된 기반 휴리스틱은 다음과 같다.

- 정적 학습(static learning)
- 거리 기반의 테스트 용이도 척도
- 단일 후방 추적 기법(single backtrace)

회로의 자동 검사 패턴 생성 시에 패턴 생성의 전처리 단계에서 학습(learning) 과정을 수행하여 어떤 특정 신호선이 특정 논리값을 갖기 위한 필요조건을 미리 구하여 놓고 논리값 정당화 과정에서 활용하게 된다. 이러한 학습 과정은 회로의 게이트 함수가 반영되는 구조적인 정보를 저장하게 된다. 학습의 수행은 결정적 검사 패턴 생성(DTPG) 수행 단계에서 미리 특정 신호선 간의 관계 정보를 저장하여 Implication시에 탐색 공간을 최적화 시켜 수행 시간을 감소시키고, 어려운 고장에 대한 검출 여부를 판별하는데 도움을 준다. 그림 3과 그림 4에서 보면, 신호선 D와 H의 관계가 설명된다. 신호선 D가 '0'값을 가지면 신호선 H는 '0'을 갖게 된다. 학습 기준에 의해 두 신호선의 관계는 가치 있는 학습정보로 저장이 되고, 이것은 H=1로부터의 Implication 수행 시에 곧바로 D=1이 된다.

테스트 용이도 척도는 회로내의 게이트간의 연결정보를 이용해서 DTPG 수행 시 후방 추적 과정에서 안

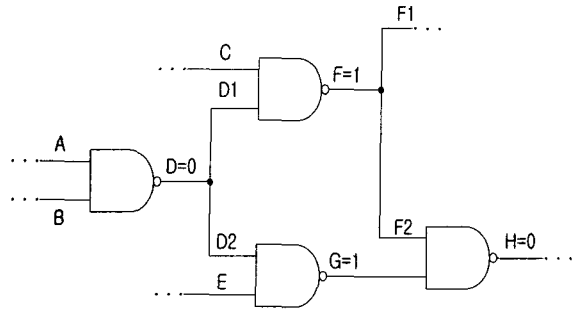


그림 3. D=0으로부터의 Implications
Fig. 3. Implications from D = 0.

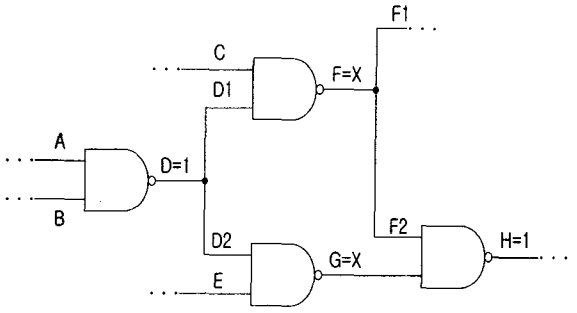


그림 4. H=1로부터의 Implications
Fig. 4. Implications from H = 1.

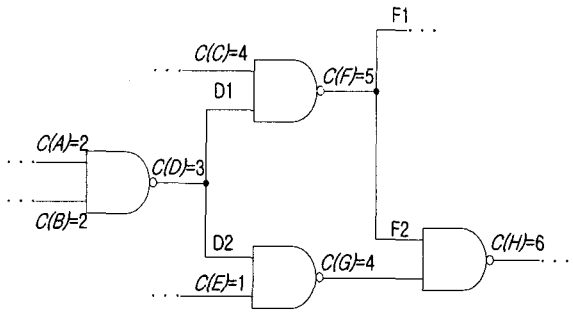


그림 5. 거리 기반 제어 용이도 척도의 적용
Fig. 5.Example of distance-based testability.

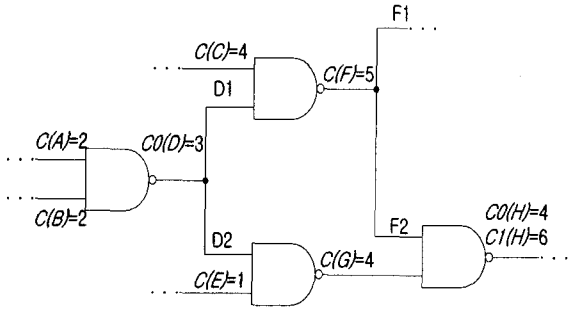


그림 6. 학습 정보를 이용한 제어 용이도 척도의 적용
Fig. 6. Example of learning-based testability.

내역할을 돕는다. 본 논문에서 이용한 거리 기반의 테스트 용이도 척도 계산법에 의해 그림 3의 회로에 대해 각 신호선의 테스트 용이도 척도를 계산하면 그림 5와 같다. 먼저 제어 용이도 척도는 신호선 A, B, C, E에 대하여 제어 용이도 척도 C(A)=C(B)=2, C(C)=4, C(E)=1 이라고 가정했을 때 신호선 D, F, G, H의 제어 용이도 척도 C(D)=3, C(D1)=3 ,C(D2)=3, C(F)=5, C(F1)=5, C(F2)=5, C(G)=4, C(H)=6 의 값을 가지게 된다.

다음 관측 용이도 척도는 신호선 H를 그림에서 신호선 F1에 연결된 신호선 보다 관측 용이도 값이 크며 그림의 신호선 중에서 가장 PO에 가까운 신호선이고, 이것의 관측 용이도 척도 O(H)=3이라고 가정하면. 신호선들의 관측 용이도 척도는 O(F)=O(F1)=4, O(F2)=4, O(G)=4, O(C)=O(D)=5, O(D1)=5, O(D2)=5, D(E)=5, O(A)= O(B)=6 의 값을 가진다.

그림 3에서는 학습 과정에서 신호선 D와 신호선 H의 관계를 가치 있는 정보로서 저장하게 된다. 그리고 이것은 자동 검사 패턴 생성 시 Implication 수행에서 도움을 주게 된다. 일반적으로 테스트 용이도 척도는 회로 내의 게이트간의 연결 관계만을 토대로 계산하고 있다. 그리고 학습은 회로의 구조적 정보 즉, 회로내의 게이트의 기능성에 의해 필연적으로 정해지는 신호선들의 관계를 밝혀내고 있다. 그러므로 회로의 학습의 정보를 이용하여 신호선들의 테스트 용이도 척도를 회로의 구조 및 기능을 더 정확하게 반영하여 계산할 수 있다.

그림 3의 회로를 포함한 임의의 회로에 대해서 검사 패턴을 생성하고자 할 때, 전처리 과정에서 수행된 학습은 신호선 D와 신호선 H에 대해서

$$(D = 0) \Rightarrow (H = 0) \quad (6)$$

학습 기준에 의해서

$$(H = 1) \Rightarrow (D = 1) \quad (7)$$

의 학습 정보를 저장하고 있다.

여기에서, 신호선 D가 '0'값을 가지게 되면 필연적으로 신호선 H가 '0'값을 가지게 된다는 것을 알 수 있다. 이 관계는 신호선 H는 신호선 D가 '0'값을 가지기만 하면 항상 '0'값을 가지게 되므로 결과적으로 신호선 H에 '0'값을 설정하기 위해서는 PI에 신호선 D를 '0'으로 설정할 수 있는 초기 입력을 인가하면 된다.

그러므로 신호선 H의 논리값 '0'에 대한 제어 용이도 척도 $C_0(H)$ 는 신호선 D의 논리값 '0'에 대한 제어 용이도 척도 $C_0(D)$ 에 의존하게 된다. 제안된 학습의 정보를 적용하면, 그림 6에서 $C_0(H)$=4가 되고 제안된 테

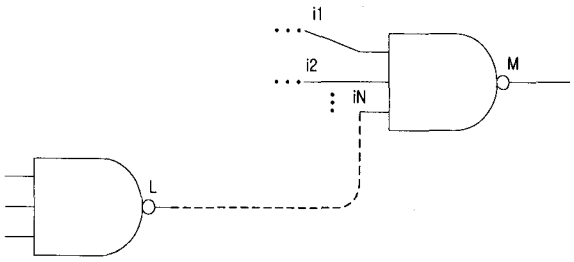


그림 7. 신호선 L과 M의 학습
Fig. 7. Learning signal lines L and M.

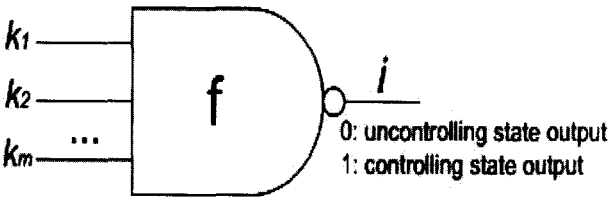


그림 8. 신호선 i의 제어 용이도 척도
Fig. 8. Controllability of signal line i.

스트 용이도 척도 기법에 의한 신호선 H의 '0' 제어 용이도 척도 $C_0(H)$는

$$C0(H)=\min\{C0(D) + 1, C0(H)\} \quad (8)$$

로 계산 된다.

그림 7의 회로에서 신호선 L과 M이 가치 있는 학습 정보로 저장된다고 가정하고 이들의 관계는 다음과 같다고 하면,

$$(L = 0) \Rightarrow (M = 0) \quad (9)$$

$$(M = 1) \Rightarrow (L = 1) \quad (10)$$

이 경우, 신호선 M에 대한 제어 용이도 척도 $C_0(M)$은 다음과 같다.

$$C0(M) = \min\{C0(L)+1, C0(M)\} \quad (11)$$

마찬가지로 신호선 L과 M의 학습 관계가

$$(L = 1) \Rightarrow (M = 0) \quad (12)$$

$$(M = 1) \Rightarrow (L = 0) \quad (13)$$

이라고 하면, $C_0(M)$은 다음과 같다.

$$C0(M) = \min\{C1(L)+1, C0(M)\} \quad (14)$$

모든 게이트의 출력 신호선의 제어 용이도 척도는 출력값이 uncontrolling state인지 controlling state인지에 따라서 계산하는 방식이 달라진다. 그림8에서 NAND

```
learn_calculate_testability_measure()
{
  for every signal i in levelized order
  {
    asign_value(i,0)
    implication()
    analyze_result(i)
    asign_value(i,1)
    implication()
    analyze_result(i)
  }
}

analyze_result(i)
{
  vi = value of signal i
  f = gate with output signal i and input signals k1, k2,., km
  tf = type of gate f
  g = gate with output signal j
  tg = type of gate g
  vj = value of signal j
  for every signal j with value of j≠X and with j≠i
  {
    if( vj=1 and (tg=AND or tg=NOR ) or
        vj=0 and (tg=OR or tg=NAND ) or
        tg=XOR or tg=XNOR) then
    {
      if (no direct path from signal j to i exist) then
      {
        save_implication(j=v̄j → i=v̄i)
        Cvi(j)=min{Cvi(i)+1, Cvi(j)}
      }
    }
  }
  if(tf=XOR or tf=NAND or tf=XNOR) then
  {
    w=v̄i
  }
  else
  {
    w=vi
  }
  if( vi=1 and (tf=AND or tf=NOR ) or
      vi=0 and (tf=OR or tf=NAND ) or
      tf=XOR or tf=XNOR) then
  {
    new=max{Cw(k1), Cw(k2), ... , Cw(km)}+1
    Cvi(i)=min{Cvi(i), new}
  }
  else
  {
    new=min{Cw(k1), Cw(k2), ... , Cw(km)}+1
    Cvi(i)=min{Cvi(i), new}
  }
}
```

그림 9. 학습에 기반한 테스트 용이도 척도 계산 알고리즘
Fig. 9. Algorithm for calculating learning-based testability

게이트를 예를 들어 제어 용이도 척도 계산 방식을 설명한다. NAND 게이트에서 신호선 i의 '0' 제어 용이도 척도 $C_0(i)$은 uncontrolling state(입력선을 모두 제어)의 경우

$$C_0(i)=\min\{\max\{C_1(k_1), C_1(k_2),.. , C_1(k_m)\}+1, C_0(i)\} \quad (15)$$

이며, 신호선 i의 '1' 제어 용이도 척도 $C_1(i)$은 controlling state(입력선 중 하나만 제어)의 경우

$$C_1(i)=\min\{\min\{C_0(k_1), C_0(k_2),.. , C_0(k_m)\}+1, C_1(i)\} \quad (16)$$

로 계산된다.

일반적인 학습에 기반한 제어 용이도 척도는 그림9의 learn_calculate_testability_measure()에 자세히 보여준다. 초기에 모든 PI의 제어 용이도 척도 $C_0(PI)$와 $C_1(PI)$와 모든 PO의 관측 용이도 척도 $O_0(PO)$와 $O_1(PO)$는 0이고, 모든 신호 i의 $C_0(i)$와 $C_1(i)$는 신호 i의 PI들로 부터 거리 중 최대값을 가지게 하며 $O_0(i)$와 $O_1(i)$는 신호 i의 PO들로부터 거리 중 최소값을 갖도록 계산한다. 학습에 기반하여 모든 신호 i에 대하여 $C_0(i)$와 $C_1(i)$를 구하는 것은 learn_calculate_testability_measure()를 수행함으로써 구해질 수 있다.

IV. 단일 후방 추적 기법의 이용

일반적으로 테스트 용이도 척도는 검사 패턴 생성 과정에서 탐색 공간의 효율적인 가지치기를 위해 안내 역할을 한다. 테스트 패턴 생성과정 시 후방 추적 과정에서 hardest problem*과 easiest problem**이 발생할 경우 최선의 해결책을 찾아서 가지치기를 시도하도록 해야 한다. 기존의 FAN 알고리즘의 후방 추적 기법(backtrace)은 테스트 용이도 척도를 hardest problem과 easiest problem에 적용하고 있지 않으므로 본 논문에서는 FAN 알고리즘의 단일 후방 추적 기법(single backtrace)에 hardest problem과 easiest problem을 적용하여 성능을 개선하였다.

그림 10은 제안된 단일 후방 추적 기법의 흐름도를

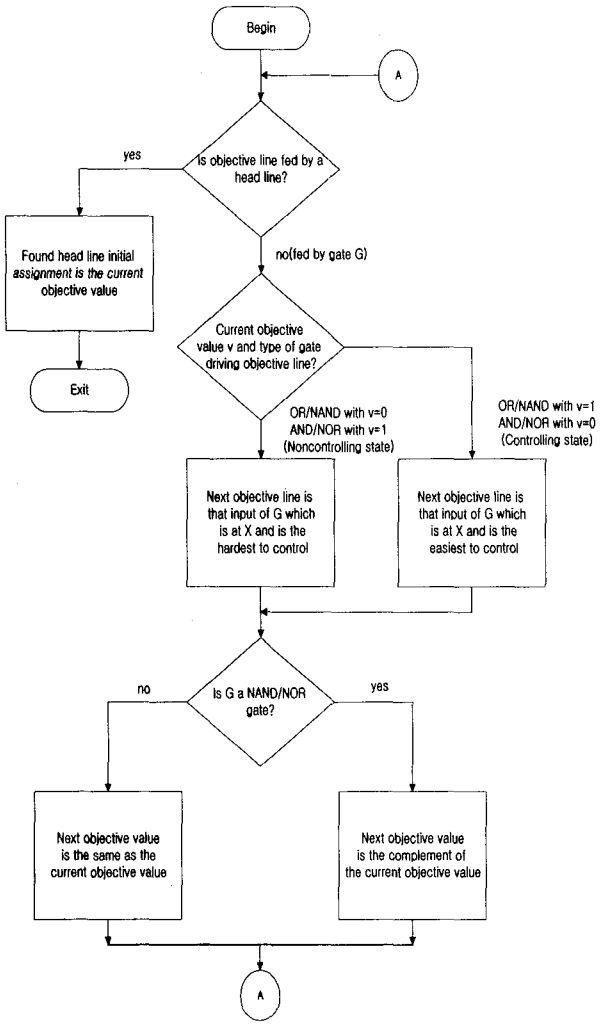


그림 10. FAN 알고리즘에 적용된 단일 후방 추적 흐름도
Fig. 10. Flowchart of single backtrace in FAN algorithm.

보여준다. hardest control인 경우에 입력의 제어 용이도가 가장 큰 입력을 선택하고, easiest control의 경우에는 입력의 제어 용이도가 가장 작은 입력을 선택한다. 전자의 경우 상대적으로 목표 달성이 어려우므로 목표 달성의 실패 여부를 빨리 감지하여 되돌아감을 수행하여 다른 공간을 탐색하는 시간을 감소시키고 후자의 경우 목표를 달성할 수 있는 확률이 상대적으로 높으므로 목표 달성 처리 시간을 감소시킨다.

제안한 방법은 후방 추적 과정에서 소비되는 시간을 감소시키고, 신호선에 값을 빨리 인가하여 그 효과를 판단하게 되었다. 이것은 되돌아감의 수행 여부 판단과 수행 시간을 감소시킬 수 있다. 또한, 검출이 힘든 경우에 모순(conflict)을 조기에 감지하여 검출 여부를 판단하게 되어, 검출이 가능한 고장은 더 빠르게, 검출이 힘든 고장은 검출 불능 판별을 빨리하도록 개선하였다.

* 해결 불가능 문제(unsolvable problem)
** 해결 가능 문제(solvable problem)

V. 실 험

본 논문의 자동 검사 패턴 생성 알고리즘은 C 언어를 이용하여 유닉스 상에서 구현되었다. 새롭게 제안된 자동 검사 패턴 생성 기법의 성능 측정을 위하여 기존의 FAN 알고리즘을 이용한 방법과의 실험을 통하여 비교하였다. 실험은 SUN Solaris/Ultrasparc 167Mhz의 CPU에 256Mbyte 메모리를 가진 워크스테이션에서 실행되었다. 그리고 검사 패턴 생성을 위한 테스트 회로로는 자동 검사 패턴 생성 알고리즘의 성능 측정에 통상적으로 이용되는 ISCAS '85 성능 분석용 회로와 ISCAS '89 순차회로를 완전 스캔하여 조합 회로화한 성능 분석용 회로를 이용하였다.

실험에 이용된 회로의 특성은 표 1에 정리하였다. 회로에 존재하는 게이트의 수(Gate), 주요 입력단의 수(PI), 주요 출력단의 수(PO), 고장의 수(Fault), 학습된 정보량(Learn)의 순서대로 정리하였다.

1. 기존의 방법에 의한 검사 패턴 생성

우선, 기존의 방법을 이용하여 검사 패턴 생성을 실험하였다. 무작위 검사 패턴 생성(RTPG)과 패턴 축약(Pattern Compaction)은 수행하지 않았다. 되돌아감 횟수 제한은 한 fault당 10회로 제한하였다.

표 2에 생성된 패턴의 수와 고장의 검출과 관련된 정보를 나타내었다. 표의 각 열들은 생성된 패턴의 수, 전체 고장의 수, 검출된 고장의 수, 검출 불가능하다고 결정된 고장의 수, 패턴 생성을 포기한 고장의 수, 고장 검출율을 나타낸다. 고장 검출율은 다음과 같이 계산되었다.

$$\text{고장 검출율} = \frac{\text{검출된고장의수}}{\text{전체고장의수}} \qquad (17)$$

표 3에는 검사 패턴 생성의 시간에 관한 정보를 나타내었다. 되돌아감과 수행 시간과의 관계를 보기 위해 되돌아감(backtrack)의 횟수와 결정적 검사 패턴 생성(DTPG) 소요 시간, 고장 시뮬레이션 시간, 전체 패턴 생성 시간을 나타내었다.

2. 제안된 방법에 의한 검사 패턴 생성

다음으로 제안된 방법을 이용하여 실험한 결과를 표 4과 표 5에서 보여준다. 약간의 고장 검출율의 증가와 약 8.4%의 패턴 생성 시간의 감소했음을 알 수 있다. DTPG 시간만 비교하면 더 많은 소요 시간의 감소를

표 1. 성능 분석에 이용된 회로 및 구조정보
Table 1. The circuits and their structural information used for performance analysis.

Circuit	Gate	PI	PO	Fault
c1908	33	25	880	1879
c3540	50	22	1669	3428
c5315	178	123	2307	5350
c6288	32	32	2416	7744
c7552	207	108	3512	7550
s1238	32	32	508	1355
s1423	91	79	657	1515
s5378	214	213	2779	4551
s9234.1	247	250	5597	6927
s13207.1	700	790	7951	9815
s15850.1	611	684	9772	11725
s38417	1664	1742	22179	31180
s38584.1	1464	1730	19253	36303

표 2. 기존의 방법에 의한 검사 패턴 수와 고장 검출율
Table 2. The number of test patterns and fault coverage in the previous method.

Circuit	Pattern	Fault	Detected	Untestable	Aborted	Fault Coverage
c1908	170	1879	7	3	99.468	
c3540	215	3428	137	0	96.004	
c5315	171	5350	59	0	98.897	
c6288	36	7744	34	2	99.535	
c7552	287	7550	71	68	98.159	
s1238	193	1355	66	3	94.908	
s1423	86	1515	14	1	99.010	
s5378	321	4551	40	0	99.121	
s9234.1	491	6927	404	48	93.475	
s13207.1	576	9815	142	9	98.462	
s15850.1	555	11725	380	9	96.682	
s38417	1233	31180	161	4	99.471	
s38584.1	885	36303	1482	24	95.852	

확인할 수 있다.

두 가지 방법에 의한 비교를 표 6에 고장 검출율과 검사 패턴 생성 시간 면에서 보여주었다. 고장 검출율 부분에서는 대부분의 회로가 동일한 고장 검출율을 나타내고 있고 몇 개의 회로가 기존보다 우수함을 볼 수 있다. 수행 시간 면에서는 상당한 시간 단축을 확인할

표 3. 기존의 방법에 의한 되돌아감 횟수 및 검사 패턴 생성 시간

Table 3. The number of backtracks and CPU execution time for test pattern generation in the previous method.

c1908	95	1.767	0.667	2.800
c3540	140	2.867	1.883	5.983
c5315	80	3.533	1.733	6.133
c6288	63	1.750	1.767	4.383
c7552	1184	25.567	4.367	31.817
s1238	145	1.217	0.483	2.133
s1423	25	0.450	0.267	1.000
s5378	40	4.467	3.217	8.733
s9234.1	1044	63.933	19.900	87.183
s13207.1	280	80.567	45.083	132.567
s15850.1	505	99.967	54.683	161.717
s38417	201	1151.750	241.483	1405.950
s38584.1	1816	496.733	216.550	920.150

표 4. 제안된 방법에 의한 검사 패턴 수와 고장 검출율

Table 4. The number of test patterns and fault coverage in the proposed method.

c1908	172	1879	7	2	99.521
c3540	227	3428	129	8	96.004
c5315	188	5350	59	3	98.841
c6288	35	7744	34	0	99.561
c7552	307	7550	73	60	98.238
s1238	195	1355	67	2	94.908
s1423	93	1515	14	0	99.076
s5378	308	451	40	0	99.121
s9234.1	526	6927	382	70	93.475
s13207.1	582	9815	142	9	98.462
s15850.1	566	11725	380	9	96.682
s38417	1271	31180	156	9	99.471
s38584.1	909	36303	1479	27	95.852

표 5. 제안된 방법에 의한 되돌아감 횟수 및 검사 패턴 생성 시간

Table 5. The number of backtracks and CPU execution time for test pattern generation in the proposed method.

c1908	27	1.300	0.683	2.317
c3540	256	3.117	1.967	6.333
c5315	112	3.300	1.900	6.067
c6288	34	1.033	1.833	3.717
c7552	1103	19.650	4.333	25.783
s1238	158	1.150	0.500	2.083
s1423	14	0.350	0.333	0.950
s5378	40	3.767	3.533	8.383
s9234.1	1303	58.450	20.833	82.833
s13207.1	283	65.950	44.233	117.583
s15850.1	508	77.200	53.467	138.233
s38417	254	967.933	238.150	1219.167
s38584.1	2240	434.033	214.267	856.833

표 6. 기존의 방법과 제안된 방법의 비교

Table 6. Comparison of performance between the previous and the proposed methods.

c1908	99.468	99.521	+0.053	2.800	2.317	17.25
c3540	96.004	96.004		5.983	6.333	-5.85
c5315	98.897	98.841	-0.056	6.133	6.067	1.08
c6288	99.535	99.561	+0.026	4.383	3.717	15.2
c7552	98.159	98.238	+0.079	31.817	25.783	18.97
s1238	94.908	94.908		2.133	2.083	2.34
s1423	99.010	99.076	+0.066	1.000	0.950	5
s5378	99.121	99.121		8.733	8.383	4.01
s9234.1	93.475	93.475		87.183	82.833	4.99
s13207.1	98.462	98.462		132.567	117.583	11.3
s15850.1	96.682	96.682		161.717	138.233	14.52
s38417	99.471	99.471		1405.950	1219.167	13.29
s38584.1	95.852	95.852		920.150	856.833	6.88

수 있다. 되돌아감의 시도 횟수가 전반적으로 늘어남에도 불구하고 패턴 생성 시간은 현저히 감소하고 있는데. 이것은 앞에서 예상한 것처럼 hardest problem이 발생하였을 때, 빨리 값을 인가하여 모순됨을 판단하고 다른 탐색 공간으로의 가지치기를 수행하였음을 보여준다. 즉, 모순이 일어날 확률이 높은 경우는 초기에 판별하도록 안내를 하였기 때문이라고 할 수 있다. 생성된 패턴의 수가 다소 증가됨을 볼 수 있는데 이것은 패턴 축약 과정을 거치면 제거됨을 확인하였다. 되돌아감 제한을 10이상으로 증가시키면 aborted fault의 수는 줄어

들 것이다.

VI. 결 론

본 논문에서는 새로운 방식의 테스트 용이도 척도(testability measure)를 제안하였다. 이를 위하여 결정적 검사 패턴 생성 알고리즘으로 기존의 FAN 알고리즘을 기반으로 하여 서로 다른 목적으로 제안된 학습(learning)과 거리 기반의 테스트 용이도 척도(distanced based testability measure)의 개념을 이용하였고, FAN에서 이용된 단일 후방 추적 개념을 채용하여 제안된 방식의 효과를 극대화하였다.

학습을 통하여 얻어진 회로의 구조적 기능적 특성을 테스트 용이도 척도 계산에 반영하여 후방 추적 과정에서 Implication까지의 수행 시간을 현저하게 감소시키고 있음을 실험에서 확인할 수 있었다. 그리고 약간의 고장 검출율의 상승효과까지 얻을 수 있었다. 이것은 이전에 효과적인 탐색 공간의 가지치기가 이루어지지 못하던 것을 새로운 테스트 용이도 척도로 인해 좀 더 나은 공간의 탐색을 수행하였다고 생각할 수 있다. 되돌아감의 시도 횟수가 전반적으로 늘어나는 것도 있지만, 수행 시간은 현저히 줄어들었다. 이것은 되돌아감을 판단하는 시점이 아주 빨라졌음을 의미한다. 이로 인해 다른 공간으로의 탐색이 빠르게 이루어져 그 만큼의 시간이 절약되고 있음을 알 수 있다. 표에서는 보여주지 않지만, 기법의 특성상 되돌아감(backtrack)의 시도 횟수 제한을 늘려주면 패턴 생성 시에 발생하는 aborted fault의 수가 줄어 들 것이다. 하지만, 수행 시간의 증가는 만족할 만큼의 범위 안에 여전히 유지되게 될 것이다.

본 논문에서는 거리 기반의 테스트 용이도 척도(distanced-based testability measure)를 기반으로 학습 정보를 이용하였다. 차후 다중 후방 추적 방식에 제한된 방법을 응용하여 그 결과를 고찰해 보거나 조합 회로뿐만이 아니라 순차 회로에도 적용해도 좋은 결과를 기대할 수 있을 것이다.

참 고 문 헌

[1] Rubin A. Parekhji, "Testing embedded cores and SOCs-DFT, ATPG and BIST solutions," Proceedings of the 16th International Conference on VLSI Design, pp. 17, 4-8 Jan. 2003

[2] H. Date and T. Hosokawa, "A SoC test strategy based on a non-scan DFT method," Proceedings of the 11 th Asian Test Symposium, pp.305-310, 18-20 Nov. 2002

[3] J. Hirase, "Test pattern length required to reach the desired fault coverage," Proceedings of the 12th Asian Test Symposium, pp.508, 16-19 Nov. 2003

[4] Z. Jiang and S. K. Gupta, "A test generation approach for systems-on-chip that use intellectual property cores," Proceedings of the 12th Asian Test Symposium, pp.278-281, 16-19 Nov. 2003

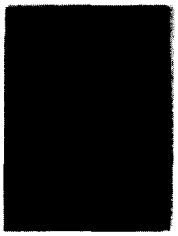
[5] M. H. Schulz and E. Trischler, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System", IEEE Trans. on Computer-Aided Design, Vol.7, no.1, pp.126-137, Jan. 1988.

[6] W. kunz and D. K. Pradhan, "Accelerated Dynamic learning for Test Pattern Generation in Combinational Circuits," IEEE Trans. on Computer-Aided Design, Vol.12, no.5, pp.684-694, May. 1993.

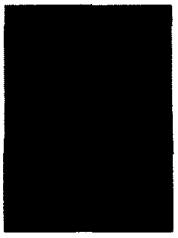
[7] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms," IEEE Trans. on Computer-Aided Design, Vol.C-30, pp. 215-222, 1983.

[8] Miron Abromovico and Melvin A. Breuer, "Digital Systems Testing and Testable Design," IEEE Press, 1990.

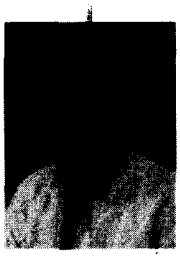
저 자 소 개



김 지 호(학생회원)
2000년 중앙대학교
전자전기공학부 학사
2002년 중앙대학교
전자전기공학부 석사
2002년~현재 중앙대학교
전자전기공학부 박사과정
<주관심분야 : Fault Tolerant Computing, VLSI Design and Testability, Ubiquitous Computing>



송 오 영(정회원)
1980년 서울대학교
전기공학과 학사
1982년 한국과학기술원
전기전자공학과 석사
1991년 University of Massachusetts
컴퓨터공학과 박사
1994년~현재 중앙대학교 전자전기공학부 교수
<주관심분야: Fault Tolerant Computing, Mobile Computing & Communication, Ubiquitous Computing>



배 두 현(학생회원)
2001년 중앙대학교
제어계측공학과 학사
2003년 중앙대학교
전자전기공학부 석사
2003년~현재 중앙대학교
전자전기공학부 박사과정
<주관심분야: IT SoC & Platform Design, Wireless Network Security, Ubiquitous Computing>