특 집

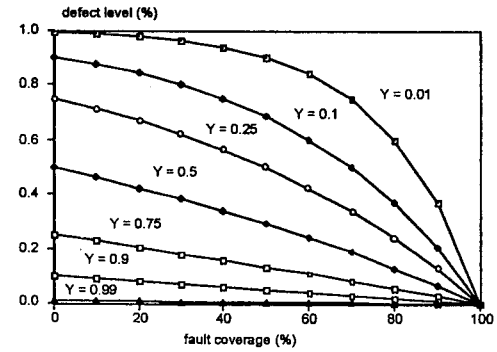
VLSI 회로의 테스트 패턴 생성 기술

朴 恩 世

漢陽大學校 電子工學科

I. 서 론

VLSI 테스트 기술은 불량 반도체 소자를 검출하는 것으로 회로의 집적도가 증가될수록 더욱 철저한 테스트 과정이 요구된다[1,2]. 현재 IBM, NEC, HP 및 AT&T등 외국의 첨단 전자회사에서는 테스트 과정을 통과한 반도체 소자의 품질을 100ppm(part per million), 즉 고장난 부품이 100만개 중에 100개 미만을 허용하는 높은 기준을 정하고 있다. 이러한 품질 기준은 defect level이라고 하여 테스트를 통과한 부품을 출하했을 때 불량품으로 판명되어 반품되는 부품의 비율이다. 테스트를 많이 하면 할수록 고장검출율이 올라가고 아울러 defect level은 낮아지게 되지만 테스트의 비용 증가로 반도체 소자의 가격이 상승된다. 따라서, 제품의 품질과 가격을 최적화하는 효율적인 테스트 패턴 생성은 매우 중요한 문제이다. 고장 검출율, Defect Level 및 수율 간의 관계는 테스트의 효율과 VLSI의 설계 및 제작 비용과 밀접한 관계를 갖고 있어 많은 연구가 수행되었다. 현재 주로 인용되고 있는 defect level 모델은 $DL=1-(\text{수율})^{1-\text{고장검출율}}=1-Y^{(1-T)}$ 과 같다[3, 4]. 그림 1에서는 여러가지 수율 값에 따른 고장검출율과 defect level간의 관계를 보여주는데, 수율이 낮을수록 고장검출율이 높아야 낮은 defect level을 얻을 수 있음을 알 수 있다.



〈그림 1〉 Defect Level, 수율 및 고장검출율의 해석적 관계

전자 회로에 대한 고장은 주로 stuck-at 고장 모델로 해석하여 왔다. stuck-at 고장은 회로 내의 신호선이 0(stuck-at-0) 또는 1(stuck-at-1) 값으로 고착되어 있는 고장의 형태로 1959년 Eldred에 의하여 처음 제안되어[5] datamation이라는 진공관 컴퓨터의 고장을 검출하는데 사용된 이래 현재까지도 주문형 반도체의 대표적인 고장 유형으로 사용되어 왔으며 대부분의 주문형 반도체 제조 회사에서 테스트 패턴을 생성할 때 stuck-at 고장에 대한 검출율로 테스트 패턴의 효율을 표시하고 있다. 그러나 최근 CMOS 회로가 고집적화 됨에 따라 stuck-at 고장 모델만으로는 100~1000ppm 정도의 품질 기준을 만족시키기 어렵다는 결과가 발표되고 있다. 미국의 HP 연구진에 의하여 발표된 바에 의하면 8,500 게이트급의 주문형 반도체에 대하여 실험한 결과[6], 92% 정도의 stuck-at 고장 검출율로는 10,000ppm 정도의 품질을 갖게 된다고 한다. 1000ppm 이내의 최적의 품질을 보장하려면 95% 정도의 stuck-at 고장 검출율 외에 delay 고장(시간지연고장)[7, 8, 9] 및 Iddq current 테스팅[10, 11]이 필요하다는 연구 결과가 발표되었다.

이러한 여러가지 테스트의 문제는 결국 테스트 패턴을 어떻게 효율적으로 생성하겠는가 하는 문제와 직결된다. 테스트 패턴 자동 생성(ATPG : automatic test pattern generation)은 주어진 설계 회로에 대하여 고장 모델을 가정하여 패턴 생성을 자동화시키는 것이다. VLSI 회로에서 stuck-at 고장을 검출하기 위한 테스트 패턴을 생성하는 방법은 크게 세가지로 구분된다.

1) Manual generation : 설계 회로의 동작 검증을 위하여 논리 시뮬레이션에서 사용한 기능 테스트 패턴을 변형시켜 고장 검출을 위한 테스트 패턴으로 사용하는 것으로 일반적으로 높은 고장 검출율을 얻기는 어려우나 회로의 동작 속도로 테스트를 실시할 경우 stuck-at 고장 외에도 delay 고장 등을 검출할 수 있는 장점이 있다.

2) Pseudo-random pattern generation : 고장 모델이나 회로의 구조에 비교적 무관하게 소프트웨어 또는 하드웨어적인 방법으로 임의의 테스트 패턴을 생성하는 것으로 패턴 생성이 용이하고 built-in self testing등의 방법에 적용될 수 있다.

3) Algorithmic(또는 deterministic) test generation : 회로도와 사용된 소자의 동작 라이브러리를 이용하여 소프트웨어 방법으로 테스트 패턴을 자동 생성하는 것으로 주어진 고장 모델에 대하여 비교적 높은 검출율을 얻을 수 있다.

본 고에서는 회로의 고장을 검출하기 위한 테스트 패턴 생성 방법 중에서 algorithmic 테스트 패턴 생성에 대하여 검토한다. 테스트 패턴 생성 방법은 회로의 형태 및 고장 모델 등에 의해 여러가지로 분류되 수 있으므로 우선 가장 기본이 되는 조합 회로 게이트 레벨의 stuck-at 고장을 검출하기 위한 테스트 패턴 생성 알고리즘을 대상으로 기본적인 사항을 소개한다. 또한, 회로 형태 및 고장 모델에 따른 기본 알고리즘의 변형 등에 대하여 개략적으로 기술하기로 한다. 아울러 앞으로의 개발 방향 및 종합적인 테스트 패턴 생성 소프트웨어의 기본 조건에 대하여도 고찰한다.

II. Stuck-at 고장 검출율 위한 게이트 레벨 ATPG 방식

1. 기본 개념

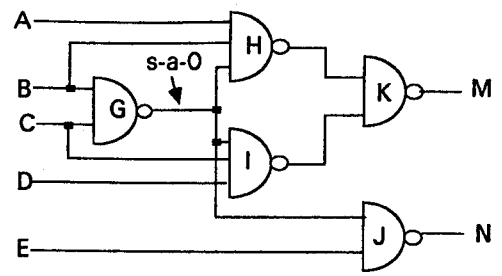
실용적인 ATPG 알고리즘은 회로의 소자의 종류와 신호 연결 정보를 바탕으로 path tracing 기법을 이용하여 주어진 고장에 대한 테스트 패턴을 생성하는데, 크게 fault excitation과 fault propagation 단계가 필요하다.

1) Fault excitation 단계 : fault excitation 단계에서는 고장이 존재하는 신호선에 고장이 있을 경우와 없을 경우에 대한 차이를 발생시켜 고장이 유발되도록 하는 과정이다. 예를 들어 그림 2에서 게이트 G의 출력 신호선에 stuck-at-0 고장이 존재한다고 가정하자. Fault excitation은 고장이 없을 때는 논리값 1, 없을 때는 논리값 0이 발생하도록 주입력 (primary input) 신호에 논리값을 결정하는 것이다. 따라서 G의 출력 신호에 논리값 1

을 배정하는 것을 objective로 설정한다. 즉, objective는 신호선과 그에 대한 논리값으로 구성되며 그림 2의 경우에는(G의 출력 신호, 논리값 1)이 된다. 그 다음, objective를 성취하기 위한 주입력 신호의 논리값을 정하여 주어야 하며 이러한 과정을 logic value(또는 line) justification이라 한다. 그 결과, 주입력 A와 B에 각각 논리값 1을 배정하게 된다.

2) Fault propagation 단계 : 일단 고장이 유발되면 그 차이를 회로의 외부에서 관찰할 수 있도록 하기 위하여 fault propagation이 필요하다. 그림 2에서 게이트 G의 출력 신호의 고장 반응을 회로의 주출력단 (primary output) M 또는 N으로 전파시키기 위하여 G에서 M 또는 N까지의 경로 중 적어도 한 개 이상의 경로를 통하여 고장 효과(good 1/faulty 0)를 주출력 단까지 전달하기 위하여 한다. 따라서, 고장 전달 경로상에 있는 모든 게이트의 입력 신호 중에서 경로상에 있지 않은 입력 신호의 논리값을 고장 효과가 전파될 수 있도록 하는 objective를 설정하고 결정하고 동시에 line justification을 통하여 objective를 달성하기 위한 주입력 신호의 논리값을 결정하여야 한다.이러한 과정을 path sensitization이라고 하며 single path sensitization과 multiple path sensitization이 있다. 예를 들어 주입력 신호(A,B,C,D,E)=(0,1,1,0,1)은 경로 G, J, N을 통하여 고장 효과를 주출력단 N으로 전달시키는 single path sensitization을 하게 된다. 한편, 주입력 신호 (A,B,C,D,E)=(1,1,1,1,0)은 경로 G, H, K, M과 G, I, K, M으로 동시에 고장 효과를 주출력단 N으로 전달시키는 multiple path sensitization을 하게 된다. 경우에 따라서는 고장효과가 반드시 multiple path sensitization으로 주출력 단에 전달되기도 한다.

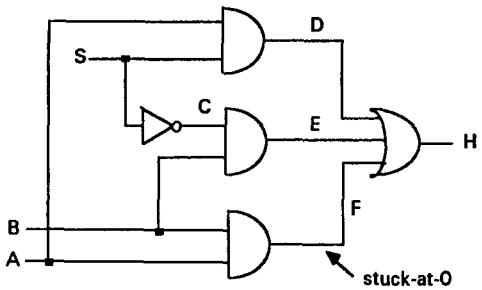
테스트 생성 과정은 fault excitation과 propagation을 동시에 만족시키기 위한 주입력단의 논리값을 결정하게 되는데 이 과정에서 결국 테스트 생성 문제는 space search 문제가 된다. 따라서 테스트 패턴 생성 알고리즘은 search space를 어떤 방식으로 탐색하는가에 따라 효율성이 결정되게 되며 탐색 과정에서 설정된 objective에 대한 decision



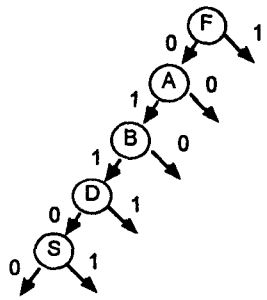
〈그림 2〉 Fault excitation 및 propagation 예

making과 그에 따른 justification 과정을 fault excitation과 propagation이 동시에 만족될 때까지 반복하여 필요한 주입력단의 논리값을 결정하게 된다. 탐색 과정 중의 decision은 항상 justification이 필요한데 경우에 따라서는 decision을 만족시킬 수 없는 경우가 발생할 수 있다. 그림 3에서 신호선 F에 stuck-at-0 고장이 있다고 가정하면, 우선 고장을 유발시키기 위하여 F에 논리값 1을 배정하고 그에 대한 justification으로 주입력 신호 A, B을 각각 논리값 1로 결정하게 된다. 그 다음 고장 효과를 주출력 H에 전달 시키려면 신호선 D와 E를 동시에 논리값 0으로 결정하여야만 한다. 이때 신호선 D에 논리값 0을 justification 하려면 주입력 신호 D에 논리값 0을 가하여야 한다. 그러나 D에 논리값 0을 배정하기 위하여는 신호선 C를 논리값 1로 정하여 주어야 하므로 그 결과 신호선 E를 1로 만들게 되어 신호선 F에 있는 고장 효과가 주출력단 H로 전달되려는 것을 막게 된다. 이러한 현상을 conflict가 발생하였다고 하며 그에 대한 해결을 위하여는 가장 최근의 decision을 취소하여야 한다. 즉, 신호선 D에 논리값 0을 배정하려는 결정을 취소하여야 하므로 결국 신호선 D에 논리값 1을 배정하려는 decision을 하게 된다. 그러나 이 결정 역시 고장 효과의 전달을 봉쇄하게 되므로 잘못된 decision임을 알 수 있다. 따라서 그 이전의 decision에 대한 취소를 하여야 한다. 이러한 과정을 backtrack이라고 한다. 이 과정을 도식적으로 나타내면 그림 4와 같다. 결국 모든 decision을 취소하게 되며 신호선 F의 stuck-at-0 고장에 대한 테스트 패턴 생성이 불가능을 함을 알 수 있게 된다. 이러한 고장을 redun-

dant 고장이라고 하며 회로의 동작에 영향을 미치지 않는 고장임을 알 수 있다. 따라서 그림 3에서 신호선 F를 전체 회로에서 삭제하더라도 회로 동작에 지장이 없게 된다.



〈그림 3〉 Redundant 고장과 backtrack



〈그림 4〉 Decision tree backtrack

테스트 생성 알고리즘은 탐색 과정이 테스트 패턴을 생성하거나 고장이 redundant함을 증명할 수 있을 때 까지 지속되므로 과도한 계산 시간이 걸릴 수 있다. 따라서 컴퓨터에서 테스트 생성을 할 때 일정한 시간이 지나거나 backtrack 수를 제한하여 그 범위 안에서 패턴 생성을 시도하고 만약 테스트 패턴 생성 작업이 완료되지 않으면 패턴 생성을 중단하게 된다. 그러한 고장을 aborted fault라고 부르게 되며 redundant 고장인지 또는 테스트가 가능한 고장인지는 판단 불능 상태가 된다. 조합 회로의 경우 주입력의 수가 N이면 search space의 크기는 2^N 이상이 되므로 N의 값이 크면 search space를 모두 탐색하기는 사실상 불가능한 경우가 발생할 수 있다. 이러한 이유로 테스트 패턴 생성 알고리즘의 복잡도는 NP-complete임이 증명되어 있다[12].

2. 조합 회로용 ATPG 알고리즘의 비교

전자 회로의 설계 형태는 순차 회로(sequential circuit)와 조합 회로(combinational circuit)로 구분되는데, 순차 회로는 테스트하기가 매우 어려워 고장 검출율이 높은 테스트 패턴을 생성하기 어렵다. 한편, 조합 회로는 최근 활발한 연구 개발로 비교적 높은 고장 검출율을 얻을 수 있는 테스트 패턴 생성이 가능하다. 따라서, 주문형 반도체의 효율적인 테스트를 위하여는 structured design-for-test 기법인 scan-path 및 LSSD등의 스캔 설계 방식이 주로 사용되고 있다[13, 14]. 따라서 여기서는 조합 회로의 ATPG 알고리즘에 대하여 간략하게 비교 분석하기로 한다. space search 방식에 따라 구분되는 대표적인 알고리즘은 D-algorithm[15], PODEM[16]과 FAN 알고리즘[17]이 있다.

1) **D-algorithm** : D-algorithm은 Roth에 의하여 발표되었으며 테스트 패턴 생성 문제에 대한 첫번 째 알고리즘이다. D-algorithm은 설정된 objective에 대하여 회로 내의 모든 내부 게이트의 출력 신호선을 search space로 하여 논리값을 배정하고 그에 대한 forward 및 backward implication을 수행한다. Forward implication은 시뮬레이션과 같이 배정된 신호에 대한 회로 동작 반응을 결정하는 것이며 backward implication은 line justification과 같은 의미로 objective의 달성을 위하여 주입력단을 향하여 새로운 decision을 내리고 그에 따른 justification을 하는 것이다. D-algorithm의 단점은 하나의 decision을 내린 후 그에 대한 justification을 끝내지 않은 채 또 다른 decision을 내리게 되어 line justification과 implication시 conflict의 발생 소지가 많아 backtrack의 수가 증가될 수 있어 비효율적이다. 또한, path senistization에서 single 및 multiple path sensitization을 각각 분리하여 수행하는 단점이 있다. 이와 같은 path sensitization시의 문제점을 해결하기 위하여 multiple logic value를 채용한 9-valued D-algorithm이 발표된 바 있다[18].

2) PODEM algorithm : PODEM(Path-Oriented DEcision Making) 알고리즘은 Goel에 의하여 제안된 알고리즘으로 D-algorithm과 달리 search space를 주입력 신호에 국한한다. 설정된 objective에 대한 line justification시 회로의 내부 신호에는 논리값을 배정하지 않고 소자의 극성을 이용하여 path tracing을 주입력단까지 한 뒤 주입력에 비로서 논리값을 배정하게 된다. 따라서 PODEM에서는 forward implication만을 수행하게 된다. 또한, path sensitization시 single 및 multiple path에 대한 구분이 없이 implicit space enumeration 기법을 수행하게 되므로 D-algorithm에 비하여 일반적으로 효과적이다. 또한, 한 번에 하나의 decision을 내리고 그에 따른 objective를 설정 후 justification을 하게 되며 여러 개의 decision을 동시에 내리지 않아 conflict의 발생 소지가 적다. 그림 5는 PODEM 알고리즘의 기본 구조를 보여 준다.

```
PODEM(input : fault under test, output : test)
{
    for each signal,
        initialize with the logic value X
    while(status != TEST_ FOUND && status != TEST_ABORTED) {
        objective=get_objective0; /*objective 설정 */
        if objective is empty due to a conflict
            status=backtrack0; /* if backtrack limit is reached */
                               /* backtrack0 returns TEST_ABORTED */
        else {
            pi_value=justification(objective);
            if(status=x_path_check0)==BLOCKED)
               status=backtrack0; /* fault propagation is not possible with current */
                        /* PI values, so backtrack the PI decision tree */
            else if(status==FAULT_PROPAGATED_TO_PO)
                 status=TEST_FOUND;
          }
      }
  } /* end of test generation */
```

〈그림 5〉 PODEM 알고리즘의 구조

3) FAN algorithm : FAN 알고리즘은 Fujiwara에 의하여 제안되었으며 현재 가장 많이 사용되고 있다. FAN 알고리즘은 D-algorithm과 PODEM의 장점을 최대한 살린 방법으로 테스트 패턴 생성 문제가 주로(reconvergent) fanout stem에 의하여 어려워 지는 것에 착안하여 fanout stem 신호에 배정하는 논리값 선정 시 효율적인 방법을 도입한 것이다. 또한, search space를 줄이기 위하여 headline 개념을 도입하고 있는데, headline이란 headline에서 주입력 단까지의 cone of influence의 내부에 fanout이 없는 일종의 tree의 루트를 의미하므로 line justification시 주입력단에 논리값을 결정하지 않고 headline에 논리값을 결정하게 되면 headline의 논리값은 fault excitation과 propagation을 마친 후 결정할 수 있어 결과적으로 search space를 줄이는 효과가 있게 된다.그러나 실용적인 회로에서 주입력 신호가 아닌 headline은 거의 없어 headline 개념의 도입에 의한 효과는 그다지 크지 않다. 또한, D-algorithm과 같이 회로 내부에서 forward 및 back-

ward implication을 수행하게 되어 conflict의 조기 발견이 가능하다.

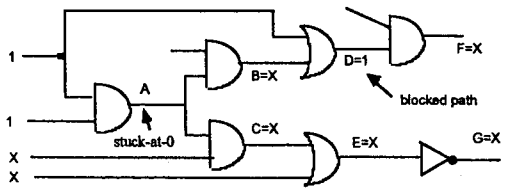
이상으로 주요 테스트 패턴 생성 알고리즘에 대한 검토를 하였다. 이외에도 여러가지 다른 알고리즘이 발표되었으나[19, 20], PODEM 또는 FAN 알고리즘과 비슷한 space search 방식을 도입한 것으로 다른 점은 효율적인 heuristic을 적용하였다는 것이다.

3. 효율적인 탐색 Heuristic

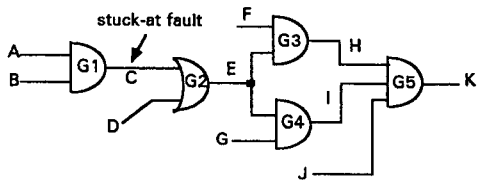
일반적으로 테스트 패턴 생성 문제는 redundant 고장의 확인이나 특별히 어려운 고장(hard fault)에 대한 테스트 생성에 거의 대부분의 시간을 소모한다. 따라서, 테스트 생성을 위한 탐색 과정 중에서 반드시 만족시켜야 할 조건(mandatory condition)을 미리 알고 있으면 전체 search space를 대폭 감소시킬 수 있어 효율적인 알고리즘의 설계가 가능하다. 본 절에서는 대부분의 테스트 생성 알고리즘에 적용이 가능한 효율적인 heuristic에 대하여 소개한다.

1) x-path check(propagation path che-ck) : PODEM 알고리즘에서 처음 사용된 방법으로 주입력단에 논리값을 배정하고 forward implication을 수행하고 난 뒤에 고장 지점에서 주출력 단까지의 path sensitization이 가능한지 여부를 조사한다. 고장 전달 경로상에 있는 신호선이 논리값 X 이외에 0 또는 1의 값이 있으면 그 경로로는 고장이 전달될 수 없음을 알 수 있다. 예를 들어, 그림 6에서 fault excitation이 끝난 후 고장 지점 A에서 주출력단 F와 G까지의 경로에서 신호선 D가 논리값 1을 갖게 되므로 고장 효과가 D를 통과할 수 없음을 알 수 있다. 따라서, 고장 효과는 A-C-E-G를 통과하여야 하므로 그에 따른 objective의 설정을 하여야 한다. 만약, 모든 경로가 막혀 있다면 backtrack을 하여 최근에 결정된 주입력단의 논리값을 바꾸어 주어야 한다.

2) Dominator : Dominator는 고장 전달 경로에서 고장 효과가 반드시 통과되어야 할 게이트를 의미한다. 예를 들어, 그림 7에서 신호선 C에 stuck-at 고장이 있다고 가정하면 dominator는 G1과 G5이며 이들 게이트의 입력 신호 중에서 고장 경로상에 있지 않은 입력 신호는 반드시 고장 효과가 전달될 수 있도록 논리값이 결정되어야 한다. 따라서, 신호선 D에는 논리값 0, 신호선 J에는 논리값 1이 배정되어야 한다. Dominator 개념은 TOPS에서 제안되었으며[21] redundant 고장 검출 효과가 크다. 회로상에서 graph-theoretic 방법으로 dominator를 찾는 알고리즘은 Tarjan[22]에 의하여 발표되었다.

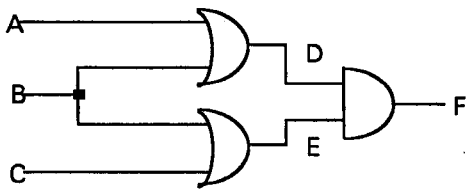


〈그림 6〉 X-path check



〈그림 7〉 Dominator에 의한 mandatory assignment

3) Learning : Learning은 SOCRATES 테스트 시스템[19] 에서 처음 제안된 방식으로 특히 어려운 고장 검출에 큰 효과가 있다. 이 방식은(P → Q) ↔ ((NOT Q) → (NOT P))와 같은 contrapositive logical identity를 이용하여 회로 내에서 어떤 특정 신호선이 특정 논리값을 갖기 위한 필요조건을 미리 구하여 놓고 line justification시에 활용하게 된다. 예를 들어, 그림 8과 같이 신호선 B가 논리값 1을 갖게 되면 신호선 F가 논리값 1로 결정되므로 F가 논리값 0을 가지려면 반드시 B는 논리값 0을 가져야 하는 필요 조건을 만족하여야 한다. 따라서 line justification시에 F에 논리값 0을 배정하려면 먼저 B에 0을 배정하려는 시도가 있게 되어 conflict의 발생을 방지할 수 있다.



〈그림 8〉 Learning 예제 회로

이밖에도 여러가지 효율적인 heuristics가 있다. 예를 들어, unique sensitization, dynamic learning, dynamic unique sensitization 등이 있고[19] 최근에는 recursive learning 방식이 제안되었다[23].

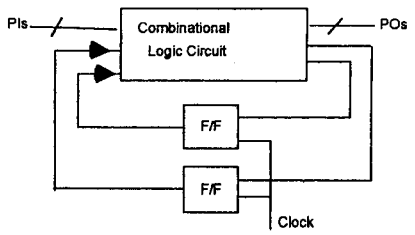
III. 그 밖의 테스트 패턴 생성 문제들

1. 순차 회로의 테스트 패턴 생성

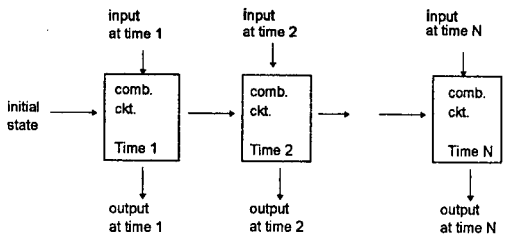
순차 회로는 그림 9와 같이 플립플롭과 같은 메모리 소자를 포함하고 있어 조합 회로용 테스트 알고리즘의 적용이 안된다. 일반적으로 순차 회로의 테스트 방식은 순차 회로 중의 조합 회로 부분을 time frame별로 복사하여 조합 회로용 테스트 패턴 생성 알고리즘을 적용하는데 일반적으로 search space의 증가로 고장검출율이 높은 테스트 생성이 어렵다. 현재까지 발표된 순차 회로용 테스트 패턴 생성 알고리즘으로는 BACK 알고리즘[24], HITEC[25] 등이 있으나 순수한 순차 회로에는 효과가 적어 partial scan 설계 방식과 병행하여 사용되고 있다[26]. 또한 최근에는 boolean difference와 BDD (binary decision diagram)[27] 등을 이용하여 순차 회로의 테스트 문제를 해결하려는 시도도 있었으나 과도한 메모리 사용으로 인하여 비교적 크기가 작은 회로(1000~2000 게이트급)에만 적용이 가능한 상태이다.

2. 스위치 및 Register-Transfer Level 테스트 생성

VLSI 회로의 테스트 패턴 생성 알고리즘은 설계된 회로 형태와 회로 표현 양식, 그리고 고장 모델



(a) Huffman model



(b) iterative network

〈그림 9〉 순차 회로의 테스트 생성

에 따라 기능상의 차이가 있다. 회로 표현 양식에 따라, 게이트 레벨 및 스위치 레벨, 그리고 behavioral-level 회로를 처리할 수 있는 테스트 패턴 생성 알고리즘 및 소프트웨어 개발이 필요한데, 특히 최근의 설계는 주로 게이트 레벨과 behavioral-level 표현을 복합적으로 사용하기 때문에 종래의 테스트 소프트웨어로는 모델링의 한계가 있어 새로운 구조적인 변환이 필요하다. 이러한 추세는 회로의 고집적화에 따라 주문형 반도체 라이브러리 중 매크로 라이브러리의 효율적인 모델링 기술을 요하고 있어 앞으로의 테스트 패턴 자동 생성 소프트웨어 개발에 있어 중요한 관건이 될 것이다.

스위치 레벨 회로에 대한 테스트 소프트웨어는 일반적으로 MPU 및 MCU와 같이 대량 생산되는 완전 주문형 반도체 설계에서 주로 사용되는데, 스위치 레벨 회로가 매우 다양하기 때문에 테스트 알고리즘 개발에 어려운 점이 많다. 특히, 스위치 레벨 회로에는 static CMOS 회로뿐만 아니라 dynamic CMOS 회로 또는 pass-transistor 회로에 대하여 신호강 도를 고려한 테스트 패턴 생성이 필요한데 현재까지 제안된 방법에서는 신호 강도를

무시하거나 static CMOS 회로의 경우만을 처리하는 것이 고작이었다[28, 29]. 최근 신호 강도를 일부 고려하고 설계 형태에 비교적 제약이 없는 효율적인 테스트 패턴 생성 알고리즘이 발표된 바 있다[30]. 앞으로의 완전 주문형 반도체 설계나 저전력 회로 설계에서는 pass transistor 회로와 같은 스위치 레벨 회로가 게이트 레벨 회로와 혼용되어 mixed-level로 많이 사용될 것으로 전망되고 있어 보다 효율적인 알고리즘의 개발이 필요하다. 그리고 스위치 레벨 고장 모델로 한동안 연구가 많이 수행되었던 stuck-open 고장의 경우, 실용성과 효율성이 크게 떨어져 최근에는 별다른 연구가 없으며 실제적으로도 이용되지 않고 있는 실정이다[31]. 그대신 delay 고장에 대한 연구가 활발하게 진행되고 있고 대부분의 stuck-open 고장이 iddq testing이나 delay 테스트로 검출이 가능한 것으로 알려져 있다[6, 7, 8, 9]. 따라서, 일부 상용 테스트 소프트웨어 시스템에서도 delay 고장용 테스트 패턴 생성을 하고 있는 실정이다.

한편, register-transfer 레벨의 테스트 패턴 생성은 주로 마이크로프로세서용 테스팅에 응용되어 왔다[32, 33]. RT-level 구조와 instruction set을 이용하여 register decoding, instruction decoding, data transfer등의 기능에 대한 테스트 패턴을 생성하여 왔지만 아직까지는 추상적인 모델의 사용으로 실제 회로에서 적용하기 어렵고 주로 4-bit 또는 8-bit 마이크로프로세서에 적용되었다. 특히, ALU와 같은 data manipulation 부분에 대한 고장 검출이 어려운 단점이 있다. 그러나 앞으로의 설계가 수백만 게이트급까지 증가할 것이므로 모든 회로를 게이트 레벨에서 처리하기는 거의 불가능할 것으로 예상되므로 많은 연구가 있어야 할 것이다.

IV. 국내외 기술 개발 현황 분석

국내에서 현재까지 개발된 상용 테스트 패턴 자동 생성 소프트웨어는 없었으며 일부 대학에서 연구 활동이 있었지만 실용화되기에는 성능과 기능면에서 적합하지 않았다. 실용적인 테스트 소프트웨어를 개발하기 위하여는 학문적인 이론에 근거한 테스트 패턴 생성 알고리즘 기술과 사용자의 편의와 설계 흐름상의 다른 CAD 및 테스트 장비와의 연계 사용을 돕는 각종 user interface (Motif 사용한 GUI 및 data formatter등) 소프트웨어 기술의 결합이 필요하다. 또한 VHDL을 이용한 논리 최적화 소프트웨어와 연동하여 스캔 설계에 따른 회로 면적 및 동작 속도 감소를 최소화하는 테스트 합성 기술과의 접목 방식들이 연구 개발되어야 한다. 실용적인 테스트 소프트웨어를 개발하기 위하여 필요한 기술들을 다음과 같이 요약하였다.

- o Scan-based sequential circuit용 테스트 패턴 자동 생성 소프트웨어
- o Fault model : stuck-at 고장, delay 고장, iddq current testing 기능
- o Scan Design Rule Check 기능 : scan-path, LSSD 및 partial scan 설계 규칙 검증
- o 고장시뮬레이숀 기능 : 고장 검출 및 진단, 테스트 패턴 compaction 기능
- o 논리 합성 및 최적화 도구와의 표준 회로양식을 통한 연계 사용 기능
- o 회로 모델
  - Netlist format : 국제 표준 회로 양식 인식 (Verilog, EDIF, VHDL)
  - ASIC library를 위한 계층적 구조의 모델링 기능
  - Pure gate-level 및 tri-state device 해석
  - Dynamic 또는 static switch-level 회로 해석 기능(신호 강도 해석 기능)
  - Non-scan element 및 ROM, RAM 등 behavioural-level 모델 해석 기능
- o 테스트 장비와의 interface를 위한 stimulus file format 생성
- o Graphical User Interface 기능(MOTIF standard)

V. 결 론

본 문에서는 회로의 고장을 검출하기 위한 테스트 패턴 생성 방법에 대하여 기술하였다. 조합 회로 게이트 레벨의 stuck-at 고장을 검출하기 위한 테스트 패턴 생성 알고리즘을 중심으로 기본 개념과 각종 회로 형태 및 고장 모델에 따른 기본 알고리즘의 변형등에 대하여 검토하였다. 아울러 앞으로의 개발 방향 및 종합적인 테스트 패턴 생성 소프트웨어의 기본 조건에 대하여도 고찰하였다. 국내의 전자 통신 관련업계에서 최근 HDTV, ATM 및 멀티미디어 관련 주문형 반도체 설계에 박차를 가하고 있고 국내 대표적인 반도체 업계들이 2000년 무렵에는 주문형 반도체의 예상 매출액이 50억 $에 달할 것으로 전망되고 있다. 이러한 고속 고집적 주문형 반도체 소자의 개발에는 반드시 테스트를 고려한 설계와 테스트 소프트웨어의 활용이 있어야 하므로 테스트 분야의 연구 개발이 보다 활성화되어야 하겠다.

참 고 문 헌

[1] T. Williams, *VLSI Testing*, North-Holland Publishers, 1986.

[2] A. Miczo, *Digital Logic Testing and Simulation*, Harper & Row, Publishers, 1986.

[3] T. W. Williams and N. C. Brown, "Defect Level as a Function of Fault Coverage," *IEEE Trans. on Comput.*, Vol.C-32, No.12, pp.987-988, 1981.

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

[5] R. D. Eldred, "Test Routines Based on Symbolic Logic Statements," *Journal of ACM*, vol. 6, no.1, pp.33-36, 1959.

[6] P. C. Maxwell, R. C. Aitken, V. Johansen, and I. Chiang, "The Effectiveness of IDDQ, Functional and Scan Tests : How Much Fault Coverages Do We Need?", *IEEE Proc. Int. Test Conf.*, pp.168-177, 1992.

[7] J. A. Waicukauski, E. Lindbloom, B. K. Rosen, and V. S. Iyengar, "Transition Fault Simulation," *IEEE Design and Test*, pp.32-38, April, 1987.

[8] E. S. Park and M. R. Mercer, "An Efficient Delay Test Generation System for Combinational Logic Circuits," *IEEE Trans. on Computer-Aided Design*, Vol.11, No.7, pp.926-938, July 1992.

[9] B. K. Rosen, V. S. Iyengar, and I. Spillinger, "Delay Test Generation 2 -- Algebra and Algorithms," *IEEE Proc. Int. Test Conf.*, pp. 867-874, 1988.

[10] J. M. Soden and C. F. Hawkins, "Electrical Properties and Detection Methods for CMOS IC Defects," *IEEE Proc. European Test Conf.*, pp.159-167, 1989.

[11] T. Storey, W. Maly, J. Andrews, and M. Miske, "Stuck Fault and Current Test Comparison Using CMOS Chip Test," *IEEE Proc. Int. Test Conf.*, pp.311-318, 1991.

[12] O. H. Ibarri and S. K. Sahni, "Polynomially Complete Fault Detection Problems," *IEEE Trans. on Comput.*, vol.C-24, pp.242-249, 1985.

[13] S. Funatsu, N. Wakatsuki, and T. Arima, "Test Generation Systems in Japan," *ACM/IEEE Proc. Design Automation Symp.*, pp.114-122, 1975.

[14] E. B. Eichelberger and T. W. Williams, "A Logic Design Structure for LSI Testabili-

ty," *ACM/IEEE Proc. Design Automation Conf.*, pp.462-468, 1977.

[15] J. P. Roth, "Diagnosis of Automata Failures : A Calculus and A Method," *IBM Journal of Research and Development*, vol. 10, no.4, pp.278-291, 1966.

[16] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Trans. on Comput.*, vol.C-30, no.3, pp.215-222, 1981.

[17] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms," *IEEE Trans. on Comput.*, vol.C-30, pp.215-222, 1983.

[18] P. Muth, "A Nine-Valued Circuit Model for Test Generation," *IEEE Trans. on Comput.*, vol.C-25, no.6, pp.630-636, 1976.

[19] M. H. Schulz and E. Auth, "Improved Deterministic Test Pattern Generation with Applications to Redundancy Identification," *IEEE Trans. on Computer-Aided Design*, vol.8, no.7, pp.811-816, 1989.

[20] E. S. Park, "A Pragmatic Test Pattern Generation System for Scan-Designed Circuits with Logic Value Constraints," *IEEE Asian Test Symposium*, November 1993.

[21] T. Kirkland and M. R. Mercer, "A Topological Search Algorithm for ATPG," *ACM/IEEE Proc. Design Automation Conf.*, pp.502-508, 1987.

[22] R. E. Tarjan, "Finding Dominators in Directed Graphs," *Proc. 7th Annual Princeton Conf. on Information Science and Systems*, pp.414-418, 1973.

[23] W. Kunz and D. Pradhan, "Recursive Learning : An Attractive Alternative to the Decision Tree for Test Generation in Digital Circuits," *IEEE Proc. Int. Test Conf.*, pp.816-825, 1992.

[24] W.-T. Cheng, "The BACK Algorithm for Sequential Test Generation," *IEEE Proc. Int. Conf. Computer Design*, pp.66-69, 1988.

[25] T. Niermann and J. Patel, "HITEC : A Test Generation Packages for Sequential Circuits," *IEEE Proc. European conf. on Design Automation Conf.*, pp.214-218, 1991.

[26] K.T. Cheng and V. D. Agrawal, "An economical scan design for sequential logic test generation," *IEEE Proc. Int. Symp. on Fault Tolerant Computing*, pp.28-35, 1989.

[27] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Trans. on Comput.*, vol.C-35, no.8, pp.677-691, 1986.

[28] R. E. Bryant, "A Survey of Switch-Level Algorithms," *IEEE Design and Test of Computers*, pp.26-39, August 1987.

[29] K. J. Lee, C. A. Njinda, and M. A. Breuer, "SWiTEST : A Switch Level Test Generation for CMOS Combinational Circuits," *ACM/IEEE Proc. Design Automation Conf.*, pp.26-29, 1992.

[30] E. S. Park and M. R. Mercer, "Switch-level ATPG using Constraint-Guided Line Justification," *IEEE Proc. Int. Test Conf.*, pp.616-625, 1993.

[31] R. L. Wadsack, "Fault Modeling and Logic Simulation of CMOS and MOS Integrated Circuits," *Bell System Tech. Journal*, Vol.57, pp.1449-1488,1978.

[32] S. M. Thatte and J. A. Abraham, "Test Generation for General Microprocessor Architecture," *IEEE Fault Tolerant Computing Symp.*, pp.203-210, 1979.

[33] M. S. Abdir and H. K. Regbati, "Functional Test Generation for LSI Circuits Described by Binary Decision Diagram," *IEEE Proc. Int. Test Conf.*, pp.493-492, 1985.

저 자 소 개

朴 恩 世

1957年 11月 11日生
1980年 2月 서울대학교 전기공학과(학사)
1982年 2月 한국과학기술원 전기 및 전자공학과(석사)
1989年 8月 The University of Texas at Austin(박사)

1982年 3月～1995年 2月 한국전자통신연구소 책임연구원
1989年 9月～1990年 10月 미국 Mentor Graphics사 연구원
1990年 10月～1991年 11月 미국 Motorola사 연구원
1995年 3月～현재 한양대학교 전자공학과 조교수

주관심분야 : VLSI CAD 및 Testing