論文95-32A-4-9

順次回路를 爲한 檢査性 分析法의 擴張

(An Extension of Testability Analysis for Sequential Circuits)

金 信 澤*, 閔 炯 福*

(Shintaek Kim, and Hyoung Bok Min)

요 약

회로에 대한 결함 시험도(fault coverage)를 정확하게 분석하기 위해 결함 시뮬레이터(fault simulator)를 사용한다. 그러나 회로의 크기가 증가함에 따라 그것의 제곱에 비례하여 계산 시간과 메모리의 양이 증가한다. 최근에는 이러한 문제점을 극복하기 위해 근사 값을 이용하는 방법들이 발표되고 있다. COP는 계산 시간이 빠르지만 순차회로에 대한 적용이 불가능하다. 이에 반해 STAFAN에서는 순차회로에 대한 적용은 가능하지만, 결함 배제 시뮬레이션(fault-free simulation)의 결과를 이용하므로 많은 계산을 요구한다. 본 연구에서는 EXTASEC을 제안한다. 이 기법은 조합회로에 대하여 COP와 동일한 방법을 사용하지만 순차회로에 대한 적용이 가능하다. EXTASEC의 특징은 논리 X에 대한 제어율과 후진 선(backward line)에 대한 해석을 도입한 것이다. EXTASEC의 유용성을 입증하기 위해, ISCAS 회로에 대하여 EXTASEC, 결함 시뮬레이터, STAFAN, 그리고 COP의 결과들을 비교하였다.

Abstract

Fault simulators are used for accurate evaluation of fault coverages of digital circuits. But fault simulation becomes time and memory consuming job because computation time is proportional to square of size of circuits. Recently, several approximate algorithms for testability analysis have been published to cope with the problems. COP is very fast but cannot be used for sequential circuits, while STAFAN can be used for sequential circuits but requires large amount of computation because it utilizes logic simulation results. In this paper EXTASEC(An Extension of Testability Analysis for Sequential Circuits) is proposed. It is an extension of COP in the sense that it is the same as COP for combinational circuits, but it can handle sequential circuits. X-controllability and backward line analysis are key concept for EXTASEC. Performance of EXTASEC is proven by comparing EXTASEC with a falut simulator, STAFAN, and COP for ISCAS circuits, and the result is demonstrated.

* 正會員, 成均館大學校 電氣工學科

(Dept. of Electrical Eng., Sung Kyun Kwan Univ.)

接受日字 : 1994年 3月 29日

I. 서 론

최근 VLSI를 정확하게 그리고 효과적으로 검사(test)하기 위해 많은 연구가 진행되고 있다. VLSI에 대한 검사는 검사 신호(test vector)를 회로에 입력한

후 결함(fault)으로 인한 현상을 출력에서 관찰하는 것이다. 즉 임의의 입력에 대하여 결함이 존재할 때와 존재하지 않을 때의 출력 값이 서로 달라야 그 결함을 검출할 수 있으며, 이때 주어진 회로에서 오직 하나의 결함만이 발생하는 것으로 가정하는 것이 일반적이다. 결함을 모형화 하는 방법으로 가장 널리 사용되는 것이 stuck-at 결함이다. Stuck-at 결함 모형의 장점은, 첫째, 논리 레벨에서 모든 해석이 가능하고, 둘째, 검사의 정확도를 쉽게 확인할 수 있다. 최근에는 CMOS의 stuck-open과 stuck-short등과 같이 공정에 따라 특별한 형태의 결함을 정의하기도 한다[11-12].

검사하려는 회로가 조합회로(combinational circuit)이면 stuck-at 형태의 결함을 한 개의 입력 값으로 검출할 수 있으나, 순차회로(sequential circuit)의 경우는 연속적인 입력 값이 요구된다. 일반적으로 조합회로에 대한 결함 해석보다 순차회로의 결함 해석이 더 복잡하다[3].

회로에서 발생하는 결함을 분석하기 위해 결함 시뮬레이터(fault simulator)를 사용하면 결함 시험도(fault coverage)가 정확하게 계산된다. 그러나 회로를 구성하는 게이트(gate)의 수가 증가함에 따라 그것의 제곱에 비례하여 계산 시간과 메모리의 양이 증가한다. 특히 VLSI의 집적도가 높아짐에 따라, 위와 같은 특성이 큰 제한 요소로 작용한다. 따라서 결함을 임의로 선택하고 그것들에 대하여 분석하기도 하지만, 이 방법으로는 선택되지 않은 결함에 대한 자료를 전혀 얻을 수가 없다. 위의 문제점들을 해결하기 위해 근사값을 사용하거나 확률의 개념을 도입하는 방법이 연구되고 있으며, 확률의 개념을 도입한 대표적인 기법이 COP[4]와 STAFAN[5]이다.

F.Brglez가 제안한 COP에서는 최초 입력(primary input)에 1과 0이 인가될 확률을 각각 50[%]로 정한 후, 회로의 최종 출력 쪽으로 검색하면서 회로 내부의 각 선에 대한 제어율(controllability)을 계산한다. 또 최종 출력(primary output)에 나타난 신호를 관측할 수 있는 확률을 100[%]로 정한 후, 반대 방향으로 검색하면서 각 선의 관측률(observability)을 계산한다. 이 기법은 계산 시간이 빠르고 회로의 복잡도에 대하여 계산 시간이 선형적으로 증가한다. 그러나 순차회로에 적용하기 위해서 그것을 조합회로로 변환(모든 루프를 절단하고 플립플롭을 스캔 가능한 플립플롭으로 대치)하여야 하는 단점이 있다. S.K.Jain과 V.D.Agrawal이 제안한 STAFAN에서는 검사 신호들을 인가하여 결함 배제 시뮬레이션(fault-free simulation)을 수행하면서 회로 내의 각 선에 대한 정보를 수집한다. 그리고 그 정보를 통계적으로 처리하여 각 선에 대한 제어율과 관측률을 계산한다. STAFAN은 순차회로에 대한 적용이 가능하고 정확도가 높다. 그러나 검사 신호에 대하여 결함 배제 시뮬레이션을 매번 수행하므로, 검사 신호의 횟수가 증가할 경우 계산 시간이 길어지고 필요한 메모리의 양도 증가한다[6].

본 연구에서는, STAFAN을 변형하여 순차회로를 해석하고 COP를 확장하여 회로의 각 선에 대한 제어율과 관측률을 계산하는 EXTASEC(An Extension of Testability Analysis for Sequential Circuits)을 제시한다. 즉, 계산 속도가 빠르지만 순차회로에 적용이 곤란한 COP와 속도는 느리지만 순차회로를 해석할 수 있는 STAFAN으로 부터 EXTASEC을 제시함으로서, 순차회로에 대한 해석이 가능하면서 또한 계산 시간을 크게 단축할 수 있게 되었다.

Ⅱ. 선에 대한 제어율, 관측률, 그리고 검출률

회로의 최초 입력에 검사 신호를 인가했을 때 이것에 의해 내부의 선 l이 어떤 논리 값으로 제어될 확률을 그 선의 제어율(controllability)이라 하고, 논리 값 1과 0에 대한 확률을 C1(l)과 C0(l)로 각각 표기한다. 특히 순차회로의 내부에 존재하는 D-타입 플립플롭의 초기 값을 논리 X(논리 1도 아니고 논리 0도 아닌 상태)로 가정하면, 그림 1 처럼 각 게이트의 전달 지연(delay)과 후진 선(backward line)등으로 인하여 회로 내에서 논리 X가 계속 존재한다.

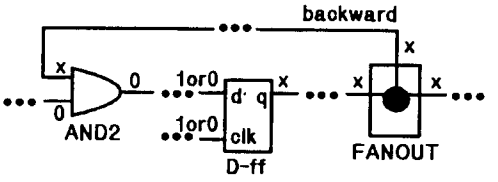


그림 1. D-타입 플립플롭(D-ff)의 초기치
Fig. 1. Initial value of D-type flipflop (D-ff).

본 연구에서는 순차회로에 대한 적용을 위해 논리 X에 대한 제어율을 도입한다. 즉, 검사 신호를 회로에 인가했을 때 선 l이 논리 X로 제어될 확률을 CX(l)로 표기하고, 회로 내의 모든 선에 대하여 CX(l)를 식(1)로 계산한다.

$$CX(l) = 1 - C1(l) - C0(l) \quad (1)$$

선 l의 현재 값(1, 0, 또는 X를 구분하지 않음)을 회로의 최종 출력에서 관측할 수 있는 확률을 그 선의 관측률(observability)이라 하고, 이것을 OB(l)로 표기한다. 회로 내부의 선 l에 대한 C1(l), C0(l), 그리고 OB(l)들을 계산하기 위해 회로의 최초 입력 PI(primary input)에서 l 까지 그리고 l에서 회로의 최종 출력 PO(primary output) 까지의 경로에 포함된 모든 게이트들을 검색한다. 그리고 검색한 각 게이트들에 대하여, 이미 알고 있는 입력 선의 제어율과 출력 선의 관측률을 이용하여 출력 선의 제어율과 입력 선의 관측률을 차례로 계산한다. 이때 각 게이트들의 논리 동작에 따라 계산 방법이 모두 다르다.

선 l에 대하여 검출률(detectability) D1(l)과 D0(l)를 새로이 정의한다. 검출률의 의미는 해당 선에 존재하는 stuck-at-1 또는 -0의 결함을 임의의 한 입력 값을 인가하여 검출할 수 있는 확률이다. 예를 들어 선 l에서 발생한 stuck-at-1의 결함을 검출하기 위해서는, 입력한 검사 신호가 선 l을 논리 0으로 제어하여야 하고 동시에 그 값을 회로의 최종 출력에서 관측할 수 있어야 한다. Stuck-at-0의 경우도 유사한 개념을 적용하여 식(2)과 (3)으로 D1(l)과 D0(l)를 각각 계산한다.

$$D1(l) = C0(l) * OB(l) \quad (2)$$

$$D0(l) = C1(l) * OB(l) \quad (3)$$

Ⅲ. 게이트에 대한 계산식

각 게이트들에 대하여 출력 선의 제어율과 입력 선의 관측률을 계산할 때 적용하는 식은 다음과 같다.

1. 간단한 게이트

최초 입력(PI: Primary Input)에 논리 값 1과 0이 인가될 확률은 각각 50 [%] 이므로, 이들 선의 제어율을 아래 식(4)와 (5)로 계산한다. 그리고 최종 출력(PO: Primary Output)에 나타난 값 들은 항상 관측이 가능하므로 이들 선의 관측률을 식(6)으로 계산한다.

$$C1(PI) = 0.5 \quad (4)$$

$$C0(PI) = 0.5 \quad (5)$$

$$OB(PO) = 1.0 \quad (6)$$

AND 게이트는 입력 선이 모두 1일 때 출력 선도 1로 되고 입력 선이 하나라도 0이면 출력 선이 0이다. 그리고 한 입력 선의 값이 출력 선에서 관측되기 위해서는, 입력 선들 중 해당 입력 선을 제외한 나머지 선들의 값이 모두 1일 경우이다. 따라서 입력 선의 수가 n개인 AND 게이트에 대하여 출력 선을 m이라 하고 i번째 입력 선을 l_i라 할 때, 제어율과 관측률을 식(7), (8), 그리고 (9)로 계산한다.

$$C1(m) = \prod_{i=1}^{n} C1(l_i) \quad (7)$$

$$C0(m) = \bigcup_{i=1}^{n} C0(l_i) \quad (8)$$

$$OB(l_i) = \prod_{k=1,k\neq i}^{n} C1(lk) * OB(m) \quad 단, 1 \le i \le n \quad (9)$$

NAND 게이트는 AND 게이트에서 출력만 반대이다. 입력 선의 수가 n개인 NAND 게이트에 대하여 출력 선을 m이라 하고 i번째 입력 선을 l_i라 할 때, 출력 선에 대한 제어율을 식(10)와 (11)로 계산한다. 입력 선의 관측률은 식(9)와 같다.

$$C1(m) = \bigcup_{i=1}^{n} C0(l_i) \quad (10)$$

$$C0(m) = \prod_{i=1}^{n} C1(l_i) \quad (11)$$

OR 게이트는 입력 선이 모두 0일 때 출력 선도 0이 되고 입력 선이 하나라도 1이면 출력 선이 1이다. 그리고 한 입력 선의 값이 출력 선에서 관측되기 위해서는, 입력 선들 중 해당 입력 선을 제외한 나머지 선들의 값이 모두 0일 경우이다. 따라서 입력 선의 수가 n개인 OR 게이트에 대하여 출력 선을 m이라 하고 i번째 입력 선을 l_i라 할 때, 제어율과 관측률을 식(12), (13), 그리고 (14)로 계산한다.

$$C1(m) = \bigcup_{i=1}^{n} C1(l_i) \quad (12)$$

$$C0(m) = \prod_{i=1}^{n} C0(l_i) \quad (13)$$

$$OB(l_i) = \prod_{k=1,k\neq i}^{n} C0(l_k) * OB(m) \quad 단, 1 \le i \le n \quad (14)$$

NOR 게이트는 OR 게이트에서 출력만 반대이다. 입력 선의 수가 n개인 NOR 게이트에 대하여 출력 선을 m이라 하고 i번째 입력 선을 l_i라 할 때, 출력 선에 대한 제어율을 식(15)와 (16)으로 계산한다. 입력 선의 관측률은 식(14)와 같다.

$$C1(m) = \bigcap_{i=1}^{n} C0(l_i) \quad (15)$$

$$C0(m) = \bigcup_{i=1}^{n} C1(l_i) \quad (16)$$

DELAY 게이트는 입력 선의 값이 그대로 출력으로 전달되며 또 입력 선의 값은 출력 선에서 항상 관측이 가능하다. 따라서 출력 선 m의 제어율과 입력 선 l의 관측률을 식(17), (18), 그리고 (19)로 계산한다.

$$C1(m) = C1(l) \quad (17)$$

$$C0(m) = C0(l) \quad (18)$$

$$OB(l) = OB(m) \quad (19)$$

INVERTER 게이트는 DELAY 게이트에서 출력만 반대이므로, 출력 선 m의 제어율을 식(20)과 (21)로 계산한다. 입력 선 l의 관측률은 식(19)와 같다.

$$C1(m) = C0(l) \quad (20)$$

$$C0(m) = C1(l) \quad (21)$$

XOR 게이트는 입력 선에 나타난 1의 수가 홀수 또는 짝수이냐에 따라 출력 선의 값이 1 또는 0으로 된다. 그리고 한 입력 선의 변화는 나머지 다른 입력 선들의 현재 상태에 간섭을 받지 않고 출력 선에서 항상 관측된다. 이때 입력 선의 수가 n개인 XOR 게이트를 해석하기 위해 n-1개의 XOR2 게이트가 순차적으로 연결된 그림 2의 등가 회로를 이용한다. 그림 2에서 첫번째 게이트에 대한 출력 선의 제어율은 식(22)와 (23)으로 계산되고, 나머지 i번째 게이트에 대한 출력 선의 제어율은 식(24)와 (25)로 계산된다.

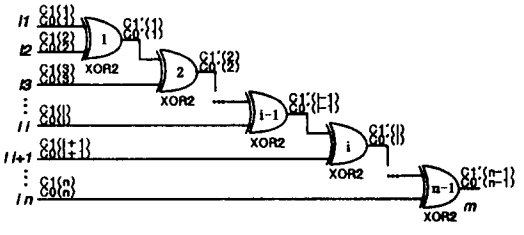


그림 2. 입력이 n개인 XOR의 등가 회로
Fig. 2. Euivalent circuit to n inputs XOR.

$$C1'(1) = (C1(1) \cap C0(2)) \cup (C0(1) \cap C1(2)) \quad (22)$$

$$C0'(1) = (C1(1) \cap C1(2)) \cup (C0(1) \cap C0(2)) \quad (23)$$

$$C1'(i) = (C1'(i-1) \cap C0(i+1)) \cup (C0'(i-1) \cap C1(i+1)) \quad (24)$$

단, $2 \le i \le n-1$

$$C0'(i) = (C1'(i-1) \cap C1(i+1)) \cup (C0'(i-1) \cap C0(i+1)) \quad (25)$$

단, $2 \le i \le n-1$

따라서 입력 선의 수가 n개인 XOR 게이트에 대하여 출력 선을 m이라 하고 i번째 입력 선을 l_i라 하면, 제어율과 관측률을 식(26), (27), 그리고 (28)로 각각 계산한다.

$$C1(m) = C1'(n-1) \quad (26)$$

$$C0(m) = C0'(n-1) \quad (27)$$

$$OB(l_i) = \bigcap_{k=1, k\neq i}^{n} (C1(l_k) + C0(l_k)) * OB(m)$$

단, $1 \le i \le n$ (28)

XNOR 게이트는 XOR 게이트에서 출력만 반대이다. 입력 선의 수가 n개인 XNOR 게이트에 대하여 출력 선을 m이라 하고 i번째 입력 선을 l_i라 할 때, 출력 선에 대한 제어율을 식(29)와 (30)으로 계산한다. 입력 선의 관측률은 식(28)과 같다.

$$C1(m) = C0'(n-1) \quad (29)$$

$$C0(m) = C1'(n-1) \quad (30)$$

FANOUT에서는 입력 선의 값이 출력 선으로 그대로 전달되며 또 입력 선의 값을 임의의 출력 선에서 항상 관측할 수 있다. 이때 FANOUT의 모든 출력 값들이 회로의 최종 출력 쪽으로 진행하면서 다시 재수렴(reconvergent)하지 않는 독립적인 관계라고 가정한다. 따라서 출력 선의 수가 n개인 FANOUT에서 i번째 출력 선을 m_i라 하고 입력 선을 l이라 할 때 제어율과 관측률을 식(31), (32), 그리고 (33)으로 계산한다.

$$C1(m_i) = C1(l) \quad 단,\ 1 \le i \le n \quad (31)$$

$$C0(m_i) = C0(l) \quad 단,\ 1 \le i \le n \quad (32)$$

$$OB(l) = \bigcup_{i=1}^{n} OB(m_i) \quad (33)$$

2. D-타입 플립플롭

D-타입 플립플롭은 입력 선 d와 clk, 그리고 출력 q로 구성되며 입력 선 clk에 상승 신호(rising edge)가 발생할 때 동작한다. 따라서 clk에 상승 신호가 발생할 때, 입력 선 d의 값에 따라 출력 선 q의 값이 변하고

또 입력 선 d의 상태가 출력 선 q에서 관측된다. 그리고 입력 d의 값이 변경(toggle)된 후 clk에 최초로 나타난 상승 신호는 출력 q에서 관측된다. 입력 선 clk에 상승 신호가 발생할 확률 Prob(R)와 입력 d의 값이 변경될 확률 Prob(T)를 식(34)와 (35)로 각각 정의하고, 이들 식을 이용하여 출력 선의 제어율과 입력 선의 관측률을 식(36), (37), (38), 그리고 (39)로 계산한다.

$$Prob(R) = C0(clk) * C1(clk) \tag{34}$$

$$Prob(T) = C1(d) * C0(d) + C0(d) * C1(d) = 2 * C1(d) * C0(d) \tag{35}$$

$$C1(q) = Prob(R) * C1(d) \tag{36}$$

$$C0(q) = Prob(R) * C0(d) \tag{37}$$

$$OB(d) = Prob(R) * OB(q) \tag{38}$$

$$OB(clk) = Prob(T) \tag{39}$$

본 논문의 검증에서 사용하는 ISCAS89[7] 회로의 경우, 그림 3 처럼 모든 D-타입 플립플롭의 clk 선이 하나의 최초 입력(PI: Primary Input)에 연결된다.

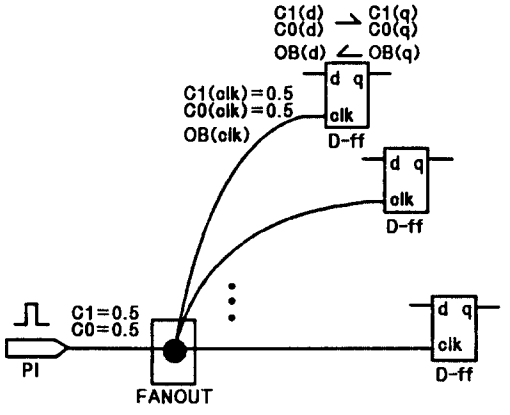


그림 3. 모든 D-타입 플립플롭(D-ff)에 입력되는 공통 clk 신호
Fig. 3. Common clk signal to all D-type flipflop(D-ff).

그림 3과 같은 회로에서 검사 신호를 인가할 때마다 D-타입 플립플롭의 clk에 상승 신호(rising edge)를 발생시키면, 식(34)를 식(34-1)로 나타낼 수 있고 식(36), (37), 그리고 (38)들이 식(36-1), (37-1), 그리고 (38-1)의 간단한 형태로 각각 변환된다.

$$Prob(R) = 1.0 \tag{34-1}$$

$$C1(q) = C1(d) \tag{36-1}$$

$$C0(q) = C0(d) \tag{37-1}$$

$$OB(d) = OB(q) \tag{38-1}$$

Ⅳ. 순차회로에 대한 해석

회로의 게이트(gate)와 네트(net)를 각각 점과 선(line)으로 표현한 위상(topology)을 깊이 우선 탐색(DFS: Depth First Search) 기법으로 검색하여 각 선을 아래와 같이 4 가지로 분류한다. 이때 후진 선에 의해 루프(loop)가 형성되며, 그림 4의 예에서 후진 선 4-2와 5-3에 의해 루프 4-2-3-4와 5-3-4-5가 각각 형성됨을 알 수 있다[8]-[9].

- 가지 선(tree line) : 처음 통과하는 점에 연결되는 선.
- 후진 선(backward line) : 가지 선에 접하고 있는 점에 후방으로 연결되는 선.
- 전진 선(forward line) : 가지 선에 접하고 있는 점에 전방으로 연결되는 선.
- 횡단 선(cross line) : 위의 3가지 종류에 속하지 않는 선.

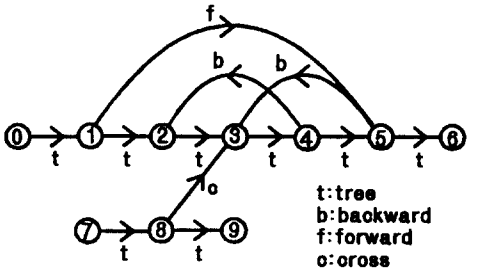


그림 4. 선의 분류
Fig. 4. Classification of lines.

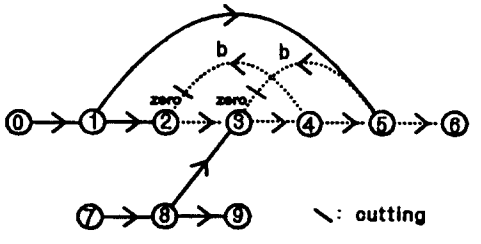


그림 5. 제어율의 오차 확산
Fig. 5. Propagation of invalid controllability.

루프를 구성하는 게이트들에 대하여 필요한 확률을 정확하게 계산하기 위해 일반적으로 다음 2 가지 방법을 사용한다. 첫째, 루프를 구성하는 게이트를 추적하면서 각 선들의 값을 계산한다. 이때 모든 게이트에서

확률값의 변화가 없을 때 까지 추적 및 계산을 계속한다.

```
Procedure Compute_Controllability()
begin
  Build a circuit digraph GRAPH(VERTEX, LINE).
  ITERATION = Cut_Backward_lines(GRAPH)
  T_SORT = Topological_Sorting(VERTEX)
  Initiate_Controllabilities(VERTEX)
  Controllabilities_of_All_Gates(ITERATION, T_SORT)
  return
end
Procedure Controllabilities_of_All_Gates(ITERATION, T_SO
RT)
begin
  repeat
    for every g ∈ T_SORT
      begin
        for every i ∈ {input lines of g}
          Read C1(i), C0(i), and CX(i).
        for every o ∈ {output lines of g}
          Compute new C1(o) and C0(o).
          CX(o) = 1 - C1(o) - C0(o)
      end
  until ITERATION
  return
end
Procedure Initiate_Controllabilities(VERTEX)
begin
  for every g ∈ VERTEX
    for every i ∈ {input lines of g}
      begin
        C1(i) = C0(i) = 0
        CX(i) = 1 - C1(i) - C0(i)
      end
  return
end
Procedure Topological_Sorting(VERTEX)
begin
  Perform topological sorting for all v ∈ VERTEX and make
  a list of sorted vertices, T_SORT.
  return T_SORT
end
Procedure Cut_Backward_lines(GRAPH)
begin
  Search GRAPH for backward lines in depth first
    search(DFS) and make the list of backward lines,
    BACK.
  if BACK == {} then
    return 1
  else
    begin
      for every b ∈ BACK
        Eliminate b from GRAPH.
      return I_LIMIT
    end
end
```

그림 6. 제어율의 계산
Fig. 6. Computation of controllability.

이 방법을 적용하기 위해서는 루프를 구성하는 게이트의 입력 값들이 모두 독립적이라는 가정이 필요하다. 그러므로 한 신호가 서로 다른 경로를 통하여 동일한 루프의 다른 곳으로 연결될 경우에는 사용이 부적당하다. 둘째, 각 루프에 대한 sensitization 확률을 이용한다. 즉, 루프를 구성하는 게이트들의 입력 선 중에서 루프를 구성하는 선들을 제외한 나머지 선들을 모두 찾고, 그 선들의 값이 해당 게이트를 sensitize하는 확률을 먼저 계산한다. 그러나 다수의 후진 선이 서로 간섭을 하면서 루프를 형성하는 경우, 회로 내의 모든 루프를 검색하는데 소요되는 계산 시간 상의 overhead가 크다.

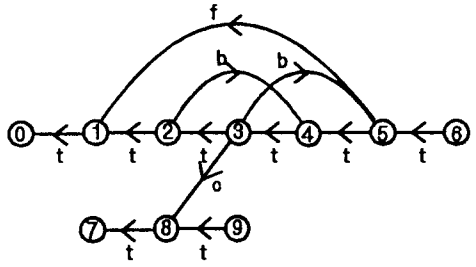


그림 7. 선의 방향을 반대로 수정
Fig. 7. Turning oppositely the direction of lines.

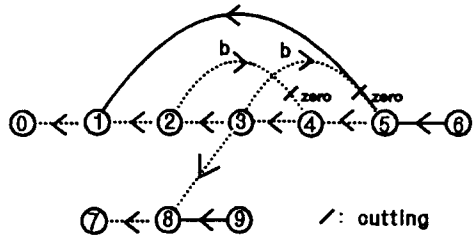


그림 8. 관측률의 오차 확산
Fig. 8. Propagation of invalid observability.

위의 문제점들을 해결하기 위해 본 연구에서는 다음과 같은 계산 방법을 제시한다. 먼저 그림 5의 모든 후진 선에 대하여 사선으로 표시한 부분을 절단한다.

이때 절단된 후진 선(backward line) b에 대하여 최초 제어율을 $C1_0(b)$, $C0_0(b)$, 그리고 $CX_0(b)$라 하면, D-타입 플립플롭의 초기 상태에 의해 확률 값 $CX_0(b)$는 1이 되고, 식(1)에 의해 확률 값 $C1_0(b)$와 $C0_0(b)$는 모두 0이 된다. 이 상태에서 최초 입력(primary input)에서 시작하여 최종 출력(primary output)에 도달하거나 절단된 후진 선을 만날 때 까지, 게이트를 검색하면서 각 선의 제어율을 계산한다. 그러면 절단된 선 b에 대하여 새로운 제어율 $C1_1(b)$, $C0_1(b)$, 그리고 $CX_1(b)$을 얻을 수 있다. 이때 선 b에 대하여 앞서 결정한 초기 값 $C1_0(b)$, $C0_0(b)$, 그리고 $CX_0(b)$들은 검사 신호를 계속 입력할 때의 상태를 반영하지는 않는다. 그러므로 $C1_1(b)$, $C0_1(b)$,

$CX_1(b)$ 그리고 그림 5에서 점선으로 표시된 선들의 제어율에는 오차가 포함되었다. 그리고 $C1_1(b)$, $C0_1(b)$, 그리고 $CX_1(b)$들을 이용하여 동일한 방법으로 각 게이트의 제어율을 다시 계산하면 그 오차가 감소한다. 이런 과정을 반복하면서 제어율을 계산하는 과정을 그림 6에 나타내었다.

```
Procedure Compute_Observability()
begin
  Build a circuit digraph GRAPH(VERTEX, LINE).
  Build a digraph R_GRAPH(R_VERTEX, R_LINE) whose
     edges have opposite directions to GRAPH.
  ITERATION = Cut_Backward_lines(R_GRAPH)
  R_T_SORT = Topological_Sorting(R_VERTEX)
  Initiate_Observabilities(VERTEX)
  Observabilities_of_All_Gates(ITERATION, R_T_SORT)
  return
end
Procedure Observabilities_of_All_Gates(ITERATION, R_T_SO
RT)
begin
  repeat
    for every g ∈ R_T_SORT
      begin
        for every o ∈ {output lines of g}
          Read OB(o).
        for every i ∈ {input lines of g}
          begin
            Read C1(i), C0(i), and CX(i).
            Compute new OB(i).
          end
      end
  until ITERATION
  return
end
Procedure Initiate_Observabilities(VERTEX)
begin
  for every g ∈ VERTEX
    for every o ∈ {output lines of g}
      OB(o) = 0
  return
end
```

그림 9. 관측률의 계산
Fig. 9. Computation of observability.

각 선의 관측률을 계산하기 위해 그림 4의 그래프에서 각 선의 방향을 반대로 하여 그림 7의 그래프를 구성한다. 구한 그래프에서 후진 선을 검색하여 그림 8처럼 사선으로 표시한 부분을 절단한다.

그리고 절단한 곳의 최초 관측률 OB0(b)를 0으로 한 후, 제어율을 계산할 때 적용한 반복법을 동일하게 적용한다. 관측률을 계산하는 과정은 그림 9와 같다.

이때 그림 6의 Cut_Backward_lines()에서 사용되는 반복 회수 I_LIMIT의 최소 값은 실험적으로 결정된다. 회로 s5378에서 여러 가지 반복 회수에 대한 결함 시험도(FC: Fault Coverage)의 변화를 그림 10에 나타내었다.

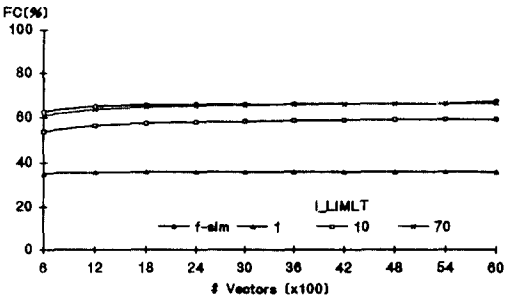


그림 10. 회로 s5378에 대한 반복 회수 I_LIMIT와 결함 시험도(FC)의 관계
Fig. 10. Correlating I_LIMIT with fault coverage(FC) on s5378.

이때 I_LIMIT을 충분히 크게 하면 모든 선들에 대하여 제어율과 관측률이 일정한 수치로 수렴하지만 다음의 이유 때문에 반복 회수를 제한하여야 한다. 즉 D-타입 플립플롭은 clk 선에 상승 신호가 있을 때 동작하며, 루프를 구성하는 게이트들에 대한 입력 선 중 일부가 서로 종속적일 수 있다. I_LIMIT=60일 때 결함 시뮬레이터(fault simulator)의 결과에 가장 근접함이 관측되었다.

V. 검 증

지금까지 설명한 계산식을 이용하여 회로 내의 모든 선에 대한 검출률 D1(l)과 D0(l)를 계산한다. 이들 값은 선 l에서 발생하는 stuck-at-1과 -0의 결함을 검출할 수 있는 확률을 각각 의미한다.

크기가 v인 검사 신호(test vector)를 인가하여 검출률이 d_i인 결함 i를 검출할 확률을 검사성(testability)이라 정의한다. 검사성은 식(40)으로 계산되며 이 식은 v개의 입력으로 결함을 검출하지 못할 확률의 여집합을 의미한다. 그리고 회로에 존재하는 stuck-at 형태의 결함이 모두 k개일 때, 회로에 대한 결함 시험도(FC: Fault Coverage)를 식(41)로 계산한다[15].

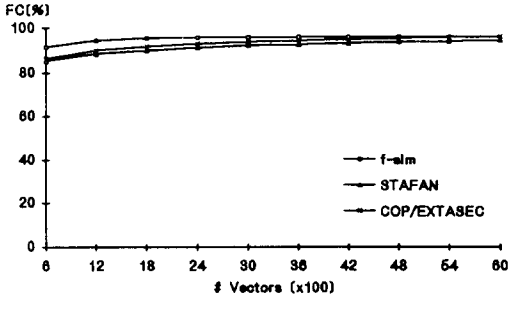
$$t(d_i, v) = 1 - [1 - d_i]^v \tag{40}$$

$$FC = \frac{1}{k}\sum_{i=1}^{k} t(d_i, v) = 1 - \frac{1}{k}\sum_{i=1}^{k}[1 - d_i]^v \tag{41}$$

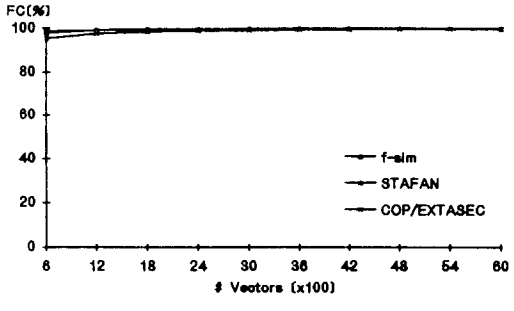
제시한 기법의 유용성을 확인하기 위해 구현한 소프트웨어의 전체 구성을 그림 11에 나타내었다.

```
Procedure EXTASEC()
begin
  Build a circuit digraph GRAPH(VERTEX, LINE).
  Build a digraph R_GRAPH(R_VERTEX, R_LINE) whose
    edges have opposite directions to GRAPH.
  ITERATION = Cut_Backward_lines(GRAPH)
  if ITERATION == I_LIMIT
    ITERATION = Cut_Backward_lines(R_GRAPH)
  Initiate_Controllabilities(VERTEX)
  T_SORT = Topological_Sorting(VERTEX)
  Controllabilities_of_All_Gates(ITERATION, T_SORT)
  Initiate_Observabilities(VERTEX)
  R_T_SORT = Topological_Sorting(R_VERTEX)
  Observabilities_of_All_Gates(ITERATION, R_T_SORT)
  Compute fault coverage.
  return
end
```

그림 11. EXTASEC의 구조
Fig. 11. Structure of EXTASEC.



(a)

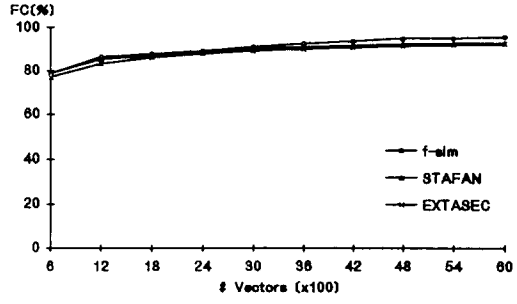


(b)

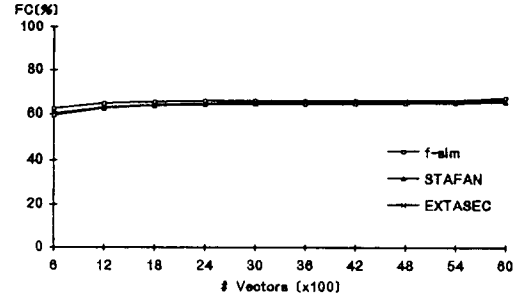
그림 12. 조합회로에 대한 결함 시험도(FC)의 변화
(a) c3540. (b) c5315
Fig. 12. Fault coverage(FC) variations on combinational circuits.
(a) c3540. (b) c5315.

ISCAS85 회로에 EXTASEC을 적용하여 결함 시험도(FC: Fault Coverage)를 계산하였고, STAFAN[5]과 결함 시뮬레이터(f-sim)의 결과들을 그림 12에 함께 나열하였다. 그림에서 c3540 회로보다 c5315 회로가 작은 오차를 나타내고 있으며, 신호(test vector)의 크기가 증가함에 따라 결과가 더욱 정확하다.

ISCAS89 회로에 대한 결과를 그림 13에 나열하였다. 그림에서 알 수 있듯이 회로 s5378의 경우 계산의 정밀도 면에서 EXTASEC이 STAFAN보다 더 정확하다.



(a)



(b)

그림 13. 순차회로에 대한 결함 시험도(FC)의 변화
(a) s1196. (b) s5378
Fig. 13. Fault coverage(FC) variations on sequential circuits.
(a) s1196. (b) s5378.

조합회로와 순차회로에 대한 결함 시험도의 최종 치를 표 1과 2에 나타내었다.

순차회로의 경우 검출되지 않는 결함(redundant fault)들이 존재할 수 있으며, 그것은 회로의 설계 자체에 원인이 있거나 D-타입 플립플롭에 대한 초기 값의 설정이 부적절하기 때문이다[3]. 특히 최초 입력단(primary input)에 근접한 곳에서 재수렴(reconvergent) 현상이 발생하면 회로의 대부분이 영향을 받는다. 표 2에서 이러한 회로들을 '*'로 표시하였으며 이들에 대하여는 I_LIMIT=1을 적용하였다.

표 1. ISCAS85 회로에 대한 결함 시험도의 추정
Table 1. Fault coverage estimates on ISCA S85 circuits.

[%]

CKT	f-sim	STA FAN	EXTAS EC (COP)	CKT	f-sim	STA FAN	EXTAS EC (COP)
c432	98.84	99.38	100.00	c2670	84.01	86.03	87.20
c499	99.19	100.00	100.00	c3540	96.20	94.49	96.27
c880	99.60	99.38	99.63	c5315	99.41	99.95	99.66
c1355	99.70	99.94	98.85	c6288	99.45	99.58	100.00
c1908	99.65	99.45	99.22	c7552	94.30	96.86	95.00

표 2. ISCAS89 회로에 대한 결함 시험도의 추정
Table 2. Fault coverage estimates on ISCA S89 circuits.

[%]

CKT	f-sim	STA FAN	EXTAS EC	CKT	f-sim	STA FAN	EXTAS EC
s27	100.00	98.33	100.00	s526n	9.88	8.66	10.50
*s208	8.71	8.36	8.81	s641	87.55	86.23	87.67
s298	80.36	78.34	85.97	s713	83.38	83.46	88.15
s344	97.31	94.28	96.66	*s838	5.91	5.77	4.63
s349	96.76	93.80	96.68	s958	7.97	8.08	8.29
s382	13.61	11.13	13.63	s1196	95.44	92.63	92.07
s286	71.63	68.35	75.27	s1238	92.20	90.74	87.02
*s420	6.55	6.34	7.16	s5378	67.39	65.52	66.15
s444	12.16	9.87	12.03	s9234	10.02	12.32	12.91
s510	0.00	0.00	0.00	s13207	12.06	11.98	15.32
s526	9.88	8.66	10.50	s15850	35.10	26.99	34.36

* I_LIMIT = 1

EXTASEC의 계산 시간을 STAFAN의 그것과 비교하여 그림 14에 나타내었다. 이것은 6000개의 검사 신호에 대한 계산 시간이다. 내부적으로 결함 배제 시뮬레이션(fault-free simulation)을 수행하는 STAFAN에 비해 수행 시간이 크게 감소하였다.

EXTASEC은 COP와 달리 순차회로에 대한 적용이 가능하고 정밀도 면에서 STAFAN에 근접하며 경우에 따라 더 정확하다. 그리고 다른 기법들에 비하여 계산 시간이 현저하게 작다.

VI. 결 론

제어율과 관측률을 이용하여 디지탈 회로의 검사성을 분석하는 기법 EXTASEC을 제안하였고 그 유용성을 프로그램으로 확인하였다.

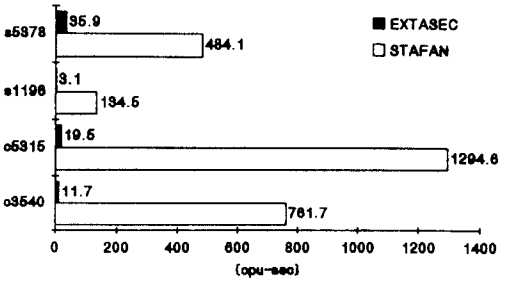


그림 14. 실행 시간의 비교
Fig. 14. Comparison of execution times.

구현한 소프트웨어로 계산한 결함 시험도를 결함 시뮬레이터의 그것에 비교한 결과 참값에 근접하였다. 또한 고속이지만 순차회로에 적용이 곤란한 COP에 반해, EXTASEC은 순차회로의 경우에도 계산의 정밀도 면에서 기존의 다른 기법들에 필적하고 일부 회로에서는 더 정확한 결과를 나타내기도 한다. 회로의 결함 시험도를 계산하기 위해 결함 시뮬레이터나 STAFAN을 사용하면, 회로의 복잡도가 커짐에 따라 메모리의 필요량이 증가하고 계산 시간이 느려진다. 그러나 구현한 소프트웨어는 메모리를 적게 차지하면서 고속으로 결함 시험도를 계산하였다.

EXTASEC을 사용함으로서 순차회로에 대한 결함 해석이 가능하고, 또 그 계산 시간을 현저하게 단축시킬 수 있다. 향후 연구 과제로 첫째, reconvergent fanout의 구조를 알고리즘에 반영하고, 둘째, 순차회로 내부의 초기 값을 분석 조정하는 기능을 추가한다면 더욱 정확한 결함 시험도를 계산할 수 있을 것으로 기대된다.

참 고 문 헌

[1] V.D.Agrawal and S.C.Seth, *Tutorial: Test Generation for VLSI Chips*, Computer Society Press, 1988.

[2] M.Abramovici, M.A.Breuer, and A.D.Friedman, *Digital Systems Testing and Testable Design*, Computer Society Press, 1990.

[3] A.Ghosh, S.Devadas, and A.R.Newton, *Sequential Logic Testing and Verification*, Kluwer Academic Publishers, pp. 11-55, 1992.

[4] F.Brglez, "On Testability Analysis of

Combinational Networks," *Proc. of the Int. Symp. on Circuits and Systems*, Montreal, Canada, pp 221-225, May 1984.

[5] S.K.Jain and V.D.Agrawal, "Statistical Fault Analysis," *IEEE Design & Test of Computers*, Vol. 2, pp. 38-44, Feb. 1985.

[6] R.Kapur, J.Ferguson, and M.Abadir, "Trade-off Analysis of the Effectiveness of Testability Estimators," *Proc. of 2nd European Test Conference*, Munich, pp 341-349, April 1991.

[7] F.Brglez, D.Bryan, and K.Kozminski," Combinational Profiles of Sequential Benchmark Circuits," *Proc. of the Int. Symp. on Circuits and* Systems, Potland, Oregon, May 1989.

[8] T.H.Cormen, C.E.Leiserson, and R.L.Rivest, *Introduction to Algorithms*, The MIT press, 1992.

[9] D.B.Johnson, "Finding All the Elementary Circuits of A Directed Graph," *SIAM J. Comp.*, Vol. 4, pp. 77-84, Mar. 1975.

저 자 소 개

金 信 澤(正會員)

1962년 5월 16일생. 84년 성균관대학교 전기공학과 공학사. 89년 성균관대학교 전기공학과 공학석사. 현재 성균관대학교 전기공학과 박사과정. 84년-86년 육군 통신장교. 89년 금성반도체 연구원. 90년 ~ 91년 삼보컴퓨터 연구원. 주 관심분야는 Interconnection Network, VLSI CAD 등임.

閔 炯 福(正會員) 第 28卷 第 12號 參照.

현재 성균관대학교 전기공학과 교수