상위·하위 수준에서 통합된 테스트 합성 기술의 개발

(Development of Unified Test Synthesis Technique on High Level and Logic Level Designs)

신 상 훈 † 송 재 훈 †† 박 성 주 †††
(Sang-hoon Shin) (Jae-hoon Song) (Sungju Park)

요 약 칩의 집적도에 비례 하여 설계검증 및 칩 제작 후의 결함점검은 갈수록 어려워지며 이러한 테스트 문제의 원초적 해결을 위하여 다양한 테스트설계 기술이 널리 개발되고 있다. 상위 수준의 테스트설계에서는 회로의 기능에 대해서는 알 수 있으나 구조에 대해서는 알 수 없고, 하위 수준의 테스트설계에서는 회로의 구조를 알 수 있으나 기능은 알 수 없다. 따라서 테스트 설계는 기능을 기술하는 상위 수준에서부터 고려되어 하위 게이트수준에서 스캔플립플롭을 선택하여야 최적화된 성능을 얻을 수 있다. 본 논문에서는 테스트용이도를 증진시키기 위해, 상위수준의 기능정보에 대해서는 테스트점을 삽입하여 제어흐름(control flow)을 변경하고, 상위 수준의 합성 후에 하위 수준에서 스캔플립플롭을 선택하여 다시 합성하는 상위·하위 수준에서 통합된 테스트 합성 기술을 제안한다. 실험결과 통합된 테스트 합성 기술이 대부분의 벤치마크 회로에서 높은 고장검출율을 보여주고 있다.

Abstract Design verification and VLSI testing have become more difficult as the complexity of VLSI circuits drastically increases. Various design for testability(DFT) techniques have been developed to resolve the problems. The circuit behavior level information is available in the high level description and the gate level information is achievable through the high level synthesis. In this paper, the testing problem is resolved by choosing scan flip-flops in the logic level after applying high level DFT techniques on the behavioral level information described by hardware description languages. Experimental results show that the fault coverages for most of the benchmark circuits are highly improved.

1. 서 론

칩의 집적도에 따라 설계검증 및 칩제작 후의 기능점검 등은 더욱 더 어려운 문제로 부각되고 있다. 그래서 테스트 문제의 원초적인 해결을 위하여 다양한 테스트 설계 기술이 널리 개발되고 있다. 상위수준에서의 테스트합성 기술은 스캔설계를 위하여 루프(loop)를 최소화하기 위한 register-module binding방법[1], 스캔설계 대신 Ad-Hoc 방법으로 추가영역을 줄이는 방법[2], 제어기의 기능파악으로 초기화시키기 힘든 플립플롭을 대상으로 스캔하는 방법 등이 Register Transfer Level (RTL)에서 개발되었다. 또한 상위·하위의 테스트설계 결과를 비교·분석한 연구[3]도 발표되었다. 상용화된 CAD 도구에서도 테스트패턴 생성 및 테스트설계 기능을 상위수준 및 하위수준에서 별도로 행하고 있다. 일반적으로 제어부(controller) 지배적인 회로에서는 데이타 통로(data path) 보다는 제어부 부위에 테스트 기능을 추가하고[4] 역으로 데이타 지배적인 회로에서는 데이타 통로 부위에 테스트 기능을 추가하는 방법이 최적의 효과를 얻을 수 있다[5]. 상위수준인, VHDL 코드를 분석하여 테스트용이도를 증진시킨 연구가 보고되었지만 실용성과 체계성에 문제가 있었다[6-8]. 상위수준에서

· 본 연구는 과학재단 특정기초연구(과제번호:2000-1-30200-002-3)의 지원을 받아서 수행하였습니다.
† 비 회 원 : 삼성전자 반도체 network부 연구원
cyberimp@woongjin.com
†† 학생회원 : 한양대학교 컴퓨터공학과
jhsong@mslab.hanyang.ac.kr
††† 종신회원 : 한양대학교 컴퓨터공학과 교수
parksj@malab.hanyang.ac.kr
논문접수 : 2000년 1월 19일
심사완료 : 2001년 3월 23일

부분스캔 구현시 세 개의 플립플롭만 스캔하면 되는데 상위 합성 후 생성된 게이트 회로에서는 이십 개의 플립플롭을 스캔해야 동일한 점검도를 이룰 수 있는 경우가 보고되었다[3,9-11]. 즉, 게이트수준에서 행해지고 있는 대부분의 체계적인 테스트설계 기술은 쉽게 적용할 수는 있지만 상위수준 정보와의 결합 없이는 최적의 방법이라고 볼 수 없다. 상위수준에서의 테스트 합성은 물리적 결함에 대한 기능적 고장모델의 설정이 어렵지만 제어부 및 데이타 통로 등의 기능정보와 구조정보가 복합적으로 포함되어 있다.

본 논문에서는 VHDL로 기술된 회로의 제어문 내부 변수들의 조정도 및 랜덤도를 랜덤패턴 시뮬레이션에 의하여 산정한 후 제어문 내부의 변수에 대하여 방문 회수에 따른 상대적 랜덤도와 조정도를 산출하고 AND, OR 및 XOR 형태의 테스트점을 삽입 논리합성 후 고장검출율을 증진시키는 기술을 제안한다.

실험을 통하여 상위에서부터 관측된 정보와 하위 수준에서의 스캔 설계가 통합되는 것이 더 높은 고장검출율을 갖는다는 사실을 확인하였다.

본 논문은 다음과 같은 순서로 구성되어 있다. 2장에서는 상위 수준에서의 테스트 설계에 대하여 테스트점 삽입 기술을 소개하고, 3장에서는 하위 수준에서의 스캔 설계에 대해 소개한다. 4장에서는 상위와 하위 수준에서 통합된 테스트 설계하여 고장검출율을 증진시킬 수 있는 기술을 살펴보고, 5장에서는 실험결과를 소개하며 끝으로 결론 및 향후 연구계획에 대해 기술한다.

2. 상위 수준에서의 테스트 설계

그림 1은 본 논문에서 삽입할 4가지 다른 종류의 테스트점을 보여주고 있다. C1은 0-조정도를 증진시킬 수 있는 AND 형태의 테스트점으로서 반복문이나 조건문에 대해 강제적으로 거짓을 인가 할 수 있고, C2는 1-조정도를 증진시킬 수 있는 OR 형태의 테스트점으로서 강제적으로 참을 인가 할 수 있으며, C3는 true와 false 문 내에서 변수의 조정도를 반전시킬 수 있는 XOR 형태의 테스트점을 그리고 C4는 변수에 필요한 값을 직접 입력받을 수 있는 테스트점을 나타낸다.

최적의 테스트점을 선택하는 방법은 다음과 같다. 상위수준의 VHDL 코드에서 문장 목록에 따라 Control Flow Graph(CFG)를 생성한다. CFG상의 통로를 분석하여 조정도에 따라 완전 제어가능 변수와 불완전 제어가능 변수를 나누고 불완전 제어가능 변수를 줄이기 위한 최소한의 제어점을 찾는다.

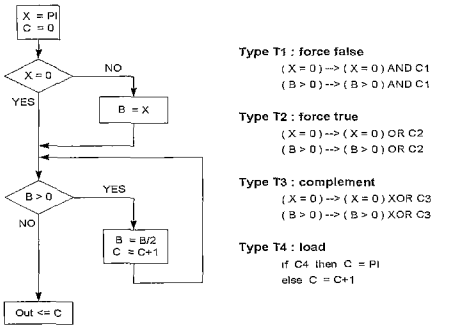


그림 1 테스트점 삽입

그러나 불완전 통로를 찾는 일은 NP-complete 문제이며 [6]에서 제안하는 발견적인(heuristic) 방법도 너무 많은 계산량을 필요로 하여서 실용적이지 못하다.

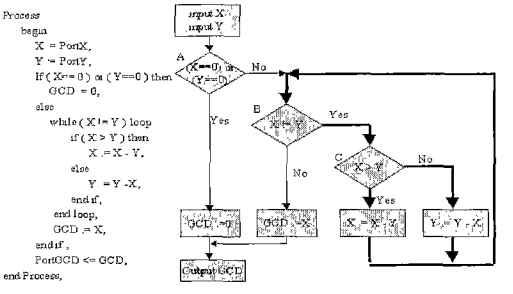


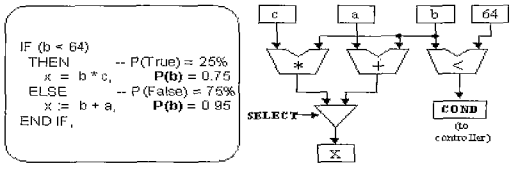
그림 2 GCD 회로의 상위수준 기술 코드 및 CDFG

그림 2는 최대공약수(GCD)를 구하는 회로의 상위수준인, VHDL 코드와 Control Data Flow Graph (CDFG)를 보여주고 있다.

CDFG에서 A 및 C는 if-then-else를 B는 while 루프의 조건문을 나타낸다. 상위수준의 기능정보에서 제어흐름을 증진시켜서 테스트용이도를 높이는 연구가 최근에 활발히 진행되고 있다[6-8]. 이런 연구에 대해 간략히 설명하면 다음과 같다.

먼저 입력 단에서 출력 단에 이르는 모든 통로(path)를 열거하고 변수를 추출하며 변수(레지스터)의 크기와 정당화(Justification) 가능도를 통로에 기준 하여 측정 전체 테스트용이도를 높였다[6]. 또 While 루프만을 고려하여 입력포트까지의 상태문장 개수를 비교하여 최대 개수를 필요로 하는 B의 while 루프에 테스트점을 삽입하였다[7]. If-then-else를 고려하여 A 혹은 C의 제어문에 XOR 테스트점을 추가하였다[8].

본 논문에서는 변수의 정당화 가능도 대신에 랜덤패턴 시뮬레이션에 의하여 조정도 및 랜덤도를 산출하며 while 루프만이 아닌 if-then-else문까지 고려하고 AND, OR 및 XOR 가운데 최적의 테스트점을 선별 삽입하도록 한다. 레지스터 단위의 조정도 및 랜덤도는 랜덤패턴 시뮬레이션을 통하여 각 비트 단위의 조정도와 레지스터 단위의 랜덤도를 산출하여 계산한다. 최대의 테스트용이도 이윤을 계산하기 위하여 본 논문에서는 상대적 레지스터 조정도 및 상대적 레지스터 랜덤도를 정의하여 사용하였다. 설명을 돕기 위하여 그림 2의 소스코드에서 반복문 내부의 if-then-else문을 그림 3. (a)처럼 변경하여, 조건제어문 내에 관심 있는 변수 b를 가지고 테스트점을 삽입해 보도록 한다.



(a) Behavior (b) Datapath

그림 3 기능도 및 데이타 통로도

표 1에서 보여주는 바와 같이 랜덤패턴 시뮬레이션을 통하여 b 레지스터의 각 비트가 갖는 0과 1의 비율을 관찰하고, 각 비트에 대한 0-조정도와 1-조정도를 산출해낸다. b 레지스터의 0-조정도는 각 비트의 0-조정도 합인 5이며 1-조정도는 3이 되므로 1-조정도를 높일 필요가 있다. 만약 조정도만을 고려한다면 레지스터의 모든 비트가 0만 되거나 혹은 1이 되어도 0-조정도와 1-조정도가 균등하게 되는 경우가 발생한다. 여기에 랜덤도가 추가된다면 레지스터가 다양한 값을 취하면서 비트 단위로 균등한 조정도를 유지할 수 있게 되므로 테스트 관점에서 더욱 적합하게 될 것이다. 그림 3의 예에서는 그림 3. (a)의 조건문에 OR 테스트점을 삽입하여 1-조정도 및 랜덤도를 증진시킬 수 있었다. 테스트점을 삽입하게 되면 while loop등의 반복문이나 조건문의 내부변수의 값을 출력변수에 보다 쉽게 전달할 수 있게되므로 조정도만이 아니라 관측도 역시 증진이 된다. 본 논문에서는 관측도의 개선량을 테스트용이도에 포함하기 위하여 상대적 레지스터 조정도 및 랜덤도를 정의하여 사용하였다. 상대적이란 의미는 테스트점을 삽입하기 이전에 비하여 삽입 후에 어느 정도 루프 등의 방문 횟수가 줄어들었는지를 수치적으로 나타내는 것으로 이는 내부 변수 신호가 출력변수에 어느 정도 더 빨리 전달될 수 있는지를 나타내는 것이다.

표 1 레지스터의 각 비트에 대한 0 및 1 조정도

	b7	b6	b5	b4	b3	b2	b1	b0
0 조정도	1.0	1.0	0.5	0.5	0.5	0.5	0.5	0.5
1 조정도	0.0	0.0	0.5	0.5	0.5	0.5	0.5	0.5

테스트점을 삽입할 위치 선정과 종류를 결정하기 위하여 상대적 레지스터 조정도 및 랜덤도를 다음과 같이 정의하였다.

- RRC : 상대적 레지스터 조정도(Relative Register Controllability)
- RRR : 상대적 레지스터 랜덤도(Relative Register Randomness)
- OC : 테스트점 삽입 전의 방문 회수(Original Counter)
- TC : 테스트점 삽입 후의 방문 회수(Test mode Counter)
- RC : 레지스터 조정도(Register Controllability)
  : $\sum 0.5 - (1\text{ 조정도})$
- RR : 레지스터 랜덤도(Register Randomness)
  : 독립적인 값의갯수 / $2^{\text{레지스터 길이}}$ $(0 < RR \leq 1)$

$$RRC = \sum_{i=1}^{k} \left\lceil \frac{TC}{OC} \times RC \right\rceil \quad (i=1,2,3,\ldots,k) \qquad (1)$$

$$RRR = \sum_{i=1}^{k} \left\lceil \frac{OC}{TC} \times RR \right\rceil (i=1,2,3,\ldots,k) \qquad (2)$$

식 1, 2는 K개의 각각의 변수에 대한 상대적 조정도 및 랜덤도를 계산하는 식이다. RRC에서 RC는 0에 근접할 때 0-조정도와 1-조정도가 균형을 이룸을 의미하고, TC/OC는 관측도의 개선 정도를 역으로 나타내며 테스트점 삽입 후에 반복문 등의 방문 회수가 줄어들면(즉 TC값이 감소하면) 주 출력단에서 더 빨리 값을 관측할 수 있게됨을 의미한다. RRR에서 RR은 변수가 가질 수 있는 모든 값 가운데 실제로 나타난 값의 비율로써 1에 가까울수록, 그리고 관측도의 개선정도를 나타내는 OC/TC는 클수록 좋다. 원래의 회로 및 테스트점 삽입에 의하여 변경된 회로에 대한 RRC와 RRR이 계산되면 식 3, 4와 같이 변경에 따른 이익을 계산하여 두 이익의 합이 최대인 곳에 테스트점을 삽입하도록 한다. 두 이익의 합이 최대인 곳이 회로 전체의 관점에서 최적의 테스트점인 것이다.

$$PFC(\text{조정도에의한이익}) = \sum_{\text{상대적조정도(변경전)}} - \sum_{\text{상대적조정도(변경후)}} \qquad (3)$$

$$PFR(\text{랜덤도에의한이익}) = \sum_{\text{상대적 랜덤도(변경후)}} - \sum_{\text{상대적 랜덤도(변경전)}} \quad (4)$$

시간 복잡도는 $O(kn^{r+1})$로서 k는 반복문의 개수, r은 각 반복문 안에 포함된 반복문의 개수, n는 반복문의 반복횟수이다. 랜덤패턴 시뮬레이션이 끝나면 연산에 소요되는 시간은 선형으로 거의 무시할 정도이다.

3. 하위 수준에서의 테스트 설계

부분스캔은 순차회로에서 최소의 비용으로 최대의 고장검출율을 목표로 하고 있다. 따라서 어떤 플립플롭을 스캔플립플롭으로 선택하느냐가 가장 중요한 문제이다. 일반적으로 테스트용이도를 스캔플립플롭 선택에 이용할 경우 각 플립플롭이 스캔되었을 때의 이익(gain)을 계산하여 가장 큰 이익을 내는 플립플롭을 선택하게 된다.

테스트용이도에 의한 부분스캔은 순차회로에서 모든 노드의 조정도 및 관측도를 계산하여 가장 효율적인 플립플롭을 스캔플립플롭으로 선택한다. 최적화 기법을 이용한 연구[9]에서는 모든 노드의 테스트용이도를 계산한 후 비용함수와 이익함수를 이용하여 최소의 비용으로 최대의 이익을 얻을 수 있는 플립플롭을 선택함으로써 고장검출율이 높은 부분스캔을 설계하였다.

하위수준의 테스트설계는 [14]의 방법을 그대로 적용하였으며 개략적으로 정리하면 다음과 같다.

테스트용이도 계산 결과 0-조정도 및 1-조정도 값이 크면 클수록 그 노드에 논리값 '0' 또는 '1'을 인가하기가 힘들다는 것을 나타내고 관측도가 크면 클수록 그 노드의 값이 주출력단으로 전달되기 힘들어진다는 것을 의미한다. 각 플립플롭의 스캔이익은 회로 전체의 테스트용이도의 합에서 그 플립플롭을 스캔한 후 테스트용이도 합의 차로 정의된다.

테스트용이도에 의해서만 스캔플립플롭을 선택할 경우에는 모든 플립플롭의 스캔이익을 구해야 된다. m개의 플립플롭이 회로에 존재할 때 m번의 이익함수 계산이 필요하게 된다. 각 플립플롭의 스캔이익을 계산하기 위해서는 매번 회로에 있는 모든 셀에 대해 0-조정도 및 1-조정도와 관측도를 계산해야 된다. 따라서 계산량이 아주 많아지게 된다.

알고리듬 1은 k개의 플립플롭을 스캔플립플롭으로 선택하기 위한 알고리듬을 보여주고 있다. 테스트용이도에 의해 스캔플립플롭을 선택하는 알고리듬의 복잡도를 구해보면 $O(n^4)$가 된다.

구조분석에 의해서 스캔플립플롭을 선택할 경우에는 순차회로를 그래프로 모델링하여 구조의 복잡도를 분석

알고리듬 1 테스트 용이도에 의한 스캔플립플롭 선택

```
compute testabilities;
compute testability-sum;
select k number of scan flip-flops {
    for n number of flip-flops {
        compute testabilities;
        compute testability-sum;
        compute gain;
    }
    find maximum gain flip-flop;
    change the flip-flop with maximum gain
    as a scan flip-flop and update the circuit;
    compute testabilities;
    compute testability-sum;
}
```

함으로써 부분스캔을 구현할 수 있다. 순차회로의 플립플롭을 그래프의 vertex로 나타내고 조합회로 요소의 연결을 arc로 나타냄으로써 순차회로를 그래프로 모델링하며 이를 S-그래프라고 한다. 구조분석에 의한 부분스캔이 테스트용이도에 의한 부분스캔보다 낮은 고장검출율을 낸다는 것은 테스트용이도에 의한 방법이 구조분석에 의한 방법보다, 고장검출율을 더 높일 수 있는 플립플롭을 선택한다고 볼 수 있다.

그래프에서 vertex의 부분집합에 속한 모든 vertex u와 v에 대해 u에서 v로 가는 방향성 통로가 존재할 때 그 vertex의 부분집합을 SCC(Strongly Connected Component)라 한다. 순차회로를 모델링한 그래프에서 SCC를 해석해보면 SCC에 포함된 각 플립플롭의 출력이 직·간접적으로 다른 모든 플립플롭의 입력으로 들어간다는 것을 알 수 있다. 즉, 모든 플립플롭들이 회귀사이클로 연결되어 있는 것이다. 순차회로에서 회귀사이클이 테스트 패턴 생성을 어렵게 만든다는 사실은 기존의 연구에서 이미 밝혀진 사실이다. 회귀사이클들로 연결된 가장 큰 플립플롭의 집합(최대 SCC)을 이익함수 계산의 대상으로 삼음으로써 이익함수 계산의 양을 줄인다.

테스트용이도에 의한 부분스캔은 구조분석에 의한 부분스캔 방법보다, 고장검출율을 더 높일 수 있는 효과적인 스캔플립플롭을 선택한다. 그러나 테스트용이도에 의한 부분스캔은 스캔플립플롭 선택 시간이 많이 걸린다. 테스트용이도에 의한 부분스캔의 장점인 높은 고장검출율을 이루며 짧은 시간에 스캔플립플롭을 선택하기 위한 새로운 부분스캔 설계 기술이 구조분석 및 테스트용이도 통합에 의한 부분스캔이다. 논문 [14]의 실험 결

과를 예로 들자면, 벤치마크회로 S5378의 경우 테스트 용이도에 의한 방법과 구조분석 및 테스트용이도 통합에 의한 방법의 고장검출율은 각각 0.959와 0.964로 비슷한데 스캔플립플롭 선택 시간은 각각 1325.80초와 524.53초로서 두 배 이상 더 소요되었고, 이 차이는 회로가 커짐에 따라서 현격히 증가함을 알 수 있었다.

구조분석 및 테스트용이도 통합에 의한 부분스캔은 그림 4와 같은 순서를 따른다. 먼저 순차회로를 그래프로 모델링한 후 SCC로 분할한다. 가장 많은 플립플롭을 포함하고 있는 SCC의 플립플롭들만을 대상으로 스캔이익을 계산하여 가장 큰 스캔이익을 얻는 플립플롭을 선택한다.

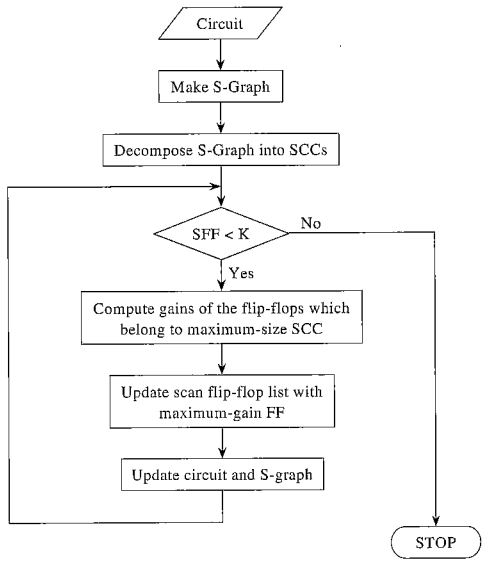


그림 4 구조분석 및 테스트 용이도 통합에 의한 부분스캔

그 다음 선택된 플립플롭으로 회로와 그래프를 갱신한다. 그림에서 SFF는 선택된 스캔플립플롭의 개수를 의미하고 K는 스캔할 플립플롭의 개수를 나타낸다.

4. 상위 · 하위 수준에서 통합된 테스트 설계 합성

본 논문에서 사용한 랜덤패턴 시뮬레이션 방법은 다음과 같다. 그림 5는 변수에 대한 0-조정도와 1-조정도를 산출하기 위하여 GCD회로에 랜덤패턴 생성기를 추가한 블록도이다. GCD 내부 제어문 주위에 출력변수를 추가하고 조정도를 측정하여 식 1,2와 같은 상대적 조정도와 랜덤도를 산출할 수 있게 했다.

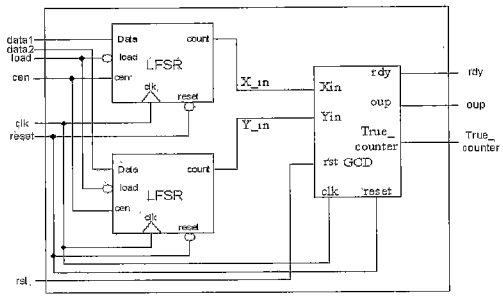


그림 5 랜덤패턴생성기가 추가된 GCD블럭도

상위수준인, VHDL 소스코드에 테스트점을 삽입하여 스캔 설계를 하는 전체 과정은 아래와 같다

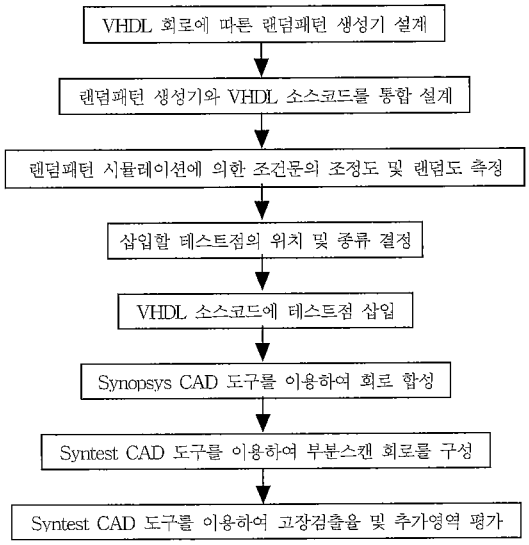


그림 6 테스트점 삽입과 스캔 설계과정

위와 같은 과정에 의해 테스트점을 삽입하여 합성된 회로에 대해서 부분스캔 회로를 추가하여 고장검출율을 구하게 된다. 이렇게 상위에서 테스트점을 찾아내고, 하위에서 스캔 설계를 함으로서 통합된 테스트 설계를 할 수 있게 된다.

5. 실험결과

실험순서는 그림 7과 같이 상위수준에서 기술된 회로를 분석한 다음 제어점을 삽입하고 상위수준의 합성도구를 사용하여 합성하였다. 합성된 게이트수준의 회로도에 대하여 순차회로 테스트패턴 생성기를 사용하여 고

장검출율의 증진여부를 검증하여 보았다. SUN Ultra 1 에서 Synopsys CAD 도구를 이용하여 VHDL 시뮬레이션, 합성 및 고장검출율을 산출하였다.

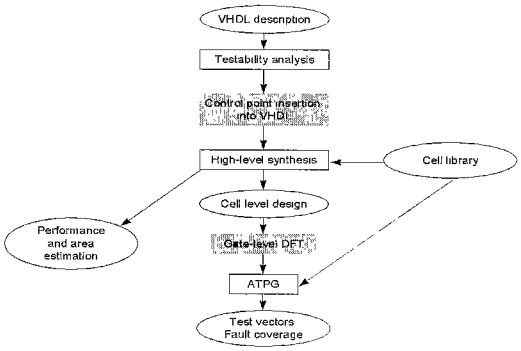


그림 7 테스트점 삽입에 대한 설계 순서도

표 2 회로 특징

회로	반복문	조건문
GCD	4	1
BARCODE	6	3
FANCY	2	5
DISPLAY	1	5

삽입하는 위치에 따라서 고장검출율의 차이가 많음을 알 수 있고 적합한 테스트점의 위치와 종류를 선택하지 않으면 표 3의 GCD회로에서 c2의 경우와 같이 오히려 고장검출율이 낮아질 수 있다. 하지만 일반적으로 테스트회로를 삽입하지 않은 회로에 비하여 고장검출율이 상당히 증진됨을 표 3에서 알 수 있다.

테스트점을 삽입하지 않은 GCD 본래의 회로는 69.40% 고장검출율인데 비하여 while 루프에 AND 형태의 테스점을 삽입할 경우는 최대 77.82%의 고장검출율이 증진되었다. 표 4는 Barcode회로에 대하여 본 논문에서 사용한 RRC, RRR에 의한 방법과 모든 분기점 제어를 주입력에 의하여 K 클럭 사이클 이내에 할 수 있도록 테스트점을 삽입하는 K-Controllable[12]에 의한 방법을 비교하였다. 미사용시에 고장검출율이 다른 이유는 합성도구 등의 환경이 다르기 때문이므로 엄밀한 비교는 하기 힘들었다. 본 논문이 조금 더 좋은 결과를 보였지만 아주 좋은 결과라고 볼 수는 없었다. 따라서 상위수준에서의 테스트점의 삽입과 하위 수준에서 부분스캔을 고려하는 상위·하위 통합에 의한 방법이 보다 적은 스캔플립플롭으로 표 3,7과 같이 추가 영역을 최소화 하며 보다 높은 고장검출율을 얻을 수 있다. 표 3의 회로 크기의 단위는 NOT 게이트의 크기로서 테스트점을 사용하지 않은 회로의 크기를 100%로 하였고, 합성 도구에서 회로의 크기를 최소화하는 것으로 최적화 시켰으며, +는 증가를 -는 감소를 의미한다.

표 4 본 논문과 기존의 방법에 의한 결과의 비교

테스트점	RRC 와 RRR에 의한 방법		K-Controllable에 의한 방법	
	고장검출율	회로크기	고장검출율	회로크기
c1	84.06%	+0.53%	73.87%	+0.88
미사용	65.68%	100%	59.11%	100%

이러한 테스트점의 종류 및 삽입할 위치를 체계적으로 설정하기 위해 랜덤패턴 시뮬레이션에 의하여 GCD 회로 변수들의 각 비트를 관측하였다. 변수의 비트가 '0'과 '1'이 될 확률을 산출하고 이상적인 0.5확률과의 차이를 합산한 결과는 표 5와 같다.

표 5는 루프의 방문 회수에 따른 상대적 레지스터 조정도(RRC)를 나타내며 C1 테스트점의 경우 미사용 시에 비해서 훨씬 더 조정도가 향상되었다. 표 3을 참조하면 C1에 의하여 고장검출율이 9% 정도 증진되었으며

표 3 while 문에 테스트점을 삽입한 회로의 고장검출율 과 회로크기의 비교

테스트점	GCD				BARCODE				FANCY	
	while(x>y)		while(y/=0)		while((white= 255) or (black=255))		while((actnum = num) and (white=255))		while(counter< b)	
	고장검출율	회로크기	고장검출율	회로크기	고장검출율	회로크기	고장검출율	회로크기	고장검출율	회로크기
c1	73.31	+1.65	77.82	+2.14	84.06	+1.84	70.85	+2.03	81.13	-1.57
c2	69.31	-2.8	73.01	+1.15	75.05	+2.32	72.17	+2.71	72.12	+1.01
c3	73.36	+2.45	74.83	+3.46	70.87	+1	68.88	+1.35	81.63	-0.56
미사용	69.40	100 %	N.A.		65.68	100 %	N.A.		48.15	100 %

표 5 GCD의 상대적 레지스터 조정도(RRC)

변수 \ 테스트점	미사용	while(y/=0) and C1	while(y/=0) or C2
x := x-y	1.2976	1.0929	1.2976
h := x	2.2240	1.7829	3.4003
x := y	1.4424	1.0734	2.4728
y := h	2.2240	1.7829	3.4003
Total	7.1880	5.7321	10.571

변수의 각 비트가 균등하게 나타남에 따라 고장검출율이 증진됨을 확인할 수 있었다

지금까지는 비트별로 얼마만큼 '0'과 '1'의 조정도가 균등하게 분포되는가를 국부적으로 관찰했다. 이제는 각각의 비트가 아닌 비트 전체인 변수에 대해 관찰해 본다. Fancy 회로 변수들에 대해서 상대적으로 원래의 회로보다 얼마나 다양한 값을 가지는 가를 관측하였다. While 루프에서 방문되는 횟수에 비해서 다양한 값을 가지게 되면 적은 시간에 많은 패턴을 발생시킬 수 있다. 따라서 이것은 적절한 제어점이 될 것이다. 식 2에 따른 Fancy 회로에서의 상대적 레지스터 랜덤도(RRR)는 표 6과 같다. While 구문에 테스트점 C1을 삽입한 경우 미사용한 경우에 비해 상대적 랜덤도는 월등히 증가한다. 즉 적은 시간 내에 다양한 패턴을 많이 생성시키므로 테스트패턴 생성이 용이해 질 것이다. 반면에 테스트점 C2를 삽입한 경우는 많은 시간을 할당하여 얻어지는 패턴이 C1에 비해 상대적으로 적으므로 적절한 제어점이 되지 못함을 알 수 있다.

적절한 제어점을 찾은 후에 Syntest CAD 도구를 이용하여 부분스캔 회로를 구현하여 고장검출율을 구하였다. 테스트점을 삽입하기 전에 회로와 테스트점을 삽입한 후의 회로에 대해 비교하였다. 여기서 스캔하는 플립플롭의 개수는 각 회로에 사용되는 플립플롭의 수에 10%, 20% 정도로 하여 부분스캔을 수행하였다. 그 결과는 표 7에 나타나 있다.

표 7 테스트점 삽입 전후의 부분스캔 고장검출율

	Total FF	SFF num	org+pscan F.C	T.P+pscan F.C
GCD	27	3	77.76	88.34
		5	83.59	89.60
FANCY	51	5	57.25	74.36
		10	78.33	81.01
BARCODE	46	5	63.13	66.80
		9	67.51	69.85
DISPLAY	42	4	76.82	80.28
		8	92.01	95.69

각 회로에 대해 표 7에서는 전체 플립플롭의 개수, 스캔플립플롭의 개수, 원래회로에 부분스캔을 수행한 후의 고장검출율, 테스트점을 삽입하고 부분스캔을 수행한

표 6 변수의 상대적 랜덤도

변 수 \ 테스트점	미사용	while and C1	while or C2
temp6a_if	9537 : 106/256=0.4141	4544 : 57/256=0.4674	11760:116/256 =0.3675
temp6a_else	16807 : 199/256=0.7773	10490 : 160/256=1.0014	16761:160 /256=0.6267
temp1b_if	25932 : 198/256=0.7734	14673 : 161/256=1.1115	28149:161 /256=0.5794
temp1b_else_if	95 : 95/256=0.3711	72 : 73/256=0.3763	72:73 /256=0.3763
temp1b_else_else	317 : 106/256=0.4141	289 : 89/256=0.3814	300:89 /256=0.3674
temp4_if	213 : 3/256=0.0117	202 : 3/256=0.0123	213:3 /256=0.0117
temp4_else	26131 : 233/256=0.9102	14832 : 194/256=1.3351	28308:228 /256=0.8221
temp3_if_if	2934 : 25/256=0.0977	1577 : 17/256=0.1235	2573:17 /256=0.0757
temp3_if_else	4203 : 71/256=0.2773	3015 : 58/256=0.3159	8224:58 /256=0.1158
temp3_else	19207 : 166/256=0.6484	10442 : 142/256=1.0203	17724:142 /256=0.6011
temp1a_if	26143 : 210/512=0.4102	14873 : 158/512=0.5424	28360:199/512=0.3590
temp1a_else_if	145 : 144/512=0.2812	117 : 118/512=0.2857	117:118 /512=0.2857
temp1a_else_else	56 : 57/512=0.1113	44 : 45/512=0.1119	44:45 /512=0.1119
counter_while	26344 : 255/256=0.9961	15034 : 238/256=1.6291	28521:255 /256=0.9201
Total	6.4941	8.7142	5.6204

후의 고장검출율 순으로 되어 있다.

표 7에서 나타난 것처럼 적은 스캔플립플롭으로 표 3과 같이 추가 영역을 최소화 하며 보다 높은 고장검출율을 얻을 수 있음을 알 수 있다.

6. 결론 및 향후 연구 계획

본 논문에서는 상위수준인, VHDL 제어문에 테스트점을 삽입하여 합성 후 게이트수준에서 스캔 설계함으로써 고장검출율을 증진시키는데 목표를 두고 있다. 랜덤패턴 시뮬레이션에 의하여 0-조정도, 1-조정도와 랜덤도를 산출하고 이익함수를 정의하여 체계적으로 신속정확하게 테스트점을 선정할 수 있는 기술을 제안한다. 조정도 산출을 위하여 GCD 회로에 랜덤패턴 생성기를 포함한 회로를 설계하여 시뮬레이션을 수행한다. 그리하여 변수들이 각 비트별로 '0'과 '1'이 균등하게 나타난 변수에 대하여 방문 회수에 따른 상대적 조정도와 랜덤도를 산출하고 AND, OR 및 XOR 형태의 테스트점을 삽입하였다. If-then-else에는 XOR 테스트점을, while 루프에는 AND 테스트점 주로 사용하여 상위합성 benchmark 회로에서 Synopsys CAD 도구로 합성 후 테스트패턴을 생성해 본 결과 고장검출율이 삽입 위치에 따라 큰 차이가 있음을 알 수 있었다. 본 논문에서 제안하는 방법에 의하여 VHDL 코드에 단일 테스트점을 삽입 합성하고 Syntest CAD 도구를 이용하여 부분스캔한 후에 고장검출율을 확인해 보았다. 그 결과 게이트수준 회로에 대한 고장검출율이 테스트점을 삽입하지 않은 회로에 비해서 최대 30%까지 증진됨을 알 수 있었다. 따라서 적절한 테스트점을 삽입한 후에 부분스캔을 하는 것이 적은 추가 영역으로 높은 고장검출율을 얻을 수 있는 효과적인 방법이라 할 수 있다.

향후 계획으로서 스캔설계의 단점인 스캔체인의 구성에 필요한 multiplexer(MUX)로 인한 추가영역의 증가 및 시스템 속도 저하를 줄이기 위하여 테스트점 삽입으로 스캔체인을 구성함으로써 각각의 플립플롭에 MUX를 추가하지 않고도 테스트모드시 자동으로 스캔할 수 있는 기술을 개발할 것이다.

참 고 문 헌

[1] Mujumdar A., Saluja K. and Jain R., "Incorporating Testability Considerations in High-Level Synthesis," Proceedings, ICCAD, pp.272-279, 1992.

[2] Dey S. and Potkonjak M., "Non-Scan Design-For-Testability of RT-Level Data Paths," Proceedings, Design Automation Conf., pp.640-645, 1994.

[3] Chickermane V., Lee J. and Patel J., "A Comparative Study of DFT Methods Using High-Level and Gate-Level Descriptions," Proceedings, ICCAD, pp.620-624, 1992.

[4] Dey S., Gangaram V. and Potkonjak M., "A Controller-Based Design-For-Testability Technique for Controller-Data Path Circuits," Proceedings, ICCAD, pp.534-540, Nov. 1995.

[5] Wagner K. D. and Dey S., "High-Level Synthesis for Testability: A Survey and Perspective," Proceedings, Design Automation Conf., pp.131-136, Jun. 1996.

[6] Chen C., Karnik T., and Saab D. G., "Structural and Behavioral Synthesis for Testability Techniques," IEEE Trans. on Computer-Aided Design, Vol. 13, No. 6, pp.777-785, Jun. 1994.

[7] Hsu F. F., Rudnick E. M., and Patel J. H., "Enhancing High-Level Control-Flow for Improved Testability," Proc. Int'l Conf. on Computer-Aided Design, pp.322-328, Nov. 1996.

[8] K. A. Ockunzzi and C. A. Papachristou., "Testability Enhancement for Behavioral Descriptions Containing Conditional Statements," Proceedings, Int'l Test Conf., pp.236-245, Nov. 1997.

[9] Chickermane V. and Patel J. H., "An Optimization Based Approach to the Partial Scan Design Problem," Proceedings, Int'l Test Conf., pp.377-386, Oct. 1990.

[10] Park S. J. and Akers S. B., "A Graph Theoretic Partial Scan Design By K-Cycle Elimination," Proceedings, Int'l Test Conf., pp.303-311, Oct. 1992.

[11] Abramovici M., Kulikowski J. J. and Roy R. K., "The Best Flip-Flops to Scan," Proceedings, Int'l Test Conf., pp.116-173, Oct. 1991.

[12] Frank F.Hsu, Elizabeth M.Rudnick and Janak H. Patel, "Testability Insertion in Behavioral Descriptions," International Symposium on System Synthesis, Oct. 1996

[13] Fulvio Corno, Matt대 Sonza Reorda and Giovanni Squillero, "High-Level Observability for Effective High-Level ATPG," IEEE Trans, pp.411-416, 2000.

[14] J. Park ,S Shin and S Park, "A Partial Scan Design by Unifying Structural Analysis and Testabilities," 2000 IEEE International Symposium on Circuits and Systems, Vol. 1, pp.88~91 May 28 - May 31, 2000.

신 상 훈
1991년 3월 ~ 1998년 2월 한양대학교 전자계산학과 학사. 전자계산학과 전공. 1998년 3월 ~ 2000년 2월 한양대학교 전자계산학과 석사 전자계산학 전공. 2000년 3월 ~ 현재 삼성전자 반도체 network 부서.

송 재 훈
1995년 ~ 2000년 한양대학교 전자 · 컴퓨터 공학부 학사. 2000년 ~ 현재 한양대학교 컴퓨터 공학과 석사.

박 성 주
1976년 3월 ~ 1983년 2월 한양대학교 전자공학과 학사. 1983년 2월 ~ 1986년 8월 금성사 소프트웨어 개발 연구원. 1986년 9월 ~ 1988년 9월 University of Massachusetts 전기 및 컴퓨터공학과 석사. 1988년 9월 ~ 1992년 5월 University of Massachusetts 전기 및 컴퓨터공학과 박사. 1992년 5월 ~ 1992년 8월 University of Massachusetts 연구교수. 1992년 9월 ~ 1995년 2월 IBM 연구소(N.Y. U.S.A) 연구 staff. 1995년 3월 ~ 2000년 2월 한양대학교 조교수. 2000년 3월 ~ 현재 한양대학교 부교수.