논문 2004-41SD-2-8

RTL 회로의 데이터패스를 위한 비주사 DFT 기법

(An Non-Scan DFT Scheme for RTL Circuit Datapath)

양 선 웅*, 박 재 흥*, 김 문 준*, 심 재 헌*, 장 훈**

(Sun-Woong Yang, Jae-Heung Park, Moon-Joon Kim, Jae-Hun Shim, and Hoon Chang)

요 약

본 논문에서는 레지스터 전송 수준의 데이터패스를 위한 효율적인 비주사 DFT 기법을 제안하였다. 데이터패스를 위해 제안된 비주사 DFT 기법은 레지스터 전송 수준(RTL : register transfer level) 회로에 대한 계층적 테스트 용이도(hierarchical testability) 분석을 통해 테스트 용이도를 향상시킴으로써 최소의 하드웨어 오버헤드를 가지고 데이터패스 버스 폭의 변화와 관계없이 항상 높은 고장 효율과 빠른 테스트 패턴 생성 시간을 보장한다. 실험 결과를 통하여 제안된 기법이 주사 기법보다 테스트 패턴 생성 시간, 테스트 패턴 적용 시간, 면적 오버헤드 면에서 우수함을 확인하였다.

Abstract

In this paper, An efficient non-scan DFT method for datapaths described in RTL is proposed. The proposed non-scan DFT method improves testability of datapaths based on hierarchical testability analysis regardless to width of the datapath. It always guarantees higher fault efficiency and faster test pattern generation time with little hardware overhead than previous methods. The experimental result shows the superiority of the proposed method of test pattern generation time, application time, and area overhead compared to the scan method.

Keywords : Non-Scan DFT, FCDFG, 계층적 테스트, Controllability, Obversability

I. 서 론

VLSI 칩의 집적도가 날로 증가하면서 칩의 전체 제조비용에서 테스팅이 차지하는 비율이 크게 증가하고 있다. 이러한 테스트 비용을 감소시키기 위하여 주사(scan) 방식, 테스트 접점 삽입(test point insertion)방식, 내장된 자체 테스트(BIST) 방식 등 다양한 DFT 기법이 적용되고 있다[1],[2]. 주사 기법은 순차회로 내에 있는 모든 기억 소자를 스캔이 가능하게 변경함으로써 회로의 제어도(controllability)와 관측도(observability)를 향상시킨 기법이다. 그러나 주사 기법을 적용할 경우, 짧은 시간에 높은 고장 검출율을 갖는 테스트 패턴을 생성하기에는 적합하지만, 회로내의 기억 소자를 주사가 가능한 소자로 바꾸는 과정에서 많은 면적 오버헤드가 발생하고 테스트 패턴을 칩에 인가하는데 많은 시간이 소요된다는 점, 칩의 정상 동작 속도로 테스트 패턴을 인가할 수 없다는 단점들이 있다. 또한 VLSI 칩의 크기가 증가함에 따라 칩을 이루는 게이트의 수가 급격히 증가하게 되었고, 이로 인해 테스트 패턴을 생성하기 위한 탐색 공간이 너무 커지게 되었다. 이러한 문제점으로 인해 최근에는 레지스터 전송 수준에서 비주사 DFT(Non-scan Design For Testability) 기법과 테스트 패턴 생성 방법에 관한 연구가 많이 진행되고 있다[3],[4],[5].

본 논문에서는 레지스터 전송 수준 회로의 데이터패스를 위한 효율적인 비주사 DFT 기법에 대하여 기술한다. 제안된 기법은 주어진 레지스터 전송 수준의 회로에 대하여 회로의 클럭이 진행되어감에 따라 동작하는 기능 모듈들과 그 기능 모듈들간의 연결선을 트리 형태로 표

* 학생회원 숭실대 컴퓨터학과
(Dept of Computing. Soongsil Univ.)
** 정회원, 숭실대학교 컴퓨터학부(주저자)
(School of Computing. SoongSil Univ.)
※ 연구는 숭실대학교 교내연구비 지원으로 이루어졌음
접수일자 : 2003년 3월 18일, 수정완료일 : 2004년 2월 11일

현하는 FCDFG(fixed control data flow graph)를 생성하고, 이를 이용하여 계층적 테스트 용이도 분석 방법으로 테스트 경로를 찾는다. 만일 테스트 경로가 존재하지 않으면 비주사 DFT 하드웨어가 추가된다. 이렇게 비주사 DFT 하드웨어를 삽입함으로써 주사 기법에 비해 적은 면적 오버헤드, 짧은 테스트 패턴 적용 시간, 칩의 정상 동작 속도로 테스트 패턴을 인가할 수 있다는 장점이 있다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 기존의 연구에 대하여 정리하고 Ⅲ장에서는 본 논문에서 제안하는 레지스터 전송 수준 회로의 데이터패스를 위한 비주사 DFT 기법에 대하여 설명한다. Ⅳ장과 Ⅴ장에서는 실험 결과와 결론을 기술한다.

Ⅱ. 기존의 연구

1. 레지스터 전송 수준 회로의 구성

000Verilog-HDL 또는 VHDL과 같은 HDL로 기술된 회로는 여러 단계의 합성 과정을 통해 최종적인 회로가 생성된다. 그림 1은 VLSI 합성 과정과 각 단계에서 적용되는 DFT 기법을 보여준다. HDL로 기술된 회로의 상위 수준 합성을 위하여 덧셈기, 곱셈기와 같은 기능 모듈들을 그래프의 노드로, 회로의 변수를 그래프의 간선으로 변환한다. 이렇게 변환된 노드와 간선을 이용하여 DFG(data flow graph)를 생성한다[6]. DFG는 회로의 기능 모듈들과 변수들간의 연결 상태를 그래프의 형태로 보여준다. 생성된 DFG를 이용하여 스케줄링(scheduling)과 할당(allocation) 과정을 통해 레지스터 전송 수준의 회로를 생성하게 된다. 그림 2의 (a), (b)는 HDL 회로와 DFG 보여주고 있고 (c), (d)는 상위 수준 합성을 통해 생성된 SDFG(scheduled data flow graph)와 모듈 할당의 결과를 보여준다[6].

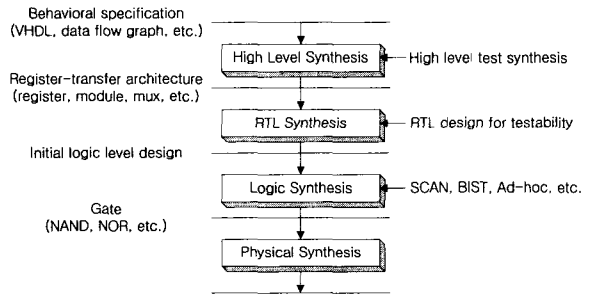


그림 1. VLSI 합성 과정과 DFT 기법
Fig. 1. VLSI synthesis procedure and corresponding DFT methodologies.

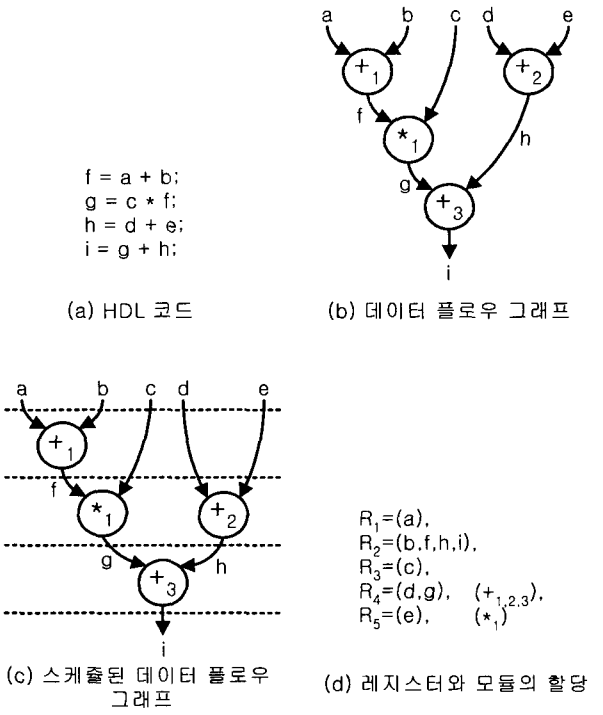


그림 2. HDL 코드, DFG, SDFG, 모듈 할당
Fig. 2. HDL code, DFG, SDFG, module assignment.

레지스터 전송 수준의 회로는 레지스터, 멀티플렉서, 덧셈기와 같은 기능 모듈, 그리고 이러한 레지스터 전송 수준의 회로를 구성하는 요소들을 연결하는 연결선으로 구성된다. 그림 3은 그림 2 (a)의 HDL 회로가 상위 수준의 합성을 통해 생성된 레지스터 전송 수준 데이터패스의 구조를 보여준다. 레지스터 전송 수준 회로가 생성된 후, 레지스터 전송 수준 회로에서 필요로 하는 적절한 제어 신호를 생성하기 위한 제어 회로가 합성된다. 최종적으로 생성된 레지스터 전송 수준의 회로는 그림 4와 같은 구조를 갖는다.

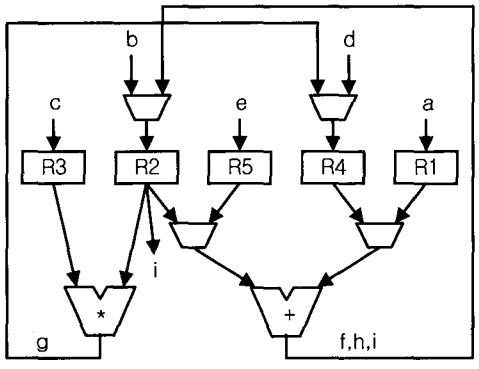


그림 3. 레지스터 전송 수준 데이터패스의 구조
Fig. 3. Structure of RTL Datapath.

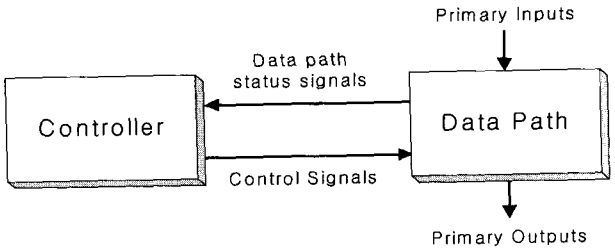


그림 4. 레지스터 전송 수준 회로의 구조
Fig. 4. Structure of RTL circuit.

2. 기존의 DFT 기법

레지스터 전송 수준의 회로를 테스트하기 위하여 주사 기법으로 H-SCAN 기법과 직교 주사(Orthogonal-SCAN) 기법이 제안되었다[5],[7]. H-SCAN 기법은 완전 주사 기법의 높은 면적 오버헤드와 테스트 패턴 적용 시간이 길다는 단점을 해결하기 위해서 제안된 기법이다[5]. 이 기법은 모든 레지스터, 주입력, 주출력을 노드로 갖는 연결 그래프를 구성한다. 그리고 연결 그래프의 분석을 통해 약간의 하드웨어를 추가함으로써 주사가 가능한 기억 소자를 이용하지 않고 회로내의 기능 모듈들을 통해 레지스터에 원하는 패턴을 병렬로 인가할 수 있도록 하였다. 직교 주사 기법은 기능 모듈과 테스트 회로를 공유함으로써 주사 기법의 면적 오버헤드와 테스트 패턴 인가 시간을 최소화하기 위해 제안된 기법이다[7]. 직교 주사 기법의 절차는 직교 주사 패스를 결정한 후, 직교 주사 패스가 구성될 수 있도록 기능 모듈을 변경한다. 이렇게 H-SCAN 기법과 직교 주사 기법은 일반적인 주사 기법의 면적 오버헤드가 크다는 단점과 테스트 패턴 적용 시간이 길다는 단점을 해결하기 위해 제안된 기법이다.

제어기 재설계 기법도 레지스터 전송 수준의 회로를 테스트하기 위하여 제안되었다. 각각 100%의 고장 검출율을 갖는 제어 회로와 데이터패스를 가지고 회로를 구성할 때 전체 고장 검출율이 100%가 나오지 않을 수도 있다. 이런 현상의 원인은 테스트 모드시에 100%의 고장 검출율을 갖는 테스트 패턴이 데이터패스에 인가될 때, 제어 회로로부터 생성되는 제어 신호에 의해 데이터패스만으로 구성된 회로에서처럼 경로가 형성되지 않을 수 있기 때문이다. 이러한 문제점을 해결하기 위하여 제어 회로의 제어 신호를 분석하여 제어 회로에 테스트 모드시 데이터패스에 테스트 패턴을 위한 경로가 생성되도록 제어 회로를 재설계하는 기법이 제안되었다[8]. 그리고 제어 회로의 제어 신호가 데이터패스의 고장 검출율에 많은 영향을 주고 있음을 실험을 통해 보여주고 데이터패스를 정상 모드, 정당화(justification) 모드, 전파(propagation) 모드로 구분하고, 각 모드에 맞춰서 제어 회로에서 적절한 제어 신호를 생성하도록 제어 회로를 재설계하는 기법도 제안되었다[9].

마지막으로 레지스터 전송 수준의 회로를 테스트하기 위하여 테스트 하드웨어 추가를 이용한 비주사 DFT 기법이 제안되었다. H. Fujiwara에 의해 제안된 기법으로 회로가 높은 테스트 용이도를 갖도록 하드웨어를 추가하는 기법이다[10]. 회로가 높은 테스트 용이도를 갖는다는 것은 대상이 되는 기능 모듈의 입력에 원하는 테스트 패턴을 인가할 수 있고 기능 모듈의 출력값을 회로의 주출력까지 출력할 수 있음을 의미한다. 테스트 용이도를 갖도록 생성된 제어 회로의 제어 신호의 모음을 test plan이라 한다. 회로가 높은 테스트 용이도를 갖도록 추가되는 하드웨어는 멀티플렉서, bypass 레지스터, hold 레지스터가 쓰인다. I. Ghosh에 의해 제안된 기법은 TCDF(test control data flow)를 이용하여 회로의 고장 검출율을 높이는 기법이다[11]. 제어 회로의 제어 신호 분석을 통해서 얻을 수 있는 TCDF는 매 클럭마다 수행되는 기능 모듈들과 레지스터 또는 래치의 값을 표현하는 자료구조이다. TCDF로부터 레지스터 전송 수준 회로의 기능 모듈들을 위한 테스트 경로가 만들어지도록 하드웨어를 추가하는 기법이다.

Ⅲ. 데이터패스를 위한 비주사 DFT 기법

높은 테스트 용이도를 갖는 게이트 수준의 회로를 얻기 위해 많은 테스트 용이화 기법들이 상위 수준의 합성 시스템에 도입되었다. 그리고 이러한 방법들의 도입으로 게이트 수준 회로의 테스트 용이도가 크게 증가되었다. 그러나 데이터패스의 데이터 버스의 폭이 넓어짐에 따라 게이트 수준 회로의 테스트 패턴 생성에 매우 오랜 시간이 걸리게 되었다[12]. 이로 인해 상위 수준에서의 정보를 이용하여 테스트 패턴 생성 시간을 줄이기 위한 계층적 테스트 생성 방법이 제안되었다. 계층적 테스트 생성은 레지스터 전송 수준의 회로를 구성하는 각각의 기능 모듈들에 대한 ATPG를 통해 각각의 기능 모듈들을 위한 테스트 집합(test set)을 구한다. 그리고 각각의 기능 모듈의 테스트 집합을 기능 모

듈에 인가하고, 그 결과를 칩 외부로 출력하기 위해 시스템 수준에서 심볼을 이용하여 테스트 경로를 찾는다.

본 논문에서는 일반적인 상위 수준 합성에 의해 생성된 레지스터 전송 수준 회로의 계층적 테스트 생성을 가능하게 하는 비주사 DFT 기법을 제안하였다. 제안된 기법은 주어진 레지스터 전송 수준의 회로의 클럭이 진행되어감에 따라 동작하는 기능 모듈들과 그 기능 모듈들간의 연결선을 트리 형태로 표현한 FCDFG를 생성한다. FCDFG의 정의는 다음과 같다.

[정의 1] FCDFG : G=(V,E)는 레지스터 전송 수준 회로의 기능 모듈인 노드 V, 기능 모듈간의 연결선인 간선 E로 구성된 트리이고, 트리의 각 레벨 경계는 클럭이다.

그리고 생성된 FCDFG를 이용한 계층적 테스트 용이도 분석 방법으로 테스트 경로를 찾는다. 만약 테스트 경로가 존재하지 않을 경우는, 레지스터 전송 수준 회로에 최소한의 비주사 DFT 하드웨어를 추가함으로써 테스트 경로를 생성한다.

1. FCDFG의 생성

FCDFG는 상위 수준 합성 후, 제어 회로의 제어 신호와 모듈 할당 정보로부터 만들어진다. 그림 5와 표 1은 레지스터 전송 수준 벤치마크 회로로 많이 쓰이는 Tseng 회로의 데이터패스 구조와 제어 신호를 보여준다. Tseng 회로는 3개의 주입력 포트(Inport1~3)와 2개의 주출력 포트(Outport1~2)가 있다. 내부 기능 모듈로는 3개의 덧셈기(ADD1~3)와 각각 뺄셈기(SUB1), 곱셈기(MUL1), OR 연산기(OR1), AND 연산기(AND1)가 1개씩 있다. 그리고 다수의 멀티플렉서와 레지스터로 구성되어 있다.

표 1. Tseng 회로의 제어 신호
Table 1. Control signals for Tseng circuit.

Input	State		OUTPUTS											
reset	PS	NS	w1	w2	w3	w4	w5	w6	m1	m2	m3	m4	m5	m6	m7
1	Any	s0	1	0	1	1	1	1	0	1	1	1	0	0	0
0	s0	s1	0	1	0	0	0	0	0	0	0	0	0	0	0
0	s1	s2	1	0	0	1	0	0	0	0	0	0	1	0	0
0	s2	s3	1	1	0	1	0	0	0	0	1	0	0	1	1
0	s3	s4	1	0	1	0	0	0	1	0	0	0	0	0	0
0	s4	s0	0	0	0	1	1	1	0	0	0	0	0	0	0

그림 5 Tseng 회로의 FCDFG는 그림 6과 같다. 그림에서 +1, +2, +3 등의 노드들은 회로의 기능 모듈 ADD1, ADD2, ADD3 등을 나타낸다. 그리고 음영이 들어간 노드는 모듈의 연산 결과가 회로의 주출력으로 출력되는 모듈을 의미한다. 리셋 상태에서 Tseng 회로의 동작을 살펴보면 다음과 같다. 레지스터 제어 신호 w1이 활성화되고, 멀티플렉서 제어 신호 m2와 m3가 1이기 때문에 one(상수 1) 값이 reg1에 적재된다. reg3는 w3 신호에 의해 데이터를 적재할 수 있는 모드가 되고,

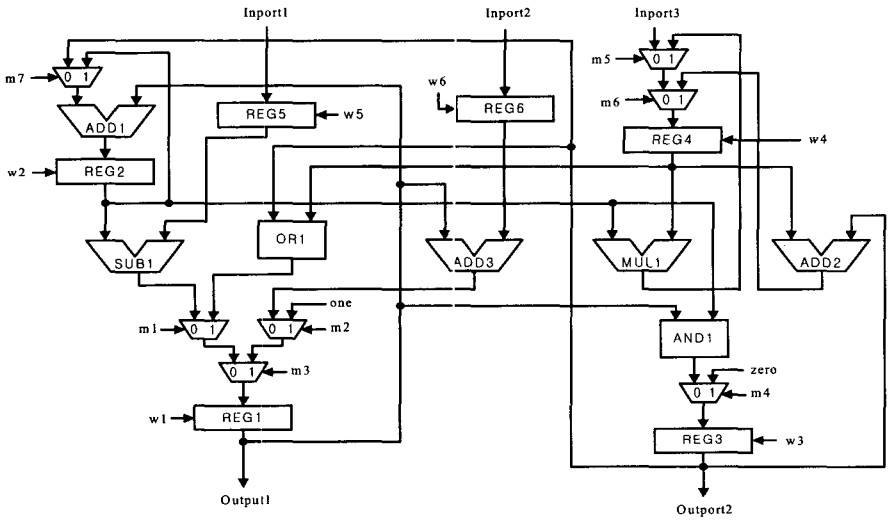


그림 5. 레지스터 전송 수준의 Tseng 회로
Fig. 5. Tseng circuit in RTL.

멀티플렉서 제어신호 m4에 의해 zero(상수 0) 값을 적재하게 된다. reg4는 w4 신호가 활성화되고, 멀티플렉서 제어 신호 m5와 m6이 0이기 때문에 주입력 값인 Inport3를 적재하게 된다. reg5와 reg6도 같은 방식으로 값이 적재되게 된다. FCDFG의 Inport3, zero, one, Inport1, Inport2는 리셋 상태에서 위에서 살펴본 방식에 의해 각 레지스터에 적재된 값을 의미한다.

2. FCDFG를 이용한 계층적 테스트 용이도 분석

계층적 테스트 용이도 분석은 심볼을 이용하여 해당 모듈이 완전 테스트 가능 모듈인지 여부를 결정한다.

[정의 2] 완전제어 경로(CCP : complete control path) : FCDFG의 간선 e에 원하는 값을 인가할 수 있을 때, 간선 e는 완전 제어 경로라 한다.

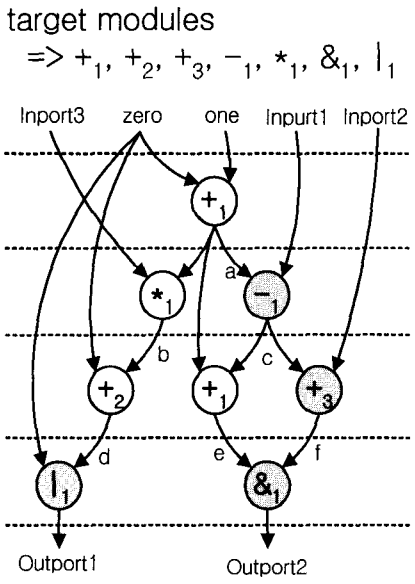


그림 6. Tseng 회로의 FCDFG
Fig. 6. FCDFG for Tseng circuit.

[정의 3] 완전 관측 경로(COP : complete observe path) : FCDFG의 간선 e의 값을 회로의 주출력으로 출력할 수 있을 때, 간선 e는 완전 관측 경로라 한다.

[정의 4] 완전 테스트 가능 모듈(CTM : complete testable module) : 회로의 주입력으로부터 모듈의 모든 입력까지 완전 제어 경로가 존재하고, 모듈의 출력으로부터 회로의 주출력까지 완전 관측 경로가 존재할 때 완전 테스트 경로가 존재한다고 하고, 완전 테스트 경로가 존재하는 모듈을 완전 테스트 가능 모듈이라 한다.

예를 들어 그림 7에서 ADD3(+3)이 완전 테스트 가능 모듈이 되기 위해서는 입력 Inport2와 SUB1(-1)의 출력이 완전 제어 경로이어야 하고, ADD3의 출력이 완전 관측 경로이어야 한다. ADD3의 입력 Inport2는 회로의 주입력이기 때문에 완전 제어 경로이다. 그리고 SUB1의 출력은 입력 Inport1이 완전 제어 경로인 회로의 주입력이기 때문에 완전 제어 경로가 된다. ADD3의 출력은 회로의 주출력이기 때문에 완전 관측 경로가 존재한다. 따라서 ADD3는 완전 테스트 가능 모듈이다.

그림 8에서 SUB1의 경우는 입력은 Inport1과 ADD1의 출력이다. Inport1은 주입력이므로 완전 제어 경로이다. 그러나 ADD1은 두 입력 모두 완전 제어 경로가 아닌 zero(상수 0)와 one(상수 1)이기 때문에 ADD1 출력은 완전 제어 경로가 되지 못한다. SUB1의 출력이 회로의 주출력이기 때문에 완전 관측 경로가 존재한다. 따라서 SUB1은 ADD1의 출력을 받는 입력이 완전 제어 경로가 아니기 때문에 완전 테스트 가능 모듈이 아니다. SUB1 모듈을 완전 테스트 가능 모듈이 되게 만들기 위해서는 ADD1의 출력을 완전 제어 경로로 만들어야 한다. 이는 테스트 하드웨어를 추가하여 ADD1의 두 입력 중에 하나를 완전 제어 경로로 만드는 것을 통해 가능하다.

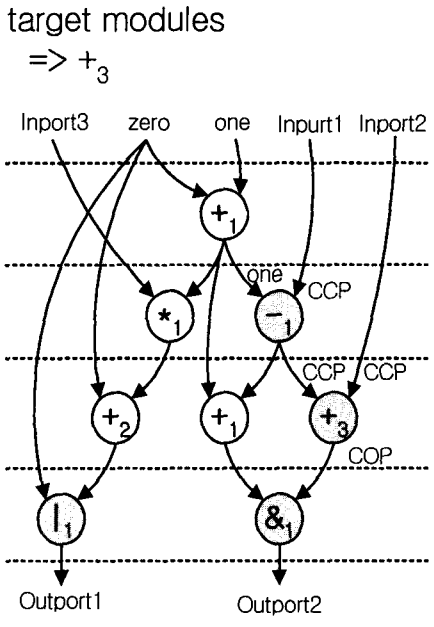


그림 7. 완전 테스트 가능 모듈의 예
Fig. 7. Example of a complete testable module.

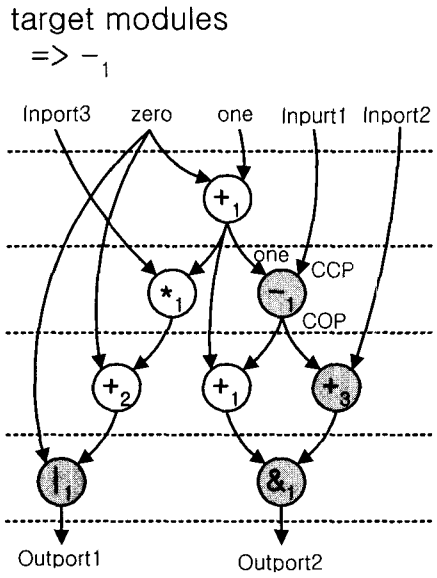


그림 8. 완전 테스트 가능이 아닌 모듈의 예
Fig. 8. Example of a non-complete testable module.

3. 데이터패스를 위한 비주사 DFT 하드웨어 추가

그림 8과 같이 레지스터 수준 회로의 모든 기능 모듈이 완전 테스트 경로가 갖고 있는 것은 아니다. 레지스터 전송 수준 회로가 계층적 테스트 패턴이 생성이 가능하기 위해서는 완전 테스트 경로가 존재해야 하는데, 이것은 적절한 테스트 포인트에 테스트 하드웨어를 추가함으로써 완전 테스트 경로를 생성할 수 있다.

[정의 5] 테스트 포인트(TP : test point) : 해당 모듈의 완전 테스트 경로를 생성하기 위해 테스트 하드웨어의 추가를 필요로 하는 지점을 테스트 포인트라고 한다.

그림 9는 완전 테스트 경로를 생성하기 위해 테스트 하드웨어를 추가하는 경우를 보여준다. (a)의 경우는 모듈의 입력이 모듈을 통과하지 않고 바로 모듈의 출력으로 보내는 경우이다. 예를 들어 모듈이 나눗셈기일 때, 입력 x가 완전 제어 경로일지라도 입력 y의 심볼 S를 모듈의 출력 z로 보낼 수 없는 경우가 있다. 즉 x에 인가되는 심볼이 X라 할 때, X/S = S가 되는 경우가 존재하지 않는 경우가 있다. (b)의 경우는 모듈의 입력을 완전 제어 경로로 만들기 위해 추가되는 로직이다. 예를 들면 회로의 정상적인 데이터 흐름을 이용해 모듈의 한쪽 입력에 원하는 심볼 S를 만들 수 없을 때에 사용하는 회로이다.

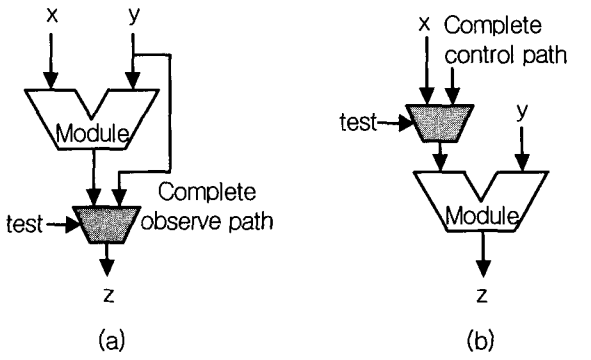


그림 9. 완전 테스트 경로를 위해 추가된 회로
Fig. 9. Added circuit for the complete testable path.

4. 데이터패스를 위한 테스트 포인트 탐색 알고리즘

데이터패스를 위한 테스트 포인트 탐색 알고리즘은 FCDFG로부터 레지스터 전송 수준 회로의 기능 모듈에 대한 CO_Module_List와 Module_List를 만든다. CO_Module_List는 모듈의 출력이 회로의 주출력과 직접 연결된 모듈에 대한 리스트이고, Module_List는 전체 모듈 리스트이다. 예를 들어 그림 5의 경우에, CO_Module_List는 {-1, +3, |1, &1}이고, Module_List는 {+1, *1, -1, +2, +3, |1, &1}이 된다. 제안된 알고리즘은 CO_Module_List에 대한 테스트 경로를 구한 후, Module_List에 대한 테스트 경로를 구한다.

Step1 : CO_Module_List의 모듈을 위한 테스트 경로 구하기

CO_Module_List에 있는 모듈들은 모듈의 입력에 대한 완전 제어 경로만 존재하면 완전 테스트 가능 모듈이 되기 때문에 CO_Module_List에 있는 모듈들에 대한 완전 테스트 경로를 먼저 구한다. 그리고 FCDFG에서 주입력으로부터 가장 가까운 모듈이 완전 제어 가능 경로를 만들기가 용이하기 때문에 CO_Module_List에 있는 모듈들 중에서 먼저 선택된다. 그림 5의 경우에는 SUB1(-1), ADD3(+3), OR1(|1), AND1(&1) 순으로 선택된다.

선택된 모듈에 대한 완전 테스트 경로가 존재하지 않을 경우에는 FCDFG의 다른 클럭에 대상 모듈이 있는지를 체크한다. 만약 대상 모듈이 존재하지 않을 경우, 현재의 클럭에서 테스트가 수행되어야 하기 때문에 비

주사 DFT 하드웨어를 추가하여 모듈을 위한 완전 테스트 경로를 생성한다. 그리고 만약 다른 클럭에 완전 테스트 경로가 있는 해당 모듈이 존재하면, 해당 모듈을 CO_Module_List와 Module_List에서 지운다. 존재하지 않을 경우에는 모듈의 출력이 회로의 주출력에 직접 연결된 모듈에 비주사 DFT 하드웨어를 추가하여 완전 테스트 경로를 생성하고, CO_Module_List와 Module_List에서 해당 모듈을 지운다. 그림 10은 CO_Module_List에 있는 모듈을 위한 테스트 경로 탐색 알고리즘을 보여준다.

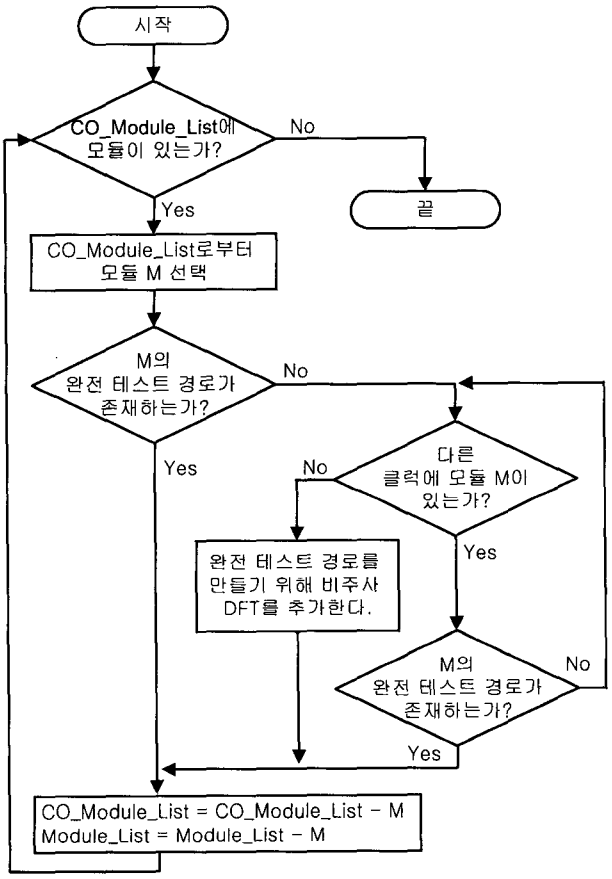


그림 10. CO_Module_List 모듈의 테스트 경로 탐색 알고리즘

Fig. 10. Test path search algorithm for CO_Module_List module.

Step2 : Module_List의 모듈을 위한 테스트 경로 구하기

Step1이 끝나면 Module_List에는 CO_Module_List에 있는 모듈을 제외한 모듈이 남아있게 된다. 즉 Module_List에 남아있는 모듈들이 완전 테스트 가능 모듈들이 되기 위해서는 완전 제어 경로와 완전 관측 경로를 가지고 있어야 한다. Module_List에 남아 있는 모듈들은 완전 제어 경로와 완전 관측 경로를 만들기 쉬운 모듈부터 테스트 경로를 생성하기 위해서 출력이 CO_Module_List에 있는 모듈까지의 거리가 짧고, 모듈의 입력부터 주입력까지의 거리가 짧은 모듈부터 선택된다. 그림 6의 경우에는 step1이 끝나면 Module_List에 {+1}만 남게 된다. ADD1(+1)은 zero와 one를 입력으로 받는 ADD1이 먼저 선택되고, 이 단계에서의 테스트 경로를 찾는다. 만약 테스트 경로가 존재하지 않는 경우는 다른 클럭 단계에 동일 모듈이 있는지를 찾게 된다. 만약 동일 모듈이 존재하지 않을 경우는 비주사 DFT 하드웨어를 추가하여 테스트 경로를 생성한다.

그리고 다른 클럭에 완전 테스트 경로가 존재하는 동일 모듈이 존재하면 Module_List에서 해당 모듈을 지운다. 그리고 다른 클럭에도 완전 테스트 경로가 존재하는 동일 모듈이 없는 경우에는 가장 처음에 선택했던 클럭 단계에서 비주사 DFT 하드웨어를 추가하여 테스트 경로를 생성한다. 그리고 Module_List에서 해당 모듈을 지운다.

그림 11은 step1과 step2 과정을 통해 레지스터 전송 회로(그림 5)의 모든 모듈이 완전 테스트 가능 모듈이 되도록 비주사 DFT 하드웨어가 추가된 회로의 구조를 보여준다.

Ⅳ. 실험결과

제안된 기법의 효율성을 검증하기 위해 레지스터 전송 수준의 연구에서 많이 사용되는 벤치 마크 회로인 Tseng, Paulin, Diffeq 그리고 DCT를 사용하였다. 벤치 마크 회로는 Verilog-HDL을 이용하여 기술하였고, 합성은 Synopsys를 사용하였다. 주사 사슬 삽입, 테스트 패턴 생성, 고장 시뮬레이션은 Syntest를 이용하였다.

표 2는 실험에 사용된 벤치마크 회로의 특성을 보여주는 표이다. 입력, 출력 열은 회로의 주입력과 주출력의 수이고, mux 열은 레지스터 전송 수준 회로에 포함된 2x1 멀티플렉서의 수이다. 기억소자 열은 회로에 포함된 기억 소자의 수이고, 기능 모듈 열은 회로에 포함된 기능 모듈의 리스트이다. 같은 기능을 하는 여러 개의 기능 모듈이 존재하는 경우는 첨자로 구분하였다.

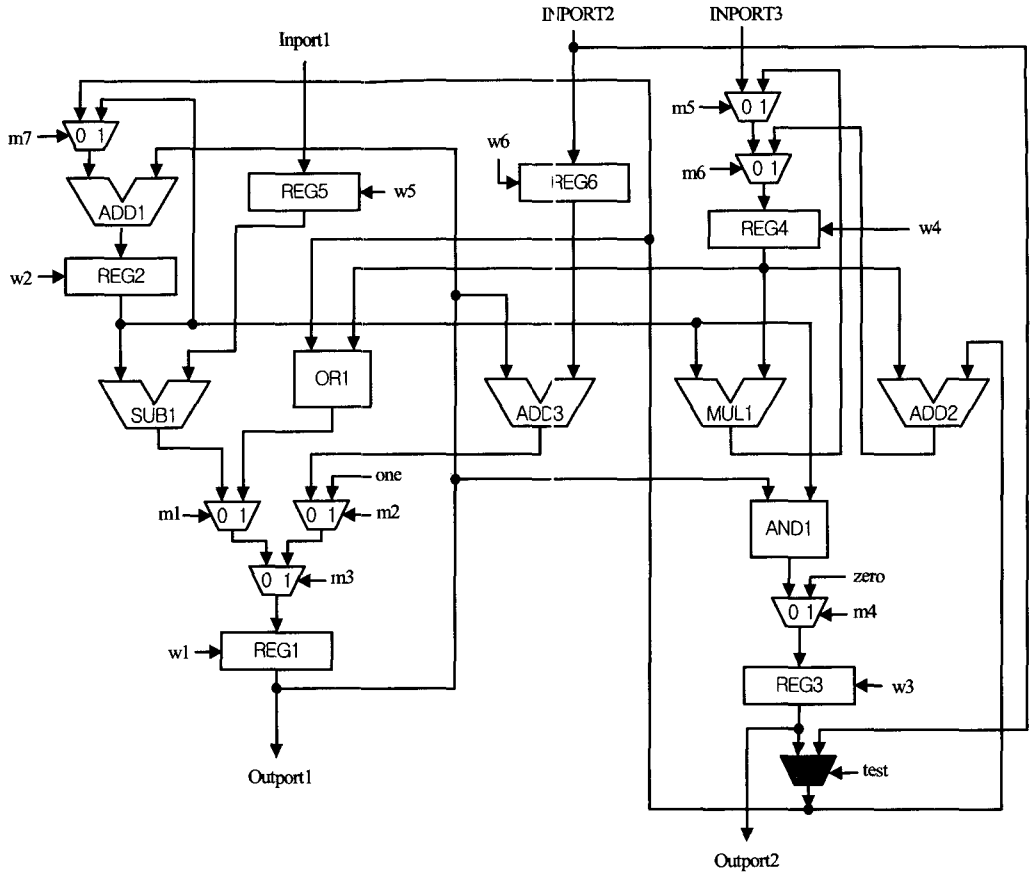


그림 11. 비주사 DFT 하드웨어가 추가된 Tseng 회로
Fig. 11. Tseng circuit added non-scan DFT hardware.

표 2. 벤치마크 회로의 데이터패스 특성
Table 2. Datapath characteristics of benchmark circuit.

Circuit	입력	출력	mux	기억소자	기능 모듈
Tseng	3	2	7	6	$+_1$, $+_2$, $+_3$, $-_1$, $*_1$, $\&_1$, $\|_1$
Paulin	2	2	11	7	$+_1$, $-_1$, $*_1$, $*_2$,
Diffeq	6	3	13	7	$+_1$, $+_2$, $-_1$,$-_2$, $*_1$, $*_2$, $>_1$
DCT	4	3	18	5	$*_1$, $*_2$, $*_3$, $+_1$, $+_2$, $+_3$, $-_1$, $-_2$

DFT 기법의 성능 평가는 테스트 패턴 생성 시간, 테스트 패턴 적용 시간, 면적 오버헤드와 같은 테스트 비용과 고장 효율과 같은 테스트 효율에 의해 평가된다. 따라서 본 논문에서 제안한 비주사 DFT 기법의 성능 분석을 위해서 DFT 기법이 전혀 적용되지 않은 original 회로, 레지스터에 테스트 패턴을 입력하기 위하여 레지스터를 하나의 주사 사슬(scan chain)로 연결하는 주사 기법이 적용된 회로, 그리고 제안한 비주사 DFT 기법이 적용된 회로에 대한 테스트 비용과 테스트 효율을 비교하였다.

표 3. 테스트 패턴 생성 시간
Table 3. Test pattern generation time(sec).

Circuit		Width			
		4 Bits	8 Bits	16 Bits	32 Bits
DCT	Original	154	908	9413	170439
	주사기법	0	0	0	2
	Ours	**0.06**	**0.11**	**0.23**	**0.55**
Tseng	Original	44	176	1251	12117
	주사기법	0	1	4	21
	Ours	**0**	**0**	**0.13**	**0.3**
Diffeq	Original	101	213	451	7303
	주사기법	0	1	5	1
	Ours	**0.06**	**0.11**	**0.24**	**0.63**
Paulin	Original	40	142	815	6520
	주사기법	0	1	2	9
	Ours	**0.04**	**0.06**	**0.1**	**0.41**

표 3은 데이터 버스의 폭의 변화에 따른 테스트 패턴 생성 시간의 변화를 보여준다. DFT 기법이 적용되지 않은 회로의 경우 버스 폭이 증가함에 따라 테스트 패턴을 생성하는 시간이 급격히 증가함을 알 수 있다. 주사 기법은 Syntest의 테스트 패턴 생성 툴에 의해 구해진 시간으로 소수점 이하의 초를 보여주지 않고 있으며, 제안한 비주사 DFT 기법은 소수점 2자리 이하에서 반올림한 결과를 보여주고 있다.

표 4는 테스트 패턴을 회로에 적용하는데 걸리는 시간을 보여주고 있다. 주사 기법의 테스트 패턴 적용 시간은 "테스트 패턴 * (기억소자의 수 + 1) + 기억소자의 수"에 의해 구했다. 주사 기법의 경우 데이터패스 버스의 폭이 커짐에 따라 기억 소자의 수(레지스터)가 증가하기 때문에 적용 시간이 증가하지만, 제안한 비주사 DFT 기법은 주사 기법에 비해 적은 영향을 받음을 알 수 있다.

표 4. 테스트 패턴 적용 시간
Table 4. Test pattern application time.

Circuit		Width			
		4 Bits	8 Bits	16 Bits	32 Bits
DCT	Original	650	793	1211	2225
	주사기법	719	1011	2015	4919
	Ours	**348**	**445**	**590**	**826**
Tseng	Original	436	882	821	1353
	주사기법	811	1611	3199	7447
	Ours	**327**	**385**	**511**	**668**
Diffeq	Original	896	776	773	1024
	주사기법	1151	2719	8447	27819
	Ours	**254**	**328**	**442**	**673**
Paulin	Original	303	734	1688	1802
	주사기법	863	1259	3363	7751
	Ours	**301**	**406**	**483**	**664**

표 5는 데이터패스에 대한 주사 기법과 제안한 비주사 DFT 기법의 면적 오버헤드를 보여주고 있다. 표의 괄호안의 값은 면적 오버헤드를 퍼센트로 표시하고 있다. 주사 기법의 경우 주사 체인의 길이는 "버스의 폭 * 레지스터의 개수"이다. 따라서 버스의 폭이 커질 경우, 주사 사슬을 형성하는 기억소자의 수가 증가하기 때문에 면적 오버헤드가 증가한다. 제안한 기법의 경우, 데이터패스에 적용되는 오버헤드는 "버스의 폭 * 비주사 DFT 하드웨어"이다. 그러나 비주사 DFT 하드웨어의 수는 회로에 포함된 레지스터 보다 훨씬 적기 때문에 면적 오버헤드가 거의 증가하지 않는다. 표 5의 실험 결과에서 4비트 회로들의 오버헤드가 큰 이유는 제어 회로의 제어 신호가 데이터패스의 출력보다 크기 때문에 제어 회로의 출력을 여러 클럭에 나누어서 회로 외부로 출력하기 위해서 레지스터를 사용하였기 때문이다.

표 5. 면적 오버헤드
Table 5. Area overhead.

Circuit		Width			
		4 Bits	8 Bits	16 Bits	32 Bits
DCT	Original	643	1556	4602.5	15310
	주사기법	771.5 (19.98)	1799.5 (15.65)	5076 (10.29)	16243.5 (6.10)
	Ours	**834.5 (28.27)**	**1710.5 (10.69)**	**4780.5 (4.76)**	**15565.5 (2.24)**
Tseng	Original	475.5	1002	2478	6955
	주사기법	627 (31.86)	1291.5 (28.89)	3043.5 (22.82)	8072.5 (16.07)
	Ours	**587.5 (23.55)**	**1097.5 (9.53)**	**2590 (4.52)**	**7107.5 (2.19)**
Diffeq	Original	578.5	1326.5	3593	11236.5
	주사기법	758.8 (31.17)	1667.8 (25.73)	4256.3 (18.46%)	12543.8 (11.63)
	Ours	**677.5 (17.11)**	**1445 (8.93)**	**3738.5 (4.05)**	**11416.5 (1.60)**
Paulin	Original	515.5	1224	3378	10801.5
	주사기법	672 (30.36)	1523.5 (24.47)	3963.5 (17.33)	11959 (10.72)
	Ours	**674 (30.75)**	**1350 (10.29)**	**3579.5 (5.97)**	**11094 (2.71)**

표 6은 DFT 기법이 적용되지 않은 회로, 주사 기법, 제안된 DFT 기법이 적용된 회로의 고장 효율을 보여준다. 제안된 기법은 주사 기법과 비슷한 수준의 고장 효율을 보장함을 알 수 있다.

표 6. 고장 효율
Table 6. Fault efficiency(%).

Circuit		Width			
		4 Bits	8 Bits	16 Bits	32 Bits
DCT	Original	70.7	69.7	67.5	59.7
	주사기법	99.2	98.9	99.0	99,2
	Ours	**99.1**	**99.2**	**99.6**	**99.8**
Tseng	Original	93.7	96.9	98.1	99.4
	주사기법	100	100	100	100
	Ours	**99.4**	**99.6**	**99.4**	**99.7**
Diffeq	Original	99.1	97.8	97.9	98.8
	주사기법	100	100	100	100
	Ours	**98.8**	**99.2**	**99.6**	**99.8**
Paulin	Original	98.1	99.2	99.7	99.9
	주사기법	100	100	100	100
	Ours	**98.8**	**99.1**	**99.6**	**99.8**

V. 결 론

VLSI 칩의 복잡도가 증가하면서 테스트 비용이 급격히 증가하게 되었다. 따라서 테스트 비용을 줄이기 위하여 다양한 DFT 기법이 연구되었다. 또한, VLSI 칩의 크기가 증가함에 따라 게이트의 수가 급격히 증가하게 되면서 테스트 패턴을 생성하기 위한 탐색 공간이 너무 커지게 되었다. 이러한 문제점으로 인해 최근에는 레지스터 전송 수준에서 비주사 DFT 기법과 테스트 패턴 생성 방법에 관한 연구가 많이 진행되고 있다.

본 논문에서는 레지스터 전송 수준의 데이터패스를 위한 효율적인 비주사 DFT 기법을 제안하였다. 제안한 기법은 상위 수준의 합성으로부터 생성되는 정보를 이용하여 클럭 단위로 동작되는 기능 모듈을 표현하는 FCDFG를 생성하였다. 그리고 계층적 테스트 용이도 분석을 통해 약간의 비주사 DFT 하드웨어를 추가한다. 이렇게 비주사 DFT 하드웨어를 추가함으로써 적은 면적 오버헤드와 짧은 시간에 테스트 패턴 적용이 가능하다. 그리고 데이터 폭의 변화에 관계없이 항상 높은 고장 검출율을 보장한다.

실험 결과 본 논문에서 제안된 기법이 주사 기법보다 데이터 폭의 변화에 관계없이 테스트 패턴 생성 시간, 테스트 패턴 적용 시간, 면적 오버헤드 면에서 우수함을 확인하였고 높은 고장 효율을 얻을 수 있었다.

참 고 문 헌

[1] R. S. Fetherson, I. P. Shak and S. C. Ma, "Testability Features of AMD-K6TM Microprocessor," IEEE Design & Test of Computers, pp. 64-69, 1998.

[2] D. Bhavsar, D. Akeson, M. Gowan and D. Jackson, "Testability Access of the High Speed Test Features in the Alpha 21264 Microprocessor," International Test Conference, pp. 487-495, 1998.

[3] I. Ghosh, N. K. Jha, S. Bhawmik, "A BIST Scheme for RTL Circuits Based on Symbolic Testability Analysis," IEEE Trans. on CAD, vol. 19, no.1, pp. 111-128, Jan. 2000.

[4] S. Bhattacharya, F.Brglez and S. Dey, "Transformations and Resynthesis for Testability of RTL Control-Data Path Specifications," IEEE Trans. VLSI Syst., vol. 1, pp, 304-318, Sept. 1993.

[5] S. Bhattacharya, S. Dey, "H_SCAN : A High Level Alternative to Full-Scan Testing with Reduced Area and Test Application Overheads," in Proc. VLSI Test Symp., pp. 74-80, 1996.

[6] Mike Tien and Chien Lee, High-Level Test Syntheis of Dighital VLSI Circuits, Artech House, 1997.

[7] B. Norwood and J. McCluskey, "Orthogonal SCAN : Low Overhead SCAN for Data Paths," International Test Conference, pp. 659-668, 1996.

[8] S. Dey, V. Gangaram and M. Potkonjak, "A Controller-Based Design-for-Testability Technique for Controller-Data Path Circuits," International Conference on Computer-Aided Design, pp.

534-540, 1995.

[9] V. Fernandez and P. Sanchez, "High-level test synthesis based on controller redefinition," IEE Electronics Letters, pp.1596 -1597, 1997.

[10] S. Ohtake, S. Nagia, H. Wada and H. Fujiwara, "A DFT Method for RTL Circuits to Achieve Complete Fault Efficiency Based on Fixed-control Testability," Asia and South Pacific Design Automation Conference, pp. 331-334, 2001.

[11] I. Ghosh, "A Design-for-Testability Technique for Register-Transfer Level Circuits Using Control/Data Flow Extraction," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp. 706-723, 1998.

[12] I. Ghosh, A. Raghunathan and N. Jha, "Design for Hierarchical Testability of RTL Circuits Obtained by Behavioral Synthesis," International Conference on Computer Design : VLSI in Computers and Processors, pp. 173-179, 1995.

저 자 소 개

양 선 웅(학생회원)

1996년 숭실대학교 전자계산학과 졸업(학사)

1998년 숭실대학교 대학원 전자계산학과 졸업(석사)

2002년 숭실대학교 대학원 컴퓨터학과 졸업(박사)

<주관심분야 : VLSI 설계 및 테스트, 컴퓨터구조, VLSI CAD>

박 재 흥(학생회원)

1999년 숭실대학교 컴퓨터학부 졸업(학사)

2002년 숭실대학교 대학원 컴퓨터학과 졸업(석사)

2002년~현재 숭실대학교 대학원 컴퓨터학과 박사과정

<주관심분야 : VLSI 설계 및 테스트, 컴퓨터구조, VLSI CAD>

김 문 준(학생회원)

2000년 숭실대학교 컴퓨터학부 졸업(학사)

2002년 숭실대학교 대학원 컴퓨터학과 졸업(석사)

2002년~현재 숭실대학교 대학원 컴퓨터학과 박사과정

<주관심분야 : VLSI 설계 및 테스트, 컴퓨터구조, VLSI CAD>

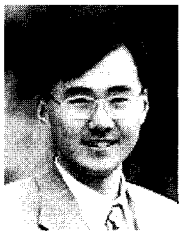
심 재 헌(학생회원)

1987년 서울대학교 물리교육과 졸업(학사)

1989년 서울대학교 물리교육과 졸업(석사)

2000년~현재 숭실대학교 대학원 컴퓨터학과 박사과정

<주관심분야 : VLSI 설계 및 테스트, 컴퓨터구조, VLSI CAD>



장 훈(정회원)

1987년 서울대학교 전자공학과 졸업(학사)

1989년 서울대학교 전자공학과 졸업(석사)

1993년 University of Texas at Austin 졸업(박사)

1991년 IBM Inc. Senior Member of Technical Staff

1993년 Motorola Inc. Senior Member of Technical Staff

1994년~현재 숭실대학교 컴퓨터학부 부교수

<주관심분야 : VLSI 설계 및 테스트, 컴퓨터구조, VLSI CAD>