논리함수처리에 의한 부분스캔순차회로의 테스트생성

최 호 용†

요 약

본 논문은 IPMT법에 부분스캔설계 방법을 적용하여, IPMT법의 적용 한계를 개선한 순차회로의 테스트생성법에 관해 기술한다. IPMT법에서의 像계산(image computation)시 방대한 계산량이 필요로 한 문제점을 해결하기 위하여, 부분스캔설계를 도입하여 테스트 복잡도를 줄인 후 IPMT법으로 테스트생성을 한다. 부분스캔설계를 위한 스캔 플립플롭의 선택은 순차회로의 狀態함수를 二分決定그래프(binary decision diagram)로 표현했을 때의 노드의 크기 순으로 한다. 본 방법을 이용하여 ISCAS'89벤치마크회로에 대해 실험한 결과, 종래의 IPMT법에서 100%의 고장검출률을 얻을 수 없었던 s344, s349, s420에 대해 20%의 부분스캔으로 100%의 고장검출률을 얻었고, s1423에 대해서는 80%의 부분스캔으로 100%의 고장검출률을 얻었다.

Test Generation for Partial Scanned Sequential Circuits Based on Boolean Function Manipulation

Hoyong Choi†

ABSTRACT

This paper describes a test generation method for sequential circuits which improves the application limits of the IPMT method by applying the partial scan design to the IPMT method. To solve the problem that the IPMT method requires enormous computation time in image computation, and generates test patterns after the partial scan design is introduced to reduce test complexity. Scan flip-flops are selected for the partial scan design according to the node size of the state functions of a sequential circuit in their binary decision diagram representations. Experimental results on ISCAS'95 benchmark circuits show that a test generator based on our method has achieved 100% fault coverage by use of either 20% scan FFs for s344, s349, and s420 or 80% scan FFs for s1423. However, test generators based on the previous IPMT method have not achieved 100% fault coverage for those circuits.

* 이 논문은 1994년도 한국학술진흥재단의 공모과제 연구비에 의하여 연구되었음

† 정 회 원:부산공업대학교 전자공학과 부교수

논문접수:1995년 11월 13일, 심사완료:1996년 3월 12일

1. 서 론

최근의 반도체 기술의 발달에 의해 집적회로(integrated circuits:IC)의 집적도가 크게 향상됨에 따라, 제조된 IC의 신뢰성 확보를 위한 고장검사(테스트)의 중요성이 더욱 높아지고 있다[1].

IC의 고장검사는 일반적으로 주어진 회로의 외부입력에 입력계열을 인가해 이때 외부출력에서 관측되는 관측출력값과 정상적으로 동작하는 회로의 정상출력값을 비교함으로써 이루어진다. 이때 인가된 입력계열을 테스트계열이라하고 이 테스트계열을 생성하는 것을 테스트계열생성(test pattern generation: 이하 테스트생성)이라 한다.

조합회로에 대해서는 비교적 테스트생성이 용이해 실용적인 규모까지 테스트생성이 가능하다[2-4]. 그러나, 순차회로는 내부상태의 설정 및 관측의 곤란함 때문에 테스트생성시 방대한 상태공간의 탐색이 필요하여, 시간적 공간적 제약 하에서는 탐색이 도중에 중단되는 경우가 많아 테스트생성이 매우 어려운 것으로 알려져 왔다[1]. 최근 순차회로의 테스트생성에 관한 다양한 연구가 행하여져, 알고리즘에 의한 테스트생성[5-10], 테스트容易化설계를 이용한 테스트[1, 12] 등 여러 테스트생성법이 제안되었다.

알고리즘에 의한 테스트생성법 중 높은 고장검출률을 얻을 수 있는 순차회로의 테스트생성법으로서 非明示的 積機械探索(implicit product machine traversal:IPMT)법[8-10]이 있다. 이 방법은 정상(fault-free)회로와 고장(faulty)회로의 積機械(product machine)의 狀態遷移그래프(state transition graph)상에서, 리세트 상태로부터 幅優先(breadth-first)방식 혹은 폭우선/깊이우선 방식으로 탐색하면서 서로 다른 출력을 내는 입력계열을 구하는 것이다. IPMT법은 상태집합을 논리함수로 표현하고, 탐색을 논리함수연산으로 처리하는 非明示的 狀態列擧(implicit state enumeration)법[11]을 이용해 효율적으로 테스트생성을 할 수 있고, 특히 다른 방법에서 취급할 수 없었던 테스트 곤란한 고장이나 리던던트(redundant)고장의 테스트생성에 유용하다. 그러나 IPMT법에서 플립플롭(FF)이 많은 회로에 대해서는 像계산(image computation)에 방대한 기억량과 계산시간을 필요로 하여, 테스트생성이 불가능하게 되는 문제점이 있다. 여기서 상계산은 주어진 現狀態雙집합으로부터 도달 가능한 다음상태쌍집합을 구하는 계산이다.

한편 테스트容易化설계를 이용한 테스트는 스캔패스(scan path)설계 방식[1] 등을 이용하여 순차회로의 테스트생성문제를 조합회로의 테스트생성 문제로 귀착시키는 방법이다. 그러나 스캔패스를 구성하기 위하여 필요한 하드웨어의 대량 증가와 동작 속도의 저하 등의 문제점이 있어 필요 최소한의 FF만을 스캔설계하여 테스트생성의 복잡도를 개선하는 부분설계 방식[1, 12]이 제안되어 왔다.

본 논문은 종래의 IPMT법에 부분스캔설계 방법을 적용하여, IPMT법의 적용 한계를 개선한 순차회로의 테스트생성법에 관해 기술한다. IPMT법에서 회로의 플립플롭을 부분적으로 스캔화하여, 방대하게 소요되는 상계산량을 대폭 줄임으로써, 종래 테스트생성이 불가능했던 고장에 대해 고장검출이 가능토록 하고자 한다. 스캔 플립플롭의 선택은 상계산에 큰 영향을 주는 狀態함수의 二分決定그래프(binary decision diagram:BDD)의 노드 크기 순으로 한다.

2장에서는 본 논문에서 취급하는 회로, 고장모델을 정의하고, 기존의 IPMT법과 문제점을 기술한다. 3장에서는 방대한 기억량과 계산시간량이 필요한 상계산을 삭감하는 방법으로서 부분스캔방식을 도입한 테스트생성에 관하여 기술한다. 4장에서는 ISCAS'89 벤치 마크회로에 대한 실험결과를 나타내고 5장에서는 결론을 기술한다.

2. 순차회로의 테스트생성

2.1 정의

본 논문에서 사용하는 순차회로는 외부입력 n개, 외부출력 m개, FF개를 갖고 리세트 상태를 설정할 수 있는 동기식 순차회로이다. 외부입력변수벡터 x와 현재상태변수 벡터 y를 각각 식(2.1)로, 리세트상태 r를 식(2. 2)로 표기한다.

$$x=(x_1, x_2,\cdots x_n),\ y=(y_1, y_2,\cdots,y_l) \tag{2.1}$$

$$r=(r_1, r_2,\cdots x_l) \tag{2.2}$$

각 외부출력 z_i의 출력함수와 다음상태 Y_j의 다음상

태함수(혹은 상태함수)를 각각 식(2.3)과 식(2.4)로 표기한고, 이들도 마찬가지로 각각 식(2.5)와 식(2.6)로 벡터 표기한다.

$$z_i = f_i(x, y),\ i = 1, 2, \cdots, m \tag{2.3}$$

$$Y_j = g_j(x, y),\ j = 1, 2, \cdots, l \tag{2.4}$$

$$f = (f_1, f_2, \cdots, f_m) \tag{2.5}$$

$$g = (g_1, g_2, \cdots, g_l) \tag{2.6}$$

고장모델은 게이트 수준(gate level)에서의 단일 stuck-at 모델을 사용한다. 고장회로에서의 변수 및 함수는 정상회로에서의 기호에 프라임(′)을 붙여 y', f_i'와 같이 표시한다. 본 논문에서는 정상회로(fault-free machine)와 고장회로(faulty machine)의 積回路(product machine)를 고려하고, 이 적회로의 상태를 정상-고장상태쌍(이하 상태쌍)이라 $\theta = (y, y')$와 같이 표기한다. x에 대하여 $f(x, y) \neq f(x, y')$를 만족하는 상태쌍(y, y')가 존재하면 이 상태쌍 (y, y')를 불일치(inconsistent)상태쌍이라 한다. 외부출력의 故障差함수 D_F와 다음상태의 故障差함수 D_G를 각각 식(2.7)과 식(2.8)로 정의한다.

$$D_F(x, (y, y')) = \bigvee_{i=1}^{m} [f_i(x, y) \oplus f_i'(x, y')] \tag{2.7}$$

$$D_G(x, (y, y')) = \bigvee_{i=1}^{l} [g_i(x, y) \oplus g_i'(x, y')] \tag{2.8}$$

이것은 고장의 영향이 외부출력에 나타났을 때는 D_F가, 다음상태 출력함수에 나타났을 때는 D_G가 참으로 되는 함수이다. 상태쌍의 집합 Θ에 대한 상태쌍집합 遷移함수 $T(\Theta)$를 식(2.9)로 정의한다.

$$T(\Theta) = \{(v, v') \mid (u, u') \in \Theta,\ v = g(x, u),\ v' = g'(x, u')\} \tag{2.9}$$

이것은 집합 Θ에 포함된 임의의 상태쌍으로부터 遷移가능한 모든 상태쌍의 집합을 나타내는 함수이다. Θ로부터 $T(\Theta)$를 계산하는 것을 像計算(image computation)이라 한다. 적회로에서 리세트상태 (r, r')로부터 시작하여 시각 t에 처음으로 도달하는 상태쌍 집합을 Φ_t로 표시하고, 이것을 시각 t에 있어서 첫도달가능상태쌍 집합이라 하고 식(2.10)과 같이 표시한다.

$$\Phi_0 = \{(r, r')\},\ \Phi_t = T(\Theta_{t-1}) - \sum_{s=0}^{t-1} \Phi_s\ (t \geq 1) \tag{2.10}$$

2.2 非明示的 積機械探索法

非明示的 積機械探索(implicit product machine traversal : IPMT)법[8-10]은 積機械상에서 리세트상태로부터 도달가능상태쌍을 비명시적으로 열거하여, 불일치상태쌍을 폭우선방식 [8, 9] 혹은 폭우선/깊이우선 방식[10]으로 탐색하는 테스트생성법이다. 이 방법은 적기계의 모든 상태쌍에 대해 탐색하기 때문에, 리던던트가 아닌 고장에 대하여 반드시 테스트생성이 가능하다는 의미에서 완전한 알고리즘이다. 이 방법은 테스트생성시간이 종래의 경로활성화에 의한 알고리즘에 비해, 테스트생성이 쉬운 고장에 대해서는 효율이 약간 떨어지나, 테스트 곤란한 고장이나 리던던트고장에 대해서는 빠른 시간내에 처리할 수 있어 유용하다.

IPMT에서 시각 t에서의 탐색은 다음과 같이 행한다.

1) 만약 Φ_t에 불일치상태쌍이 있으면, 주어진 고장은 검출되고 탐색을 종료한다.

2) Φ_t로부터 Φ_{t+1}를 구한다.

3) $t = t + 1$로 하고 다음 시각의 탐색을 한다.

IPMT법에서는 Φ_t와 같은 상태쌍집합을 특성함수(characteristic function)[11]라 하는 논리함수를 이용하여 비명시적으로 표현하고 상태집합연산을 논리함수로 처리한다. 또한 데이터구조로 BDD(binary decision diagrams)[13]를 이용하여 논리함수를 효율적으로 표시하고, 연산을 고속으로 수행할 수 있다.

2)의 Φ_t로부터 Φ_{t+1}를 구할 때 像計算(image computation)이 필요하다. 이는 현상태쌍, 다음상태쌍, 외부입력의 관계를 나타내는 遷移관계(transition relation)[11]를 계산하고, 이로부터 현상태쌍, 외부입력을 나타내는 변수를 소거하여 다음상태쌍 집합을 나타내는 논리함수를 구하는 계산이다. 이것을 논리식으로 나타내면 식(2.11)와 식(2.12)와 같다.

$$\Gamma(\Theta) = S_x S_y S_{y'} [\{\bigwedge_{i=1}^{l} (Y_i \equiv (g_i(x, y)_\Theta)\} \cdot \{\bigwedge_{i=1}^{l} (Y_i' \equiv (g_i'(x, y')_\Theta)] \tag{2.11}$$

$$T(\Theta)=[\Gamma(\Theta)]\,(Y/y,\ Y'/y') \tag{2.12}$$

여기서, $S_x f$는 f의 변수 $x=(x_1,\ x_2,\cdots,x_k)$에 의한 smoothing이고 식(2.13)으로 정의된다.

$$S_x f=S_{x1}S_{x2}\cdots S_{xk}f,\ S_{xi}f=f_{xi}+f_{\overline{xi}},\ i=1,\ 2,\cdots,k \tag{2.13}$$

여기서 f_a는 f의 literal a에 의한 cofactor이고, f_x, $f_{\overline{x}}$는 f의 각각 1, 0을 대입하여 얻은 함수이다. 또한 식(2.13)은 $\Gamma(\Theta)$에서 변수 $(Y,\ Y')$를 $(y,\ y')$로 변환하는 연산이다. 이중 식(2.12)의 변수변환은 용이하나, 식(2.11)의 연산시 FF수가 많은 경우 소요기억량과 계산시간이 방대하게 필요되어, 도중에 연산이 중단되는 경우가 있어 고장검출률이 낮게 된다.

2.3 부분스캔설계방식

부분스캔설계(partial scan design)방식[1,12]은 테스트容易化설계의 하나인 스캔설계(scan design)방식을 개량한 방식이다. 스캔설계방식에서는 순차회로의 각 FF을 쉬프트레지스터(shift register)로 구성하여 스캔패스를 만들어, 테스트모드에서 이 스캔패스를 통하여 FF의 상태의 설정 및 관측을 용이하게 한 것이다. 이 방식을 채용하면 테스트생성 효율이 비약적으로 개선되나, 테스트용이화를 위한 부가하드웨어의 증가, 회로성능의 저하, 테스트계열의 증대 등의 문제가 있다. 따라서 필요 최소한의 FF을 스캔하여 하드웨어의 증가 등의 문제를 해결하는 동시에 높은 검출률을 얻는 부분스캔방식이 유용한 테스트 방식으로 알려져 왔다[1, 12].

3. 부분스캔순차회로의 테스트생성

본 장에서는 IPMT법에서의 상계산시 방대한 기억량과 계산시간량이 필요하여, 고장검출이 도중 중단되는 경우[9, 10]가 있어 이의 해결책으로 IPMT법에 부분스캔방식을 적용한 테스트생성법 PSSTAR에 관하여 기술한다.

3.1 스캔플립플롭의 선택

IPMT법에서는 논리함수를 효율적으로 표시하고 처리하기 위해 데이터구조로 BDD(binary decision diagrams)를 이용한다. BDD에서는 n변수 논리함수를 표현하기 위해 최악의 경우의 노드수는 $2^n/\log n$이 되어[14], 변수의 수가 논리함수 표현의 크기에 크게 좌우된다. 상계산을 할 경우, 식(2.11)의 변수의 수는 $4n+l$로 되어 FF의 수에 의해 크게 변한다. 만약 p개의 FF을 스캔하면 상계산시의 변수의 크기는 $3p$개로 줄일 수 있고 논리함수의 표현이 간단하므로, 종래의 IPMT법에서 도중에 중단되었던 테스트에 대해서 테스트생성을 계속할 수 있게 된다.

부분스캔 설계에서의 스캔 FF은 다음과 같이 선택한다.

순차회로의 조합회로부에 동적가중치할당법(dynamic weight assignment method:DWA법)[12]을 적용해 외부입력변수, 상태변수를 구별하지 않고 가중치를 부여해 가중치가 큰 변수에 높은 순위를 부여한다. 이와 같은 변수의 순서부여를 이용하여 상태함수의 BDD 표현을 하고, 이 상태함수의 BDD표현의 크기가 큰 순으로 스캔 FF을 선택한다. 이는 상계산의 계산량이 상태함수의 BDD표현의 크기에 비례하고, BDD표현의 크기가 큰 상태함수(상태변수)를 우선 스캔함으로써 상계산량을 보다 많이 줄이고자 하기 때문이다.

3.2 부분스캔순차회로의 테스트생성

테스트생성법 PSSTAR는 3.1의 스캔 FF의 선택법을 이용해 구성한 부분스캔회로에 IPMT를 적용한 테스트방법이다.

부분스캔할 양이 결정되면 3.1의 방법을 이용하여 스캔할 FF을 선택하고, 스캔 FF의 출력을 외부입력으로 스캔 FF의 입력을 외부출력으로 바꾸어 넣어 부분스캔회로를 만든다. 이는 종래의 순차회로의 테스트생성시 방대한 기억량과 계산시간량이 필요했던 상계산량을 현저히 줄여줌으로써, 테스트생성이 불가능했던 고장에 대해 고장검출이 가능토록한다.

IPMT법에 부분스캔 방식을 도입한 PSSTAR 알고리즘을 (그림 1)에 나타냈고, 알고리즘을 설명하면 다음과 같다.

(그림 1)의 main()에서는, 먼저 부분스캔량이 결정되면 3.1의 방법을 이용하여 스캔할 FF을 선택하고 스캔 FF의 출력을 외부입력으로 스캔 FF의 입력을 외부출력으로 바꾸어 넣는다. 다음에 외부출력의 고

장차함수 혹은 다음상태출력의 고장차함수를 참으로 하는 입력 x와 y의 존재를 조사한다. 이와 같은 (x, y)가 존재하지 않으면 대상고장은 조합적 리던던트(combinational redundant)가 된다. 만약 (x, y)가 존재하면 시각 $t=0$에서의 탐색대상으로 하는 상태쌍집합을 초기화하고 SeqTestGen()을 부른다. SeqTestGen()은 시각 t와 첫도달가능상태쌍 집합 Φ_t와 도달가능상태쌍 집합 Ψ_t를 인수로 하여 Φ_t 및 Ψ_t로부터 도달가능한 상태쌍 중에서 불일치상태쌍의 존재를 조사한다. 만약 불일치상태쌍이 존재하면 상태쌍에 도달가능한 상태쌍을 돌려받아 테스트계열 $v(0), v(1), \cdots, v(k)$를 얻어 출력종료한다. 만약 존재하지 않으면 순차적 리던던트(sequential redundant)가 된다.

SeqTestGen()은, 먼저 시각 t에 있어서 주어진 첫도달가능상태쌍 집합 Φ_t에 대해서 불일치상태쌍의 존재를 조사한다. 불일치상태쌍이 존재하면 그 시각에 있어서의 검사입력 $v(t)$로 하고, 그의 상태쌍(ϕ_t)를 반환값으로 리턴한다. 존재하지 않으면 Φ_t와 Ψ_t로부터 Φ_{t+1}를 구한다. 여기에서 Φ_t이 공집합이 되면 고장을 외부출력에 전파하는 내부상태쌍이 존재하지 않으므로 순차적 리던던트가 되어 검사입력 없이 복귀한다. 그렇지 않으면 시각을 1 증가시켜 Φ_{t+1}을 첫도달가능상태쌍 집합, Ψ_t를 도달가능상태쌍 집합으로하여 재귀적으로 SeqTestGen()을 부른다. 그리고 다음 시각에서 상태쌍 ϕ_{t+1}(와 검사입력)을 얻어 돌아오면 Φ_t에 속한 상태쌍 중에서 ϕ_{t+1}으로 천이하는 상태쌍 ϕ_t와 그의 입력 $v(t)$(시각 t에서의 검사입력)를 구해, ϕ_t를 반환한다.

```
main()
{
  select partial scan FFs according to BDD size of
  state functions;
  replace the output of the scan FF with PI and the
  input of the scan FF with PO;
  if (there is no(x, y) s.t. D_F(x, (y, y)) ∨ (D_G(x, (y,
  y)) = 1)
    combinationally redundant;
  else {
    Ψ_0 = {(r, r)}; Φ_0 = Ψ_0;
    φ = SeqTestGen(0, Φ_0, Ψ_0);
    if(φ = null) sequentially redundant;
    else ouput a TP sequence v(0), v(1),···,v(k);
    }
  }
SeqTestGen (t, Φ_t, Ψ_t)
{
  if ((a, θ) exist s.t. D_F(a, θ) = 1 for θ ∈ Φ_t){
    v(t) = a;
    return (θ);
  } else {
    Φ_{t+1} = T(Φ_t) ∩ ¬Ψ_t;
    if (Φ_{t+1} ≠ empty){
      Ψ_{t+1} = Ψ_t ∪ Φ_{t+1};
      φ_{t+1} = SeqTestGen(t + 1, Φ_{t+1}, Ψ_{t+1});
      if(φ_{t+1} ≠ null) {
        get (a, θ) that makes a transition from θ ∈ Φ_{t+1};
        v(t) = a;
        return(θ);
      }
    }
  return (null);
  }
}
```

(그림 1) PSSTAR 알고리즘
(Fig. 1) PSSTAR algorithm

4. 실험결과

앞 장에서 기술한 부분스캔방식을 도입한 순차회로 테스트생성수법을 C언어를 사용해, SPARCstation2 상에 구현해 ISCAS'89벤치마크회로[15]에 대하여 실험하였다. 〈표 1〉에 실험한 ISCAS'89벤치마크회로의 스펙을 나타냈다. 〈표 1〉의 각 난은 외부입력수(#PI), 외부출력수(#PO), D-FF수(#DFF), 게이트수(#Gate), 대표고장수(#Faults)를 나타낸다. 모든 FF의 상태가 0인 상태를 리세트상태로 가정하였다. 테스트생성에서의 논리함수의 처리에는 SBDD 연산 프로그램을 이용하였다. PSSTAR는 본 논문에서 제안하는 알고리즘이고, STAR는 단순한 폭우선에 의한 IPMT법을 실현한 것을 나타낸 것이다. PSSATR에서의 20%,

〈표 1〉 ISCAS'89 벤치마크회로
〈Table 1〉 ISCAS'89 Benchmark circuits

Circuit	#PI	#PO	#DFF	#GATE	#Faults
s27	4	1	3	10	32
s208	11	2	8	96	215
s298	3	6	14	119	308
s344	9	11	15	160	342
s349	9	11	15	161	350
s382	3	6	21	158	399
s386	7	7	6	159	384
s400	3	6	21	162	426
s420	19	2	16	196	430
s444	3	6	21	181	474
s510	19	7	6	211	564
s526n	3	6	21	194	553
s526	3	6	21	193	555
s641	35	24	19	379	467
s713	35	23	19	393	581
s820	18	19	5	289	850
s832	18	19	5	287	870
s953	16	23	29	395	1079
s1196	14	14	18	529	1242
s1238	14	14	18	508	1355
s1423	17	5	74	657	1515
s1488	8	19	6	653	1486
s1494	8	19	6	647	1506

40%, 80%의 수치는 부분스캔률(스캔된 FF수/전FF수)을 나타낸다. 회로중의 전 고장에 대해 테스트생성을 하였고, 고장시뮬레이션은 적용하지 않았다. 실행시 SBDD의 최대노드수는 $2^{20}(\simeq 10^6)$으로 하였다.

실험결과를 〈표 2〉와 〈표 3〉에 나타냈다. 〈표 2〉는 고장검출률과 테스트생성시간을, 〈표 3〉에서는 테스트생성시의 SBDD의 최대노드수와 최대의 테스트계열길이를 비교하였다. FC는 고장검출률, Time은 테스트생성시간, N은 테스트생성시의 SBDD의 최대노드수, L은 최대의 테스트계열길이를 나타낸다. 여기서, 고장검출률은 (검출고장수 +리던던트고장수)/#Faults × 100%율, s는 초를, h는 시간을 나타낸다.

PSSTAR에서는 全 대상회로 대해 100%의 고장검출률을 얻었다. 회로 s344, s349, s420에 대해서 STAR에서는 일부 고장에 대해서 SBDD의 노드수가 2^{20}을 초과하였기 때문에 100%의 고장검출률을 얻을 수 없었던 것에 비해 PSSTAR에서는 모두 100%의 고장검출률을 얻을 수 있었다. 이는 부분스캔에 의해 테스트시 최대 필요 노드수가 10^5개 이하로 줄어들었기 때문이다. 회로 s1423은 STAR에서는 대부분의 고장에 대해서 SBDD의 노드수가 2^{20}을 초과하고 테스트생성시간이 20시간을 초과해 테스트생성이 불가능하였는 데 비해 PSSTAR에서는 80%스캔으로 1시간 내에 100% 고장검출률을 얻을 수 있었다. 스캔률이 높을수록 테스트생성시간, 최대노드수, 최대테스트계열길이가 대폭 줄어들었다. 테스트생성시간에 있어서 부분스캔률 20%의 경우 약 10배, 40%의 경우 86배, 80%의 경우 780배의 고속화를 실현했다. 이는 스캔

〈표 2〉 고장검출률과 테스트생성시간의 비교
〈Table 2〉 Comparison of fault coverage and test generation time

Circuit name	STAR		PSSTAR					
			20%		40%		80%	
	FC	TIME	FC	TIME	FC	TIME	FC	TIME
s27	100%	0.10s	100%	0.0s	100%	0.0s	100%	0.0s
s208	100%	15.1s	100%	4.6s	100%	2.3s	100%	0.7s
s298	100%	22.1s	100%	15.8s	100%	7.1s	100%	0.7s
s344	93%	0.51h	100%	208.4s	100%	18.6s	100%	1.5s
s349	92%	0.55h	100%	213.9s	100%	27.9s	100%	1.7s
s382	100%	0.92h	100%	360s	100%	18.4s	100%	1.4s

s386	100%	4.50s	100%	2.6s	100%	1.9s	100%	1.0s
s400	100%	0.96h	100%	329.7s	100%	19.1s	100%	1.5s
s420	87%	8.60h	100%	17.2s	100%	107.4s	100%	3.4s
s444	100%	1.13h	100%	318.9s	100%	30.7s	100%	1.8s
s510	100%	1.89h	100%	641.1s	100%	12.5s	100%	2.6s
s526n	100%	1.45h	100%	460.8s	100%	86.0s	100%	2.7s
s526	100%	1.13h	100%	465.1s	100%	50.4s	100%	2.8s
s641	100%	0.87h	100%	102.6s	100%	21.0s	100%	12.8s
s713	100%	0.99h	100%	71.0s	100%	24.3s	100%	9.4s
s820	100%	58.9s	100%	12.7s	100%	7.4s	100%	3.9s
s832	100%	59.0s	100%	13.0s	100%	7.7s	100%	4.0s
s953	100%	0.37h	100%	113.4s	100%	27. 8s	100%	7.4s
s1196	100%	0.25h	100%	343.3s	100%	233.0s	100%	13.9s
s1238	100%	0.26h	100%	483.5s	100%	322.7s	100%	20.3s
s1423	–	–	27.1%	13.88h	40.5%	9.07h	100%	0.92h
s1488	100%	76.5s	100%	26.3s	100%	17.7s	100%	8.3s
s1494	100%	77.8s	100%	19.1s	100%	17.6s	100%	8.4s

〈표 3〉 최대 노드수와 최대 테스트계열길이의 비교
〈Table 3〉 Comparison of max. node number and max. test sequence length

Circuit name	STAR		PSSTAR					
			20%		40%		80%	
	N	L	N	L	N	L	N	L
s27	36	2	22	2	22	2	6	2
s208	48	18	36	18	30	17	9	5
s298	2799	20	721	15	256	15	78	3
s344	898172	7	76460	7	14852	7	34	4
s349	898280	7	76460	7	18244	7	18	4
s382	376481	133	13117	18	1117	18	186	3
s386	85	9	90	4	76	4	38	3
s400	435935	133	13144	18	1102	18	183	3
s420	75	18	88	18	69	18	86	5
s444	254079	133	12029	18	2051	18	292	3
s510	40	48	49	21	70	5	17	3
s526n	273698	133	6439	30	2696	18	75	6
s526	273532	133	6419	30	2378	18	75	6
s641	19758	5	3450	5	804	5	103	2
s713	19760	5	3452	5	856	5	103	2
s820	167	12	213	7	316	4	60	3
s832	167	12	213	7	316	4	60	3
s953	59100	12	9068	3	1857	3	179	2
s1196	102408	3	9201	3	3937	2	319	2
s1238	115401	3	12460	3	4283	2	503	2
s1423	–	–	238125	3	158027	3	36552	3
s1488	114	23	253	7	226	5	48	3
s1494	114	23	149	7	226	6	48	3

화에 의해 SBDD의 최대노드수와 최대의 테스트계열길이가 감소함으로써 상계산시 계산량의 감소에 기인한다. 특히 부분스캔화에 의해 노드수의 감소가 크면 클수록, 테스트계열길이가 짧아지면 짧아질수록 테스트생성시간이 단축되었다.

5. 결 론

본 연구에서는 부분스캔방식의 도입에 의해 상계

산의 계산량을 줄이고, IPMT법의 적용 범위를 확대하는 방법을 제안하였다. 스캔 FF의 선택법으로서는 회로의 상태함수의 이분결정그래프 크기 순으로 선택하였다.

본 방법을 이용한 ISCAS'89벤치마크회로에 대해 실험한 결과, 종래의 IPMT법에서 100%의 고장검출률을 얻을 수 없었던 s344, s349, s420에 대해 20%의 부분스캔으로 100%의 고장검출률을 얻었다. 또한 테스트생성이 불가능했던 s1423에 대해서는 80%의 부분스캔으로 100%의 고장검출률을 얻었다. 테스트생성시간에 있어서 부분스캔률 20%의 경우 약 10배, 40%의 경우 86배, 80%의 경우 780배의 고속화를 실현했다.

금후의 과제는, 보다 큰 대규모의 회로에 대해서 테스트생성을 하는 문제가 있다. 대규모회로를 IPMT법을 적용할 수 있는 규모로 분할하여 적용하는 방법을 고려할 수 있다. 또한 보다 효율적인 BDD의 순서부여법을 연구하는 것이 중요하다.

참 고 문 헌

[1] M. Abramovichi, M. A. Breuer, and A. D. Friedman, 'Digital Systems Testing and Testable Design', Computer Science Press. 1990.

[2] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," IEEE Trans. Computer, Vol. C-30, No. 3, pp. 215-222 Mar. 1981.

[3] H. Hujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithm," IEEE Trans. Computer, Vol. C-32, No. 12, pp. 1137-1144, Dec. 1983.

[4] M. H. Schultz, E. Trischler, and T. M. Sarfert, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System," IEEE Trans. on Computer-Aided Design, Vol. CAD-7, No. 1, pp. 126-137, Jan. 1988.

[5] W. T. Cheng and T. J. Chakraborty, "Gentest: An Automatic Test-Generation System for Sequential Circuits," IEEE Computer, Vol. 22, No. 4, pp. 43-49, Apr. 1989.

[6] H-K. T. Ma, S. Devadas, A. R. Newton, and A. Sangiovanni-Vincentelli, "Test Generation for Sequential Circuits," IEEE Trans. on Computer-Aided Design, Vol. CAD-7, No. 10, pp. 1081-1093, Oct. 1988.

[7] A. Ghosh, S. Devadas, and A. R. Newton, "Test Generation and Verification for Highly Sequential Circuits," IEEE Trans. on Computer-Aided Design, Vol. CAD-10, No. 5, pp. 652-667, May 1991.

[8] H. Cho, G. D. Hachtel, and F. Somenzi, "Fast Sequential ATPG Based on Implicit State Enumeration," in Proc. Int. Test Conf., pp. 67-74, Oct. 1991.

[9] H. Choi, T. Kohara, N. Ishiura, and I. Shirakawa, "Test Generation for Sequential Circuits Based on Boolean Function Manipulation," in Proc. 1992 Joint Technical Conf. on Circuits/Systems, Computers and Communications, pp. 248-253, July 1992.

[10] 崔湨鎔, 小原, 石浦, 白川, 本原, "論理關數處理に基づいた順序回路のテスト生成法," 電子情報通信學會論文誌, Vol. J76-A, No. 6, pp. 835-843, June 1993 年 6月(in Japanese).

[11] H. J. Touati, H. Savoj, B. Lin, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Implicit State Enumeration of Finite State Machines Using BDD's," in Proc. IEEE Int. Conf. on Computer-Aided Design, pp. 130-133, Nov. 1990.

[12] K. T. Cheng and V. D. Agrawal, "A Partial Scan Design for Sequential Circuits with Feedback," IEEE Trans. Computer, Vol. C-39, pp. 544-558, Apr. 1990.

[13] S. Minato, N. Ishiura, and S. Yajima, "Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation," in Proc. 27th Design Automation Conf., pp. 52-57, June 1990.

[14] 石浦, "二分決定とは," in Proc. 6th Karuizawa Workshop on Circuits and Systems, pp. 155-160, April 1992 (in Japanese).

[15] F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," in Proc. Int. Symp. on Circuits and Systems, pp. 1929-1934, June 1989.

최 호 용

1980년 서울대학교 전자공학과 졸업(학사)
1982년 한국과학기술원 전기 및 전자공학과(공학석사)
1994년 일본 오오사까대학 대학원 전자공학과(공학박사)
1982년~1985년 삼성반도체 연구소 연구원
1985년 8월~현재 부산공업대학교 전자공학과 전임강사, 조교수, 부교수
관심분야: VLSI 테스트, VLSI 설계