論文97-34C-3-2

다단 회로에서 테스트 불가능한 경로 검출을 위한 효율적인 알고리듬의 설계

(Design of an Efficient Algorithm for the Detection of Untestable Paths in Multi-level Circuits)

許 勳 *, 黃 善 泳 *

(Hoon Heo and Sun Young Hwang)

요 약

본 논문에서는 다단 회로에서 테스트 불가능한 경로를 효율적으로 검출하는 알고리듬의 구현과 설계에 대하여 기술한다. 제안된 알고리듬은 다단 회로에 대해 BDD (Binary Decision Diagram)을 이용하여 멀티플랙서 구조 회로로 전환하고 이에 대한 테스트 불가능한 경로를 검출하여 제거한다. 재결합 경로에 대한 ENF (Equivalent Normal Form)를 구성하여 빠른 시간에 적은 메모리를 사용하여 효율적으로 테스트 불가능한 경로를 검출하고 이를 제거한다. 표준 테스트 회로인 MCNC/ISCAS benchmark에 대한 실험 결과, 구현된 시스템은 경로 지연 시간 결함을 검출하는데 기존 방법에 비해 사용되는 CPU 시간과 메모리량보다 각각 37.7%, 60.9% 감소하는 결과를 보인다.

Abstract

This paper presents the design and implementation of an efficient algorithm for detecting untestable paths in multi-level circuits. Transforming multi-level circuit into a multiplexor-based one through BDD (Binary Decision Diagram) construction, the proposed algorithm detects untestable paths in the transformed circuits. By constructing ENF (Equivalent Normal Form) only for reconvergent paths, the proposed system detects and removes untestable paths efficiently in terms of the run-time and memory usage. Experimental results for MCNC/ISCAS benchmark circuits show that the system efficiently detects and removes untestable paths. The run-time and memory usage have been reduced by 37.7% and 60.9%, respectively, compared to the previous methods.

I. 테스트 용이한 논리 합성

최근 반도체 공정 기술의 급속한 발전으로 실리콘 단위 면적당 칩의 컴퍼넌트들 수가 현저히 증가하면서, 설계의 복잡성및 설계 기간의 증가와 시스템 전반에 대한 전문적인 기술과 지식을 소유한 다수의 전문 설계자들의 필요성, 그리고 제작 기술의 변화에 따른 재설계 등이 VLSI설계에 있어 어려운 문제점으로 부각되었다. 이와 같은 문제점을 해결하기 위해 다양한 설계 방식과 라이브러리를 지원할 수 있는 시뮬레이터 등의 CAD툴과 설계 자동화 툴의 사용이 필수적으로 되었다. 이에 국내외의 각각 연구기관에서 설계 자동화 시스템의 연구 개발을 활발히 진행되어 발표되고 있으며, 특히 논리수준에서의 CAD툴 개발은 기술 독립적 최적화 과정인 다단 논리 최적화[1,2]와 기술 의존적 최적화 과정인 기술 매핑 분야[3,4]에서 많은 성과를 이루고 있다.

주어진 면적, 동작 속도 등의 설계 제약 조건을 만족하는 데 중점을 두어 개발된 기존의 논리 합성 시스템은 비교적 빠른 시간내에 최적화된 VLSI 회로를 설계하나, 생성된 회로에 대한 테스트에는 많은 시간이 소

* 正會員, 西江大學校 電子工學科
(CAD & Computer System Design)

接受日字:1996年月日, 수정완료일:1997年月日

요되는 문제점을 가지고 있어 실제 유용한 칩의 개발을 어렵게 하였다. 생성된 회로의 테스트 용이성을 향상시키기 위해 데이타의 scan-in/out이 가능한 scan 방식으로 설계하거나, 논리 합성후에 BIST (Built-In Self-Test)를 위하여 MISR (Multiple Input Signature Register) 혹은 LFSR (Linear Feedback Shift Register) 등의 하드웨어의 추가 과정을 거친다[5]. 이 방법들은 원하는 정도의 테스트 용이성을 얻기 위한 추가되는 하드웨어로 인한 회로 동작 속도의 저하 등의 중요한 문제점을 야기한다. 이와 같은 단점을 보완하기 위하여 최근에는 논리 합성 과정에서 동시에 테스트를 고려한 연구가 이루어지고 있다[6,7,8,9,10]. 특히, 효율적인 don't care의 추출과 이용[11], ATPG (Automatic Test Pattern Generation)를 이용한 회로의 불검출 결함(redundant faults)의 검출과 이의 제거 방법이 연구되고 있다[12,13].

일반적으로 회로를 테스트하는 경우에 발생되는 여러 형태의 결함들을 검출하기 위한 모델이 필요하다. 가장 널리 알려져 있는 결함 모델로는 컴퍼넌트들 사이의 연결을 일정한 값으로 변경시키는 결함들을 검출하기 위한 고착 결함 모델이며, 이 모델을 통하여 하나의 결함 효과가 회로에 영향을 미치지 않는 불검출 결함들을 검출한다. 이 밖에도 둘 이상의 연결들이 단락되어 회로에 영향을 주는 결함을 검출하기 위한 bridge fault model[5], 트랜지스터를 항상 구동시킴으로써 회로에 영향을 주는 결함을 검출하기 위한 stuck-on fault model등이 있다[5].

1. 지연 시간 결함과 테스트 불가능한 경로

지연시간 결함 모델은 클럭기간과 같은 특정한 제약 시간보다 신호 전달시간의 지연으로 인한 회로의 동작 속도를 저하시키는 결함들을 검출하기 위한 모델이다[14]. 이 지연시간 결함 모델에서는 크게 한 게이트의 입력이나 출력에 대한 지연시간 변화로 인한 회로의 동작 속도를 저하시키는 지연시간 결함을 검출하기 위한 게이트 지연시간 결함 모델과, 회로의 특정한 경로에 대한 지연시간의 변화로 인한 회로의 동작 속도를 저하시키는 지연시간 결함을 검출하기 위한 경로 지연시간 결함 모델로 구분된다. 본 논문에서는 지연시간 결함 모델중에서 경로 지연시간 결함 모델을 사용하여 테스트 용이한 논리 회로를 합성한다.

이들 결함 모델을 이용하여 테스트 용이한 회로들 사이의 관계는 명확히 알려져 있지 않으나, 대상 회로를 멀티플랙서 구조 회로로 변환한 경우 고착 결함에 테스트 용이하면 경로 지연시간 결함에 테스트 용이하다고 보고되었다[15]. 멀티플랙서 구조 회로에서의 고착 결함과 경로 지연시간 결함의 관계를 이용하여 테스트 용이한 회로를 생성하는 기존의 방법에서는 경로 지연시간 결함을 검출할 수 없는 경로들 즉, 테스트 불가능한 경로들을 검출하기 위해서 ENF (Equivalent Normal Form)을 이용한다[15,16,17]. ENF는 이단 논리 회로 형태의 큐브들로 구성되며 그들을 구성하는 literal들의 태그는 대상 회로에 대한 경로들을 나타내는 특성을 가지고 있어, 경로 지연시간 결함에 테스트 용이한 회로에 대한 정의가 명확히 되지 않는 다단 논리 회로에 대해서 명확히 정의된 이단 논리 회로의 경우로 전환하여 테스트 불가능한 경로들을 검출한다. 그러나, 임의의 다단 논리 회로에 대해서 ENF를 나타내는 과정은 소비되는 시간과 메모리 요구량의 급격한 증가로 인해 회로에 대한 ENF를 얻는 데에 불가능하게 되는 문제점을 안고 있다.

본 논문에서는 테스트 불가능한 경로를 효율적으로 검출하는 방법에 대하여 기술한다. 제안된 방법에서는 재결합 경로에 대하여만 ENF을 구성하여 짧은 시간에 적은 메모리를 사용하여 경로 지연 시간 결함과 테스트 불가능한 경로들을 효율적으로 검출한다. Ⅱ장에서는 BDD[18,19,20]를 이용하여 대상회로를 멀티플랙서 구조 회로로 변환 과정과 변환된 회로내에 존재하는 불검출 결함 제거 과정을 설명하고 Ⅲ장에서는 주출력단에 대한 ENF와 재결합 경로에 대한 ENF를 이용하여 테스트 불가능한 경로를 검출하는 과정을 제시하였으며, 테스트 불가능한 경로들을 제거하는 과정을 설명하였다. Ⅳ장에서는 ISCAS회로와 MCNC회로에 대하여 실험 결과를 보였으며 Ⅴ장에서는 결론을 제시하였다.

2. 테스트 용이한 논리 합성 시스템

그림 1은 테스트 용이한 논리 회로 합성 시스템에 대한 흐름도를 나타내었다. Equation, BLIF, VHDL, XNF 형태의 입력회로를 파싱하여 네트리스트를 구성한 후 회로의 주출력단에 대하여 BDD를 구성한다. 구성된 BDD의 각 노드들을 멀티플랙서로 변환하여 회로를 추출하며, 추출된 회로에 대한 불검출 결함을 확인하여 이를 제거한다. 불검출 결함을 제거한 회로들에서

는 주입력단에서 발생한 event들을 동시에 주출력단에 전달하는 서로 다른 여러 경로들이 존재할 수 있으며, 이 경로들은 회로내에 경로 지연시간 결함을 검출하지 못하는 테스트 불가능한 경로를 형성한다. 회로내의 테스트 불가능한 경로를 검출하기 위해 회로에 대한 ENF를 구성하고 이를 이용한다. 그러나, 임의의 다단 논리 회로에 대해서 ENF를 나타내는 과정은 소비되는 시간과 메모리 요구량의 급격한 증가로 인해 불가능하게 되는 문제점을 가지고 있으며, 회로의 주출력단들에 대한 ENF은 회로들내의 모든 경로에 대한 정보를 나타내지만 대부분의 테스트 불가능한 경로를 검출하는데 관련없는 불필요한 큐브들로 구성된다. 이와 같은 문제점을 보완하고자 테스트 불가능한 경로들의 특성인 재결합 경로를 중심으로 ENF를 구성하고 이를 이용하여 테스트 불가능한 경로들을 효율적으로 검출한다. 검출된 테스트 불가능한 경로를 제거하여 불검출 결함을 제거한 회로들에 대해 경로 지연시간 결함에 테스트 용이하도록 한다.

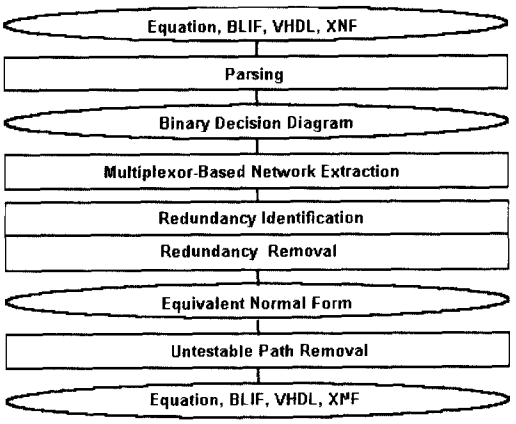


그림 1. 테스트 용이한 논리 합성 시스템에 대한 흐름도

Fig. 1. Testable logic synthesis system flow chart.

II. 이론적 배경

1. BDD로부터 회로 추출 과정

BDD의 노드 v에 대하여 low(v), high(v)를 각각 노드 v에 해당되는 변수의 값이 0일때, 1일때 가리키는 노드라 할 때, 멀티플랙서 구조 형태의 회로를 추출한다.

1) low(v)와 high(v)가 비단말 노드인 경우

$f_v = x_{index(v)}' f_{low(v)} + x_{index(v)} f_{high(v)}$의 2-입력 멀티플랙서로 변환한다. 여기서 index(v)는 BDD노드 v의 인덱스이고, $x_{index(v)}$는 노드 v에 대응되는 변수를 나타낸다. 함수 $f_{low(v)}$, $f_{high(v)}$ 는 각각 low(v), high(v)를 루트로 하는 서브그래프에 해당되는 함수이다.

2) low(v)나 high(v)가 단말 노드인 경우

$f_{low(v)} = 1$ 이면, $f_v = x_{index(v)}' + f_{high(v)}$의 형태로 변환된다.

기본적으로 BDD의 모든 노드들은 위의 과정을 통하여 멀티플랙서로 변환되나, BDD에서 low(v) 혹은 high(v)가 단말 노드를 가진 비단말 노드는 하나의 멀티플랙서로 변환하지 않고 간단화된 AND, OR, inverter 또는 다른 멀티플랙서의 입력형태로 변환된다. 그림 2는 예제 회로에 대하여 v0, v1, v2, v3 순의 배열 순서로 구성된 BDD 그래프와 이로부터 추출된 회로를 나타내었다. 노드 n6 인 경우는 간단화된 inverter로 매칭되며, 노드 n4는 AND 게이트인 NODE_0_2의 입력이 되고, 노드 n5은 다른 멀티플랙서의 입력형태로 된다. 그외의 노드들은 멀티플랙서로 매칭된다. 회로 추출 과정에 걸리는 시간은 구성된 BDD의 크기에 선형적으로 비례한다.

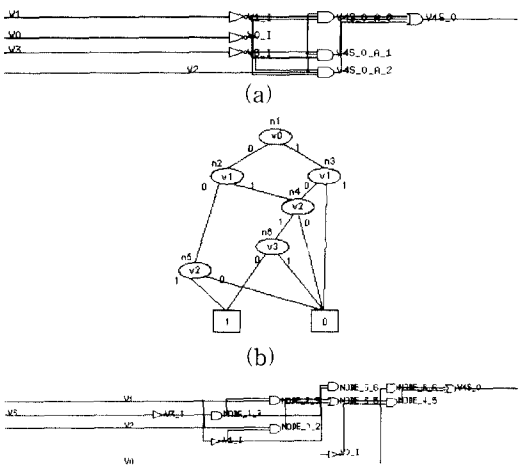
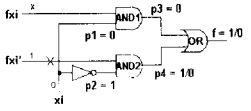


그림 2. BDD를 이용하여 멀티플랙서 구조 회로로 변환하는 예
(a) 예제 회로 (b) 예제 회로에 대한 BDD (v0, v1, v2, v3 순) (c) BDD로부터 추출된 회로

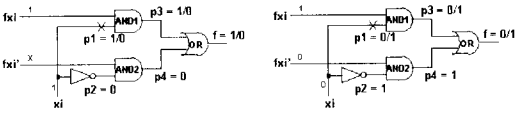
Fig. 2. Circuit extraction from BDD
(a) An example circuit (b) BDD representation (ordering :v0, v1, v2, v3) (c) Circuit extracted from (b).

2. BDD로부터 추출된 회로의 불검출 결함 제거

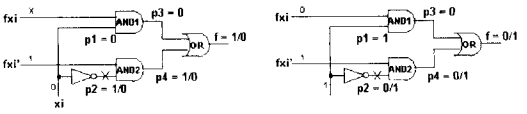
BDD로부터 추출된 회로의 크기는 구성된 BDD의 크기에 영향을 받으며, BDD에서의 중복된 서브그래프는 회로 변환시에 최적화되지 않은 회로를 형성하게 되므로 BDD를 최대한으로 축소하여 축소된 BDD로부터 회로를 추출한다. 멀티플랙서에 대한 함수 f를 $f = x_i' \cdot f_{xi}' + x_i \cdot f_{xi}$로 표현할 때 콘트롤 입력 x_i이외의 입력 f_{xi}'와 f_{xi}를 주변입력이라 정의할 때, BDD의 노드 v와 v에 대응되는 멀티플랙서를 살펴보면 노드 v에 해당되는 변수는 콘트롤 입력이 되며 low(v)를 루트로 하는 서브그래프 함수는 주변 입력 f_{xi}' 함수에 해당되며 high(v)를 루트로 하는 서브그래프 함수는 주변 입력 f_{xi} 함수에 해당된다. 이와 같은 관계를 가진 멀티플랙서에 대한 고착 결함들의 결함 효과를 출력단으로 보내기 위한 방법들을 살펴봄으로써 불검출 결함을 결정한다. 첫번째, 멀티플랙서의 주변입력들 f_{xi}', f_{xi}에 대한 고착 결함인 경우는 콘트롤 입력 x_i의 값을 조정하여 결함 효과를 출력단에 전달시킬 수가 있다. 그림 3(a)는 주변입력 f_{xi}'에 대한 고착-0 결함의 효과를 출력단에 보내는 과정을 나타내었다. AND 게이트들의 입력에 대한 고착-0 결함인 경우는 주변입력들 f_{xi}'나 f_{xi}의 값을 조정하여 결함 효과를 출력단에 보낸다. 그림 3(b)은 AND1 게이트의 입력 p1에 대한 고착-0 결함과 고착-1 결함의 효과를 출력단에 보내는 과정을 나타내었으며, 그림 3(c)는 AND2 게이트의 입력 p2에 대한 고착 0 결함과 고착-1 결함의 효과를 출력단에 보내는 과정을 나타내었다. 이 때 AND 게이트들의 입력에 대한 고착-1 결함의 효과를 출력단으로 전달하기 위해서는 주변입력들 f_{xi}'와 f_{xi}의 값을 함께 조정해야 한다. 즉, 그림 3(b)와 같이 AND1 게이트의 입력 p1에 대한 고착-1 결함은 $f_{xi}' = 1$ 과 $f_{xi} = 0$를 동시에 만족하는 경우에 출력단에 전달할 수가 있으며, 그림 3(c)와 같이 AND2 게이트의 입력 p2에 대한 고착 1 결함은 $f_{xi}' = 0$ 과 $f_{xi} = 1$를 동시에 만족하는 해가 존재하는 경우에 결함 효과를 출력단에 보낼 수가 있다. 그러나, $f_{xi}' \cdot f_{xi}' = 1$ 혹은 $f_{xi}'' \cdot f_{xi} = 1$을 만족하는 해가 반드시 존재한다는 보장이 없으므로 이들에 대한 확인이 필요하며, 만족하는 해가 없을 시에는 불검출 결함이 된다. 다음으로 AND 게이트의 출력에 대한 고착 결함은 주변입력들 f_{xi}'나 f_{xi}의 값과 콘트롤 입력 x_i의 값을 조정하여 결함 효과를 출력단에 전달할 수가 있다. 그림 3(d)는 AND2 게이트의 출력 p4에 대한 고착-0 결함 효과를 출력단에 전달하는 과정을 나타내었다. 이는 $x_i = 0$ 이고 $f_{xi}' = 0$ 인 경우에 결함 효과를 출력단에 전달할 수가 있다. 마지막으로 OR 게이트의 출력에 대한 고착 결함은 그림 3(e)와 같이 주변입력들 f_{xi}'나 f_{xi}의 값과 콘트롤 입력 x_i의 값을 조정하여 결함 효과를 출력단에 전달할 수가 있다. 이상의 분석으로부터 AND 게이트로 들어가는 입력노드 p1의 고착-1 결함과 노드 p2의 고착-1 결함을 제외한 고착 결함은 검출이 가능하다. 따라서, 회로의 모든 결함들에 대한 불검출 결함 확인은 AND 게이트로 들어가는 입력 노드들의 고착-1 결함에 대한 불검출 결함 확인만으로 제한하는 한편, 이들에 대한 조건인 $f_{xi}' \cdot f_{xi}'$ 혹은 $f_{xi}'' \cdot f_{xi}$ 의 해에 대한 확인을 BDD상에 직접적으로 연산을 수행한 결과로부터 확인할 수가 있어 기존 ATPG에 의한 방식보다는 효율적으로 검출할 수 있다[15]. 그러나 $f_{xi}' \cdot f_{xi}'$ 혹은 $f_{xi}'' \cdot f_{xi}$ 의 해에 대한 확인을 통한 불검출 결함 검출은 BDD로부터 추출된 멀티플랙서 구조 회로에만 적용이 가능하므로, 불검출 결함을 제거한 회로는 이들 관계를 통한 불검출 결함 확인이 불가능하다. 존재하는 불검출 제거를 위한 BDD의 다양한 형태 변환은 BDD에 대한 다양한 배열 순서에 의해 가능하나, BDD의 배열 순서에 따른 BDD의 크기가 지수함수적으로 증가할 수 있으므로 다양한 배열 순서를 통한 다양한 형태 변환은 불가능하다. 따라서 불검출 제거된 회로에 대하여 존재하는 불검출 결함은 시간적 효율성 측면에서 ATPG 방식에 의해 제거한다.



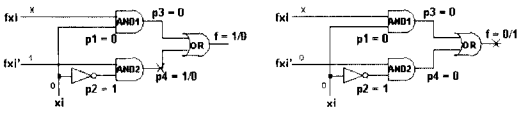
(a)



(b)



(c)



X : don't care
0/1 :결함이 없는 회로의 값 0/ 결함이 있는 회로의 값 1
1/0 :결함이 없는 회로의 값 1/ 결함이 있는 회로의 값 0

그림 3. 멀티플랙서에 대한 고착 결함 효과의 검출
(a) 주변입력 fxi,에 대한 고착-0 결함 효과의 검출 (b) AND1 게이트의 입력 p1에 대한 고착-0/고착-1 결함 효과의 검출 (c) AND2 게이트의 입력 p2에 대한 고착-0/고착-1 결함 효과의 검출 (d) AND2 게이트의 출력 p4에 대한 고착-1 결함 효과의 검출 (e) OR 게이트의 출력 p5에 대한 고착-0 결함 효과의 검출

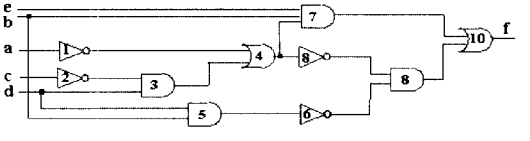
Fig. 3. Detection of fault at each node of a multiplexor.
(a) Detection of s-a-0 fault at input fxi, (b) Detection of s-a-0/s-a-1 fault at input p1 of AND1 gate (c) Detection of s-a- 0/s-a-1 fault at input p2 of AND2 gate (d) Detection of s-a-1 fault at output p4 of AND2 gate (e) Detection of s-a-0 fault at OR gate output

Ⅲ. 테스트 불가능한 경로 검출과 제거

1. 테스트 불가능한 경로 검출

1.1 ENF를 이용한 테스트 불가능한 경로 검출

임의의 다단 논리 회로에 대해서 각각의 경로 지연시간 결함을 검출하는 테스트 벡터들이 존재하는지를 확인하기 위해서 ENF를 이용한다[15,17]. ENF는 이단 논리 회로 형태의 큐브들로 구성되며 큐브들의 literal들은 다단 논리 회로들에 존재하는 경로들을 나타내는 태그들을 가진다. 이와 같은 특성을 가진 ENF를 이용하여 경로 지연시간 결함을 검출하는 테스트 벡터의 존재 조건을 다단논리 회로에서도 확인할 수가 있다. 임의의 회로에 대한 ENF으로 표현하기 위해 먼저 각각 주출력단에 대한 logic cone을 형성하고 logic cone을 구성하는 모든 노드들에 대한 게이트들을 AND, OR 게이트나 inverter로 분할하여 주출력단에 대해서 ENF를 구성한다. 그림 4는 예제 회로와 주출력단 f에 대한 ENF를 나타내었다. Inverter 1에 대한 ENF은 $a'_{\{1\}}$로 나타내며 AND 게이트 3에 대한 ENF은 $c'_{\{2,3\}} \cdot d_{\{3\}}$으로 나타낸다. OR 게이트 4에 대한 ENF은 $a'_{\{1,4\}} + c'_{\{2,3,4\}} \cdot d_{\{3,4\}}$로 나타내며 AND 게이트 9에 대한 ENF은 $a_{\{1,4,8,9\}} \cdot b'_{\{5,6,9\}} \cdot c_{\{2,3,4,8,9\}} + a_{\{1,4,8,9\}} \cdot b'_{\{5,6,9\}} \cdot d'_{\{3,4,8,9\}} + a_{\{1,4,8,9\}} \cdot c_{\{2,3,4,8,9\}} \cdot d'_{\{5,6,8\}} + a_{\{1,4,8,9\}} \cdot d'_{\{3,4,8,9\}} \cdot d'_{\{5,6,9\}}$로 나타낸다. Boolean identity 성질을 적용하지 않고 주출력단 f에 대한 ENF를 나타내면 6개의 큐브들로 구성되고, 그들의 literal은 주출력단에 이르는 각각의 경로를 나타낸다. 이와 같은 주출력단 f에 대한 ENF를 이용하여 경로에 대한 지연시간 테스트 벡터가 존재하는지를 확인할 수가 있다. 한 예로 큐브1인 $a'_{\{1,4,7,10\}} \cdot b_{\{7,10\}} \cdot e_{\{7,10\}}$는 다른 큐브들을 0으로 지정하면서 큐브1만을 0과 1로 지정하는 테스트 벡터가, {(a=0, b=0, c=1, d=0, e=1), (a=0, b=1, c=1, d=1, e=1)}, 존재하므로 지연시간 테스트가 가능하다. 따라서, 이 테스트 벡터를 이용하여 다른 경로들을 통해서 주출력단에 event들을 전달되지 않고 경로 {7,10} 만을 통하여 주출력단에 event를 전달된다. 그러나, 큐브6인 $a_{\{1,4,8,9,10\}} \cdot d'_{\{3,4,8,9,10\}} \cdot d'_{\{5,6,9,10\}}$는 큐브6만을 0또는 1로 지정하는 테스트 벡터들이 존재하나, 서로 다른 경로 {3,4,8,9,10}와 경로 {5,6,9,10}를 통하여 주입력단 d에서 발생한 event를 주출력단에 전달하므로 테스트 불가능한 경로가 된다. 큐브4인 $a_{\{1,4,8,9,10\}} \cdot c_{\{2,3,4,8,9,10\}} \cdot d'_{\{5,6,9,10\}}$, 큐브5인 $a_{\{1,4,8,9,10\}} \cdot b'_{\{5,6,9,10\}} \cdot d'_{\{3,4,8,9,10\}}$들은 이들에 대해서 1로 지정하는 테스트 벡터들에 대해서 항상 큐브6인 $a_{\{1,4,8,9,10\}} \cdot d'_{\{3,4,8,9,10\}} \cdot d'_{\{5,6,9,10\}}$에 대해서도 1로 지정한다.



(a)

큐브1 : $a'_{\{1,4,7,10\}} \cdot b_{\{7,10\}} \cdot e_{\{7,10\}}$
큐브2 : $b_{\{7,10\}} \cdot c'_{\{2,3,4,7,10\}} \cdot d_{\{3,4,7,10\}} \cdot e_{\{7,10\}}$
큐브3 : $a_{\{1,4,8,9,10\}} \cdot b'_{\{5,6,9,10\}} \cdot c_{\{2,3,4,8,9,10\}}$
큐브4 : $a_{\{1,4,8,9,10\}} \cdot c_{\{2,3,4,8,9,10\}} \cdot d'_{\{5,6,9,10\}}$
큐브5 : $a_{\{1,4,8,9,10\}} \cdot b'_{\{5,6,9,10\}} \cdot d'_{\{3,4,8,9,10\}}$
큐브6 : $a_{\{1,4,8,9,10\}} \cdot d'_{\{5,6,9,10\}} \cdot d'_{\{3,4,8,9,10\}}$

(b)

그림 4. 예제 회로에 대한 ENF를 나타낸 예
(a) 예제 회로 (b) 주출력단 f에 대한 ENF (큐브4⊂큐브6로 인한 테스트 불가능한 경로 {3,4,8,9,10}, 큐브5⊂큐브6로 인한 테스트 불가능한 경로 {5,6,9,10})

Fig. 4. ENF expression for an example circuit
(a) An example circuit (b) ENF expression for primary output f.

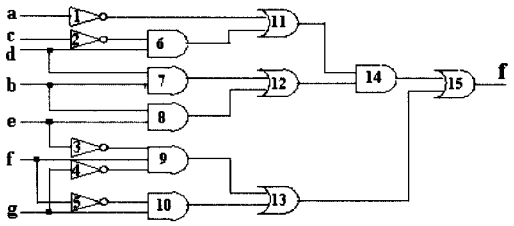
이와 같이 ENF를 이용하여 검출된 테스트 불가능한 경로들은 회로내 존재하는 재결합 경로이며 이들에 대한 ENF를 구성시에 특정한 큐브만을 1로 지정하는 모든 테스트 벡터들이 다른 큐브들을 1로 지정한다.

1.2 테스트 불가능한 경로의 효율적인 검출

테스트 불가능한 경로들을 검출하기 위해서 각각의 주출력단에 대한 ENF를 나타내는 경우는 모든 경로들에 대한 정보를 나타내는 다수의 큐브들로 구성되나, 테스트 불가능한 경로들을 검출하는데 관련없는 불필요한 큐브들이 대부분이다. 따라서, 앞 절에서 기술한 바와 같이 테스트 불가능한 경로들은 (i) 재결합 경로이고, (ii) 이들에 대한 ENF에서 특정한 큐브만을 1로 지정하는 모든 테스트 벡터들이 다른 큐브들을 1로 지정한다. 조건(ii)를 만족하는지 확인은 ENF를 구성하는 큐브들사이의 포함 관계를 통하여 가능하다. ENF를 구성하는 큐브들은 큐브들을 구성하는 경로들에 대하여 1로 지정할 수 있는 테스트 벡터의 조건을 나타내므로 한 큐브가 다른 큐브들에 포함되는 경우는 서로 다른 경로를 통하여 event가 전달되는 것을 의미하기 때문이다. 또한 BDD로부터 추출된 회로에 대해 parity가 다른 재결합 경로가 조건(i)과 조건(ii)를 만족하는 경우는 서로 다른 경로들을 통하여 반대값을 가진 결함 효과가 전달되므로 그 경로에 대해서 불검출 결함이 존재한다. 따라서 불검출 결함을 제거한 경우에는 (iii) 경로들에 대한 parity가 동일한 경로들에만 조건(i)과 조건(ii)를 확인하면 된다. 이와 같은 특성을 이용하여 BDD로부터 추출된 회로에서는 재결합 경로를 중심으로 ENF를 나타내는 경우에 보다 효율적으로 테스트 불가능한 경로를 검출할 수가 있다. 테스트 불가능한 경로를 검출하기 위해서 회로내에 존재하는 재결합 경로를 구하고 재결합 경로를 형성하는 각각 경로들에 대해서 parity를 확인하여 parity가 동일한 경우만에 제거 대상 집합에 포함시킨다. 여기서, 서로 다른 경로들이 병합되는 게이트를 reconvergent gate라고 정의하고 게이트의 출력이 2이상인 게이트를 divergent gate라고 정의할 때, 제거 대상 집합에 포함되는 경로들에 대한 reconvergent gate를 중심으로 ENF를 나타내며 이들을 이용하여 조건(ii)를 확인하여 테스트 불가능한 경로를 결정한다. 그림 5는 예제 회로에 대하여 주출력단에 대한 ENF와 재결합 경로에 대한 ENF를 나타내었다. 그림 5(b)와 같이 주출력단에 대한 ENF를 나타내는 경우에는 6개 큐브로 구성되나, 그림 5(c)와 같이 각각 재결합 경로에 대한 ENF를 나타내는 경우에 경로 {6,11,14}와 경로 {7,12,14}, 경로 {7,12}와 경로 {8,12}의 모든 재결합 경로가 동일한 parity를 유지하므로 이들에 대한 ENF를 나타내는데에 8개의 큐브들로 구성된다. 이와 같이 각각 재결합 경로에 대한 ENF를 나타내는 경우에 중복된 큐브들나 테스트 불가능한 경로 검출에 관련없는 불필요한 큐브들을 생성하여 비효율적이지만, 이들의 포함 관계를 이용하면 불필요한 정보들을 나타내는 큐브들을 생성하지 않고 효율적으로 나타낼 수 있다. 각각의 재결합 경로에 대한 ENF는 reconvergent gate에 대한 logic cone을 형성하고 이에 대해서 ENF를 나타내므로 형성된 logic cone이 다른 경로들에 대한 logic cone들에 포함되는 경우에는 포함하는 logic cone에 대한 ENF를 이용한다. 주출력단에 가까운 reconvergent gate일수록 그에 대한 logic cone은 크게 형성하므로 각각의 reconvergent gate가 다른 reconvergent gate에 대한 logic cone에 포함되는지를 확인하여 logic cone들사이의 포함 관계를 알 수 있다. 그림 5(d)는 예제 회로에 대한 재결합 경로를 효율적으로 ENF를 나타내었다.

모든 재결합 경로에 대해 주출력단에 가까운 reconvergent gate의 순으로 구성하면 경로 {6,11,14}와 경로 {7,12,14}, 경로 {7,12}와 경로 {8,12} 순으로 되며 경로 {7,12}와 경로 {8,12}에 대한 reconvergent gate인 게이트 12는 경로 {6,11,14}와 경로 {7,12,14}에 대한 logic cone에 포함되므로 경로 {6,11,14}와 경로 {7,12,14}에 대한 ENF를 이용한다. 이와 같은 방법으로 ENF를 나타내면 그림 5(c)와 같이 각각 재결합 경로에 대한 ENF를 나타내는 경우보다 효율적으로 테스트 불가능한 경로들을 검출할 수 있다. 그림 6는 재결합 경로를 이용하여 테스트 불가능한 경로를 제거하는 알고리듬을 나타내었다. 회로의 주출력단에 대한 ENF을 구성하지 않고, 먼저 재결합 경로를 구성하는 경로들사이의 parity를 확인하여 그들이 동일한 parity를 가진 경우만 mark한다. 이후에 mark된 재결합 경로들에 대한 logic cone들의 포함 관계를 비교하여 포함되는 logic cone에 대한 재결합 경로를 mark된 재결합 경로들에서 제거한다. 최종적으로 mark된 재결합 경로들에 대하여 ENF을 구성하여 효율적으로 테스트 불가능한 경로를 검출하고 이를 제거하여 테스트 용이

하도록 한다.



(a)

큐브1 : $a'_{(1,11,14,15)} \cdot b_{(7,12,14,15)} \cdot d_{(7,12,14,15)}$
큐브2 : $a'_{(1,11,14,15)} \cdot b_{(8,12,14,15)} \cdot e_{(8,12,14,15)}$
큐브3 : $b_{(7,12,14,15)} \cdot c'_{(2,6,11,14,15)} \cdot d_{(6,11,14,15)} \cdot d_{(7,12,14,15)}$
큐브4 : $b_{(8,12,14,15)} \cdot c'_{(2,6,11,14,15)} \cdot d_{(6,11,14,15)} \cdot e_{(7,12,14,15)}$
큐브5 : $e'_{(3,9,13,15)} \cdot f_{(9,13,15)} \cdot g'_{(4,9,13,15)}$
큐브6 : $f'_{(5,10,15)} \cdot g_{(10,15)}$

(b)

큐브1′: $a'_{(1,11,14)} \cdot b_{(7,12,14)} \cdot d_{(7,12,14)}$
큐브2′: $a'_{(1,11,14)} \cdot b_{(8,12,14)} \cdot e_{(9,12,14)}$
큐브3′: $b_{(7,12,14)} \cdot c'_{(2,6,11,14)} \cdot d_{(8,11,14)} \cdot d_{(7,12,14)}$
큐브4′: $b_{(8,12,14)} \cdot c'_{(2,8,11,14)} \cdot d_{(8,11,14)} \cdot e_{(7,12,14)}$
큐브5′: $b_{(7,12,14)} \cdot d_{(7,12,14)}$
큐브6′: $b_{(8,12,14)} \cdot e_{(7,12,14)}$

(c)

큐브1″: $a'_{(1,11,14)} \cdot b_{(7,12,14)} \cdot d_{(7,12,14)}$
큐브2″: $a'_{(1,11,14)} \cdot b_{(8,12,14)} \cdot e_{(9,12,14)}$
큐브3″: $b_{(7,12,14)} \cdot c'_{(2,6,11,14)} \cdot d_{(8,11,14)} \cdot d_{(7,12,14)}$
큐브4″: $b_{(8,12,14)} \cdot c'_{(2,8,11,14)} \cdot d_{(8,11,14)} \cdot e_{(7,12,14)}$

(d)

그림 5. 주출력단에 대한 ENF와 재결합 경로에 대한 ENF
(a) 예제 회로 (b) 주출력단에 대한 ENF (c) 재결합 경로들사이의 포함 관계를 고려하지 않고 ENF을 구성한 경우 (d) 재결합 경로들사이의 포함 관계를 고려하여 ENF를 구성한 경우

Fig. 5. ENF expressions for primary outputs and for reconvergent paths.
(a) An example circuit (b) ENF expressions for primary outputs (c) ENF expressions not considering the containment relation among reconvergent paths (d) ENF expressions considering the containment relation among reconvergent paths

```
    in REPi do
    if (Parities of Pi and Pj are equal) then
        mark REPj;
    end

for each pair of marked reconvergent paths
    REPi, REPj (REPi≠REPj) in N do
    begin
    if (logic cone of REPi contains
    logic cone of REPj) then
        unmark REPj;
    end

for each marked reconvergent paths REPi
in N do
  begin
    Construct a Equivalent Normal Form
    ENFi for REPi;
    Find and remove untestable paths in
    ENFi;
  end

end
```

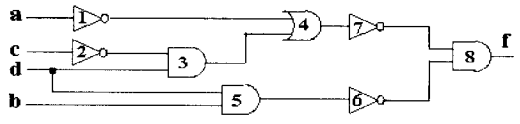
그림 6. 재결합 경로를 이용하여 테스트 불가능한 경로를 제거하는 알고리듬

Fig. 6. Algorithm for removing untestable paths using reconvergent paths.

2. 테스트 불가능한 경로 제거

Ⅲ.1.2절에서 기술한 바와 같이 ENF를 이용하여 검출되는 테스트 불가능한 경로들은 재결합 경로들이며, ENF를 구성하는 큐브들사이에 특정한 큐브만을 1로 지정이 불가능한 경로들이다. 따라서 테스트 불가능한 경로를 포함한 큐브들은 다른 큐브에 포함되는 관계를 가지거나, 한 큐브를 구성하는 동일한 literal들이 서로 다른 경로를 나타낸다. 테스트 불가능한 경로를 제거하기 위해서는 다른 큐브에 포함되는 큐브들과 둘 이상의 동일한 literal을 가진 큐브들을 제거한다. 다른 큐브에 포함되는 큐브들의 제거는 이들을 제외한 다른 큐브들에 변화를 주지 않으면서 테스트 불가능한 경로를 나타내는 literal들을 제거한다. 그러나 이들의 제거로 다른 큐브들에 변화를 주는 경우에는 테스트 불가

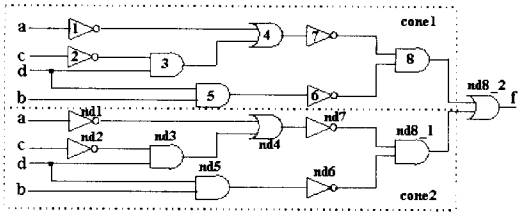
내는 큐브들이 동일하지 않아야 한다. 둘 이상의 동일한 literal들을 가진 큐브의 제거는 동일한 literal들중 한 경로만을 나타내는 literal만을 제외하고 모두 제거한다. 이와 같이 테스트 불가능한 경로를 제거하는 과정은 ENF상에서 나타내는 다른 큐브에 포함되는 큐브들과 둘 이상의 동일한 literal를 가진 큐브들을 대상으로 하면서 제거되는 literal들로 인한 큐브에 변화를 주는 경우에 logic cone을 복사하여 따로 나타내므로 전체적인 회로의 함수성에 변화를 주지 않는다.



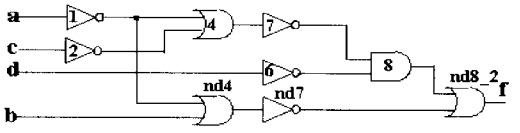
(a)

큐브1" : $a_{(1,4,7,8)} \cdot b'_{(5,6,8)} \cdot c_{(2,3,4,7,8)}$
큐브2" : $a_{(1,4,7,8)} \cdot c_{(2,3,4,7,8)} \cdot d'_{(5,6,8)}$
큐브3" : $a_{(1,4,7,8)} \cdot b'_{(5,6,8)} \cdot d'_{(3,4,7,8)}$
큐브4" : $a_{(1,4,7,8)} \cdot d'_{(5,6,8)} \cdot d'_{(3,4,7,8)}$

(b)



(c)



(d)

그림 7. 테스트 불가능한 경로를 제거하는 과정의 예
(a) 예제 회로 (b) 예제 회로에 대한 ENF (c) 테스트 불가능한 경로를 제거하기 위한 중간 회로 (d) 테스트 불가능한 경로를 제거한 결과

Fig. 7. An example removing untestable paths in circuit
(a) An initial circuit with untestable paths (b) ENF expression (c) Intermediate circuit obtained by duplication (d) Circuit obtained after removing untestable paths

그림 7은 예제 회로에 대한 불검출 경로를 제거하는 과정을 나타내었다. 그림 7(b)와 같이 예제 회로에 대한 ENF를 나타내었을 때 큐브2"와 큐브3"는 큐브4"에 포함되므로 테스트 불가능한 경로를 나타내는 큐브들의 literal들을 제거한다. 그러나 테스트 불가능한 경로 {3,4,7,8}과 경로 {5,6,8}에 대응되는 literal들을 모두 제거하는 경우에는 큐브4"에 변화를 주므로 그림 7(c)와 같이 테스트 불가능한 경로를 구성하는 경로의 수만큼 게이트 8에 대한 logic cone을 복사한다. 복사된 logic cone들에 대하여 큐브4"만을 나타내므로 하나의 logic cone만을 복사한다. 그림 7(d)는 그림 7(c)의 cone1에는 테스트 불가능한 경로 {3,4,7,8}과 경로 {5,6,8}을 모두 제거시에 큐브에 변화를 주지 않는 큐브1"을 나타내었으며 cone2에는 둘 이상의 동일한 literal을 가진 큐브4"를 테스트 불가능한 경로 {5,6,8}에 대응하는 literal d를 제거하여 나타내었다.

Ⅳ. 실험 결과

본 논문에서 제안한 경로 지연시간 결함의 효율적인 검출 방법을 UNIX 환경하에서 구현하였으며 성능 평가를 위해 MCNC/ISCAS benchmark들에 대한 실험 결과를 제시한다. 제안된 방법은 고착 결함과 경로 지연시간 결함에 대해 테스트 용이한 논리 합성 시스템의 일부로 사용되었으며, 테스트 불가능한 경로를 검출하는 기존의 알려진 방법[17]과 비교하였다. ISCAS benchmark 중의 C2670, C3540, C5315, C6288, C7552에 대해서는 BDD를 구성하는 과정에서 노드 수가 50,000개를 초과하여 실험 대상에서 제외하였다.

표 1은 기존의 방법에서 사용한 주출력단에 대한 ENF를 이용하여 테스트 불가능한 경로를 검출하는 과정과 재결합 경로에 대한 ENF를 이용하여 테스트 불가능한 경로를 검출하는 과정에 대하여 CPU 사용 시간과 메모리 사용량을 비교하였다. 주출력단에 대한 ENF를 이용하여 테스트 불가능한 경로를 검출하는 과정의 결과를 'PO-ENF'에서 제시하였으며, 재결합 경로에 대해 테스트 불가능한 경로를 검출하는 과정의 결과를 'Re-ENF'에 제시하였다. 평균적으로 재결합 경로에 대한 ENF을 이용하여 테스트 불가능한 경로를 검출하는 과정에서 사용되는 CPU 시간이 주출력단에 대한 ENF을 이용하여 테스트 불가능한 경로를 검출하는 과정에 비해 37.7% 감소하였으며, 메모리 사용량 측면에서는 60.9% 감소하는 결과를 보인다. 이는 주출력단에 대한 ENF을 이용하여 테스트 불가능한 경로를 검출하는 기존의 과정에서 부가적으로 생성되는 테스

트 불가능한 경로 검출과 관련없는 불필요한 정보들을 최소화하기 때문에 빠른 시간내에 적은 메모리량을 효율적으로 테스트 불가능한 경로를 검출한다. 특히 C499 회로에 대하여 재결합 경로를 이용하여 테스트 불가능한 경로를 검출하는 과정이 기존 방법보다 메모리 사용량 측면에서 85.6% 감소하였으며 CPU 사용시간 측면에서는 58.5% 감소하는 결과를 보인다. 테스트 불가능한 경로에 직접적으로 필요한 정보만을 회로로부터 추출하여 유지하므로써 이단 논리 회로 형태로 전환하는 ENF 구성시에 따른 사용된 CPU시간과 메모리량의 급증하는 문제점을 완화시키면서 효율적으로 테스트 불가능한 경로들에 대한 검출이 가능하게 된다.

표 1. 실행시간과 메모리 사용량에 대한 비교

Table 1. Run-time and memory usage comparisons.

Benchmark	#ins	#outs	#its	'PO-ENF'		'Re-ENF'	
				time(sec)	mem(KB)	time(sec)	mem(KB)
5xp1	7	10	293	14.9	44.4	9.5	23.6
9sym	9	1	516	25.2	127.7	13.1	42.6
con1	7	2	23	2.2	1.4	0.5	0.5
duke2	22	29	1775	48.6	990.0	32.2	289.2
rd73	7	3	843	12.3	29.7	8.2	23.1
rd53	5	3	140	5.3	3.4	4.3	2.0
C432	36	7	372	42.1	344.4	31.3	266.6
C499	41	32	616	53.3	1,234.3	22.1	176.7
C880	60	26	729	62.4	1,679.6	44.7	919.5
계				266.3	4,455.2	165.9	1,743.8
△(%)				-	-	-37.7	-60.9

'PO-ENF': 주출력단에 대한 ENF를 이용하여 테스트 불가능한 경로를 검출한 결과
'Re-ENF': 재결합 경로에 대한 ENF를 이용하여 테스트 불가능한 경로를 검출한 결과

표 2에서는 경로 지연시간 결함의 효율적인 검출 방법을 이용하여 테스트 용이한 논리 합성 결과와 기존의 테스트에 대한 고려없이 논리 합성한 결과를 비교하였으며, 이를 위해 실험에 사용한 MIS command script는 misII version 2.0에서 제공하는 standard script (~cad/lib/misII/lib/script)를 따랐다[21]. 결과 회로들의 면적을 비교하기 위해 SiLOS-II[22]를 이용하여 면적 최적화 모드로 기술 매핑을 수행하였다. SiLOS-II 시스템에서는 TECHMAP에서 사용되는 MCNC 라이브러리를 사용하며 literal 수를 기준으로 결과 회로의 면적을 계산한다. 표 2는 테스트의 고려없이 논리 최적화 과정을 수행한 결과와 테스트를 고려하여 최적화 과정을 거친 결과에 대하여 결과 회로의 면적, 고착 결함과 경로 지연시간 결함에 대한 테스트 패턴 생성 시간, fault coverage를 비교하였다. 테스트의 고려없이 최적화 과정을 수행한 결과는 'w/o Testability' 로 표시된 행에 제시하였으며, 테스트를 고려하여 최적화 과정을 거친 결과는 'Testable Design'행에 제시하였다. 이 표에 따른 전체적인 회로에 대한 결과는 테스트를 고려한 과정을 거친 경우가 테스트 고려하지 않은 경우에 비하여 평균적으로 고착 결함인 경우 테스트 패턴을 산출하는 시간이 38.0% 감소하였으며, 경로 지연시간 결함인 경우 테스트 패턴을 산출하는 시간이 27.3% 감소하였다. 고착 결함에 대한 결과를 살펴보면 C432 회로인 경우 초기의 회로에 대한 fault coverage가 90.9% 였으나, 테스트의 고려없이 최적화 과정을 수행한 경우 85.6%로 감소하는데 비해 최적화 과정을 수행한 후 테스트를 고려한 과정을 거친 경우 100%로 향상되었다. 이는 기술 독립적인 최적화 과정에서 결과 회로에 대한 fault coverage에 영향을 미치나, 반드시 fault coverage의 향상시키지 않음을 알 수 있다. 또한 테스트의 고려없이 최적화 과정을 수행하여 fault coverage가 100%인 회로들에 대해서는 테스트를 고려하여 최적화 과정을 거친 결과가 테스트 패턴을 생성하는 시간을 단축시키는 결과를 나타내었다. 경로 지연시간 결함에 대한 결과를 살펴보면 테스트 고려없이 최적화 과정을 수행한 경우 대부분이 80% 미만의 매우 낮은 경로 지연시간 결함에 대한 fault coverage를 가지는데 비해 테스트를 고려하여 최적화 과정을 거친 경우는 모두 90% 이상의 향상된 경로 지연시간 결함에 대한 fault coverage를 가진다. 특히, C880회로에서 최적화 과정만을 수행한 경우는 경로 지연시간 결함에 대한 fault coverage가 33.5%인데 비하여 테스트를 고려하여 최적화 과정을 거친 결과는 98.0%로 향상되었다.

테스트에 대한 고려없이 최적화된 임의의 이단 논리 회로에 대하여 경로 지연시간 결함에 테스트 용이성을 향상시키기는 경우에 회로를 Shannon의 확장식 형태로 변형하는 방법을 이용한다[23]. 이와 같은 회로 변환시에 최적화된 면적과 동작 시간을 증가시키는 문제점을 안고 있으나, 구현된 방식에서는 BDD를 이용하

므로써 면적과 동작시간의 증가량을 최소화하여 테스트 고려없이 최적화 과정을 수행한 결과에 비하여 SOP 표현시 literal의 수에서는 14.8% 증가되는 실험 결과를 보인다.

표 2. 면적과 테스트 용이성에 대한 결과
Table 2. Area and testability comparison.

Benchmark	'w/o Testability'					'Testable Design'				
	#lits/ Area	SF Cov.	SF TPG	PF Cov.	PF TPG	#lits/ Area	SF Cov.	SF TPG	PF Cov.	PF TPG
5xp1	154/ 112,301	100	11.4	70.2	519.4	203/ 154,699	100	5.3	93.2	320.3
9sym	272/ 183,290	100	34.2	82.3	283.1	323/ 244,031	100	22.2	100	194.1
con1	22/ 20,214	100	3.1	97.4	82.5	20/ 19,553	100	3.0	100	55.2
duke2	673/ 351,482	100	30.1	37.7	829.1	739/ 524,584	100	22.9	100	521.5
rd73	196/ 154,311	100	22.1	55.2	321.4	302/ 245,522	100	12.3	98.6	219.4
rd53	59/ 30,516	100	12.5	70.4	91.2	120/ 81,303	100	19.4	100	44.5
C432	342/ 523,102	85.6	54.2	60.3	750.8	352/ 395,578	100	15.3	100	546.3
C499	604/ 695,511	99.6	42.1	45.4	881.9	642/ 732,200	100	22.3	93.6	772.1
C880	447/ 384,325	100	25.6	33.5	1186.4	481/ 434,129	100	23.2	98.6	921.6
계	2,768/ 2,455,052	-	235.5	-	4945.8	3,182/ 3,031,499	100	145.9	-	3594.2
△(%)	-	-	-	-	-	+14.9/ +23.5	-	-38.0	-	-27.3

#lits : SOP 표현시 literal의 수
Area : 기술매핑한 결과 회로의 면적
SF Cov.: 고착 결함에 대한 fault coverage
SF TPG : 고착 결함에 대한 테스트 패턴 생성 시간
(Unit: sec on Sun Sparc Station1+)
PF Cov.: 경로 지연시간 결함에 대한 fault coverage
PF TPG : 경로 지연시간 결함에 대한 테스트 패턴 생성 시간
'w/o Testability' : 테스트 고려없이 최적화 과정을 수행한 결과
'Testable Design' : 테스트 고려하여 최적화 과정을 수행한 결과

또한, 경로 지연시간 결함에 대하여 테스트 용이하도록 합성한 기존의 방법은 회로에 대한 ENF를 이용하여 테스트 불가능한 경로를 제거하면서 literal의 수를 감소시키는 과정들을 수행하여 면적 측면에서 보다 감소하는 결과를 나타내나, 제안된 방식에 비하여 향상된 fault coverage를 가지지 못하는 결과를 보인다. 특히 경로 지연시간 결함에 대한 테스트 용이한 논리 합성 방법[17]은 구현된 방식에 비하여 결과 회로 rd53과 rd73의 literal 수가 각각 59, 137로 보다 적은 면적을 차지하나 fault coverage측면에서는 각각 100%, 93.4%로 생성하여 보다 낮은 결과를 나타낸다. 테스트 고려없이 최적화 과정을 수행한 결과에 대하여 각각의 결함에 테스트 용이하도록 하는 경우는 테스트를 고려하여 최적화 과정을 수행한 결과보다 면적과 동작시간이 증가할 것이다. 그림 8은 MCNC benchmark의 하나인 misex1 회로에 대한 실험 결과를 보인다. 그림 8(a)는 초기 회로를, 그림 8(b)는 멀티플랙서 구조로 변환된 회로를 나타내었다. 그림 8(c)는 멀티플랙서 구조로 변환된 회로의 불검출 결함을 제거하여 얻게 되는 결과를, 그림 8(d)는 불검출 결함을 제거한 후 테스트 불가능한 경로를 제거한 최종 결과를 나타내었다.

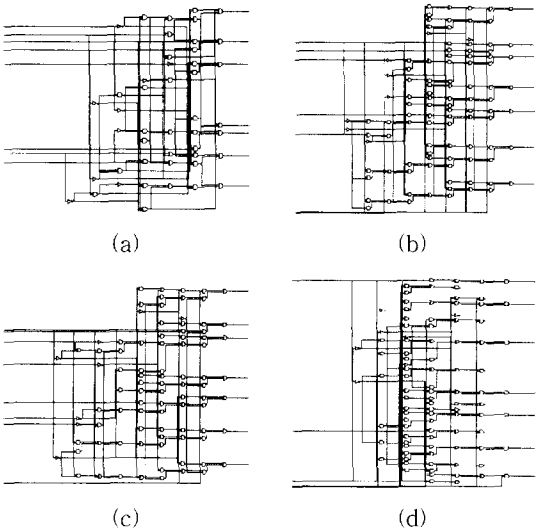


그림 8. misex1 회로에 대한 실험 결과
(a) 초기 회로 (b) 멀티플랙서 구조로 변환된 회로 (c) 회로(b)에 대한 불검출 결함을 제거한 결과 회로 (d) 회로(c)에 대한 테스트 불가능한 경로를 제거한 최종 결과 회로

Fig. 8. Experimental result for 'misex1' circuit.
(a) An initial circuit (b) Multiplexor-based circuit (c) Circuit obtained after removing all redundant faults (d) Circuit obtained after removing untestable paths

VI. 결론 및 추후 과제

본 논문에서는 다단 회로에서 테스트 불가능한 경로

를 효율적으로 검출하는 알고리듬을 제안하였으며, 제안한 방법은 주출력단에 대한 ENF를 이용하여 테스트 불가능한 경로를 검출하는 기존의 과정에 비해 CPU 사용 시간과 메모리 사용량을 감소시키는 실험 결과를 나타낸다. 이는 주출력단에 대한 ENF를 이용하여 테스트 불가능한 경로를 검출하는 기존의 과정에서 부가적으로 생성되는 테스트 불가능한 경로 검출과 관련없는 불필요한 정보들을 최소화시키기 때문이다. 따라서 ENF 구성시에 따른 CPU 사용 시간과 메모리 사용량의 급증하는 문제점을 완화하는 한편 테스트 불가능한 경로를 검출하는 과정에서 필요한 정보만을 유지하므로 보다 효과적으로 검출한다. 또한 제안한 방법은 테스트 용이한 논리 회로 합성 시스템의 일부로서 사용하여 논리 합성한 결과, 테스트를 고려하여 최적화 과정을 수행한 경우 테스트를 고려하지 않는 경우에 비하여 최적화된 면적과 동작시간을 증가시키나, 회로내에 존재하는 불검출 결함과 테스트 불가능한 경로들을 제거하여 fault coverage를 향상시키고 테스트 패턴 생성 시간을 단축시키는 결과를 나타내었다.

제안한 방법은 기존의 방법에 비하여 회로에 대한 재결합 경로를 추출하고 추출된 모든 재결합 경로들에 대해 테스트 불가능한 경로인 조건을 확인하는 전처리 과정이 필요하므로 보다 효율적인 전처리 과정에 대한 연구가 요구된다. 또한 제안한 방법을 이용하여 테스트 용이한 논리 합성하는 시스템에서는 대상 회로를 멀티플랙서 구조 회로로 변환하는 과정과 변환된 멀티플랙서 구조 회로에 대한 초기의 불검출 결함을 확인하는 과정에서 사용되어지는 BDD의 크기는 생성된 결과 회로의 면적 증가에 직접적으로 영향을 미치며 초기의 불검출 결함을 확인하는 데 소요되는 시간을 좌우하므로, 임의의 회로에 대해서 구성되는 BDD의 크기를 최소화하는 추가적인 연구가 필요하다.

참 고 문 헌

[1] R. K. Brayton et al., "MIS: A Multiple-level Logic Optimization System," IEEE Trans. on CAD, vol. CAD-6, no. 6, Nov. 1987, pp. 1062-1081.

[2] R. K. Brayton et al., "Multilevel Logic Synthesis," IEEE Proceedings, vol. 78, no. 2, Feb. 1990, pp. 264-300.

[3] K. Bartlett et al., "Synthesis and Optimization of Multilevel Logic under Timing Constraints," IEEE Trans. on CAD, Vol. CAD-5, No. 4, Oct. 1988, pp. 582-596.

[4] E. Detjens et al., "Technology Mapping in MIS," in Proc. ICCAD, Nov. 1987, pp. 116-119.

[5] M. Abramovich, M. Breuer and A. Friedman, "Digital Systems Testing and Testable Design," Computer Science Press, 1990.

[6] S. Devadas, T. Ma, A. Newton and A. Sangiovanni-Vincentelli, "Optimal Logic Synthesis and Testability : Two Faces of the Same Coin," in Proc. Int. Test Conf., Sept. 1988, pp. 3-13.

[7] R. Jacoby, P. Maceyunas, H. Cho and G. Hachtel, "New ATPG Technique for Logic Optimization," in Proc. ICCAD, Nov. 1989, pp. 548-551.

[8] S. Devadas and K. Keutzer, "Design of Integrated Circuits Fully Testable for Delay Faults and Multifaults," in Proc. Int. Test Conf., June 1990, pp. 284-293.

[9] S. Devadas and K. Keutzer, "Synthesis and Optimization Procedures for Robustly Delay-Fault Testable Logic Circuits," in Proc. 27th Design Automation Conference, June 1990, pp. 221-227.

[10] K. Roy, K. De, J. Abraham, and S. Lusky, "Synthesis of Delay Fault Testable Combinational Logic," in Proc. ICCAD, Nov. 1989, pp. 418-421.

[11] K. A. Bartlett, R. Brayton, G. Hachtel, R. Jacoby, C. Morrison, R. Rudell, A Sangiovanni-Vincentelli and A. Wang, "Multilevel Logic Minimization Using Implicit Don't Cares," IEEE Trans. on CAD, vol. CAD-7 no. 6, June 1988, pp. 723-740.

[12] D. Brand, "Redundancies and Don't Cares in Logic Synthesis," IEEE Trans. on Computers, vol. C-32, no. 10, Oct. 1983, pp. 947-952.

[13] J. P. Roth, "Minimization Using the D-algorithm," IEEE Trans. on Computers, vol. C-35, no. 5, May 1986, pp. 476-484.

[14] N. Jha and S. Kundu, "Testing and Reliable Design of CMOS Circuits," Kluwer Academic Publishers, 1990, pp. 14-24.
[15] C. Sequin, "Advanced Research in VLSI," MIT Press, 1991, pp. 35-54.
[16] D. Armstrong, "On Finding a Nearly Minimal Set of Fault Detection Tests for Combinational Logic Nets," IEEE Trans. on Computers, vol. C-15, no. 2, Feb. 1966, pp. 66-73.
[17] S. Devadas and K. Keutzer, "Synthesis of Robust Delay-Fault-Testable Circuits: Theory," IEEE Trans. on CAD, vol. 11, no 1, Jan. 1992, pp.87-101.
[18] S. Akers, "Binary Decision Diagrams," IEEE Trans. on Computers, Vol. C-27, No. 6, June 1978, pp. 509-516.
[19] C. Y. Lee, "Representation of Switching Circuits by Binary Decision Programs," Bell System Technical Journal, vol. 38, July 1959, pp. 985-999.
[20] R. Bryant, "Graph-Based Algorithm for Boolean Function Manipulation," IEEE Trans. on Computers, vol. C-35, no. 8, Aug. 1986, pp 677-691.
[21] R. K. Brayton, E. Detjens, S. Krishna, T. Ma, P. McGeer, L. Pei, N. Phillipse, R. Rudell, R. Segal, A. Wang, R. Yung, A. Sangiovanni-Vincentelli, "Multiple-level Logic Optimization Systems," in Proc. ICCAD, Nov. 1986, pp. 356-359.
[22] 김태선, 황 선영, "논리 회로의 기술 매핑 시스템 설계", 대한 전자 공학회 논문지 29-A권, 2호, 1992년 2월, pp. 147-158.
[23] N. Jha and S. Kundu, "Testing and Reliable Design of CMOS Circuits," Kluwer Academic Publishers, 1990, pp. 158-171.

저 자 소 개

黃 善 泳(正會員)

1976년 2월 서울대학교 전자공학과 졸업. 1978년 2월 한국과학원 전기 및 전자공학과 공학석사 취득. 1986년 10월 미국 Standford 대학 공학박사 학위 취득. 1976년 ~ 1981년 삼성 반도체 주식회사 연구원. 1986년 ~ 1989년 Stanford 대학 Center for Integrated Systems 연구소 책임연구원, Fairchild Semiconductor Palo Alto Reserach Center 기술자문. 1989년 ~ 1992년 삼성전자(주) 반도체 기술자문. 1989년 3월 ~ 현재 서강대학교 전자공학과 부교수. 주관심분야는 CAD 시스템, Computer Architecture 및 Systems Design, VLSI 설계 등임

許 勳(正會員)

1969년 3월 20일생. 1992년 2월 서강대학교 전자공학과 졸업. 1994년 8월 서강대학교 전자공학과 공학석사 취득. 현재 서강대학교 전자공학과 공학박사과정 재학중. 주관심분야는 CAD 시스템, Testable Logic Synthesis, Logic Synthesis for Low Power 등임