SOC 테스트를 위한 효율적인 코어 테스트 Wrapper 설계 기법

(An Efficient Design Strategy of Core Test Wrapper For SOC Testing)

김 문 준 † 장 훈 ††
(Moon-Joon Kim) (Hoon Chang)

요 약 IC 집적기술이 고도로 발달하면서 출현한 SOC(System On a Chip)는 미리 설계된 코어를 재사용하는 모듈러 기법을 회로 설계 과정에 도입시켰고, 따라서 테스트 설계에도 모듈러 기법이 도입되었다. 이러한 SOC 테스트에 소요되는 비용의 최소화를 위해서는, SOC 테스트 구조의 핵심 구성요소인 코어 테스트 wrapper의 테스트 시간과 테스트 면적을 동시에 최적화 시킬 수 있는 설계 기법이 필요하다. 본 논문에서는 최소 비용의 SOC 테스트를 위한 효율적인 코어 테스트 wrapper 설계 기법을 제안한다. 본 논문에서 제안하는 기법은 기존의 기법들이 각기 가지고 있는 장점들을 하나로 취합하고 더욱 발전시킴으로써 필드에서 실제적으로 사용될 수 있는 효율적인 코어 테스트 wrapper 설계 기법이다.

키워드 : VLSI 테스트, SOC, Wrapper, TAM

Abstract With an emergence of SOC from developed IC technology, the VLSI design has required the core re-use technique and modular test development. To minimize the cost of testing SOC, an efficient method is required to optimize the test time and area overhead in conjunction for the core test wrapper, which is one of the important elements for SOC test architecture. In this paper, we propose an efficient design strategy of core test wrapper to achieve the minimum cost for SOC testing. The proposed strategy adopted advantages of traditional methods and more developed to be successfully used in practice.

Key words : VLSI Test, SOC, Wrapper, TAM

1. 서 론

최근 IC 집적기술이 고도로 발달함에 따라 수백만 개의 트랜지스터를 한 개의 칩에 집적할 수 있는 SOC (System On a Chip)가 출현, 거대한 시스템을 단 한 개의 칩에 구현할 수 있게 되었다. 이러한 SOC는 설계 때마다 필요한 모듈을 매번 설계하는 것이 아니라 미리 설계된 모듈, 즉 코어(core)를 이용하여 구성한다[1-3]. 이러한 코어의 사용은 SOC의 모듈러(modular) 설계 기법을 가능하게 하였으며 아울러 회로설계에 있어서 재사용 설계 기법을 도입함으로써 칩의 제품 출시 기간(time-to-market)을 단축시켰다[4-6]. 또한 SOC의 설계 과정의 전체기간 중 테스트 과정이 병목(bottleneck)이 되는 것을 방지하고, 또한 테스트에 소요되는 비용과 시간을 최소화하기 위하여 테스트 기법에도 모듈러 설계 기법을 도입하게 되었다. 즉, 코어의 제공자(vendor)가 SOC 설계자에게 코어를 제공함과 동시에 코어의 테스트 패턴 셋도 함께 제공하는 것이다.

SOC에 내장되어 있는 각각의 코어들을 미리 제공된 테스트 패턴 셋을 이용하여 외부에서 테스트하기 위해서는, 테스트 대상 코어(core-under-test)가 SOC의 입출력 포트에서 접근(access)이 가능해야 한다. SOC에 내장되어 있는 코어들은 칩의 입출력 포트에서 직접 접근이 불가능하기 때문에, 이를 위한 새로운 테스트 접근 기법이 필요하게 되었다. SOC 테스트 표준화 그룹인 IEEE P1500 등에서 SOC 테스트 접근 기법을 현재 개발 중에 있다[7,8].

SOC 테스트를 위한 구성요소 중 핵심요소인 코어 테스트 wrapper를 설계하기 위해서는, 코어의 내부 스캔

· 본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음
† 비 회 원 : 숭실대학교 컴퓨터학과
mjkim@watt.ssu.ac.kr
†† 비 회 원 : 숭실대학교 컴퓨터학부 교수
hoon@compting.ssu.ac.kr
논문접수 : 2003년 3월 24일
심사완료 : 2003년 12월 2일

체인과 코어의 입출력 터미널들의 균형있는(balanced) 연결을 위한 테스트 스케줄링 알고리즘이 필요하다. 이러한 테스트 스케줄링의 최적화를 위한 여러 가지 코어 테스트 wrapper 설계 기법들이 최근 몇 년 사이 집중적으로 연구되어 졌다[9,10]. 이들 연구의 목적은 SOC 테스트를 위해 소요되는 전체 테스트 시간과 테스트 면적 오버헤드를 동시에 최적화하는 것이다. 본 논문에서는 이러한 SOC 테스트를 위한 효율적인 코어 테스트 wrapper 설계 기법을 제안한다.

본 논문은 다음과 같이 구성되어진다. 2장에서 코어 테스트 wrapper의 정의와 설계에 관해 간략히 설명한다. 3장에서는 기존에 연구되어진 코어 테스트 wrapper 설계 기법들에 대해 알아보고 그 기법들의 장단점을 설명한다. 4장에서는 본 논문에서 제안하는 효율적인 코어 테스트 wrapper 설계 알고리즘을 설명하고 5장에서 ITC'02 SOC 테스트 벤치마크 회로를 이용한 실험 결과를 보인다[7]. 마지막으로 6장에서는 결론을 맺는다.

2. 코어 테스트 Wrapper의 정의

2장에서는 SOC 테스트를 위한 구성요소인 코어 테스트 wrapper의 정의와 설계에 대해 알아본다. SOC 테스트를 위한 구조는 현재 개발 중에 있으며 현재까지 결정된 3가지 필수 구성요소는 다음과 같다[7,8].

- 테스트 패턴 소스/싱크(Source/Sink) : 소스는 코어들을 위한 테스트 패턴을 생성하며 싱크는 데스트 응답과 예상응답을 비교한다.
- 테스트 접근 장치(Test Access Mechanism) : TAM은 소스에서 테스트 대상 코어까지의 테스트 패턴을 운송하는 고속도로 역할을 하며, 또한 테스트 응답을 테스트 대상 코어로부터 싱크까지 운송하는 버스(bus)이다.
- 코어 테스트 Wrapper : 코어 테스트 wrapper는 코어와 SOC의 나머지 회로환경 사이에 존재하여 외부에서 테스트 대상 코어로 접근하기 위한 인터페이스 역할을 담당한다. 코어 테스트 wrapper는 코어의 동작 모드를 정상(normal) 모드에서 테스트 모드로 전환시켜주는 역할 등을 수행한다.

그림 1은 위의 3가지 요소를 포함하는 SOC 테스트를 위한 구조의 개략적인 모습을 보인다.

코어 테스트 wrapper는 SOC에 내장된 코어와 해당 코어를 둘러싼 SOC 회로 사이에 존재하며, 테스트 시에 이 둘 사이의 인터페이스 역할을 수행한다. 최근 몇 년 사이에 테스트 칼라[11], 테스트 쉘[12]등 코어 테스트 wrapper의 이전 모델들이 고안되었으며 IEEE P1500 SOC 테스트 표준화 그룹에서 최소한의 필수 모드를 포함한 코어 테스트 wrapper의 구조를 개발 중에

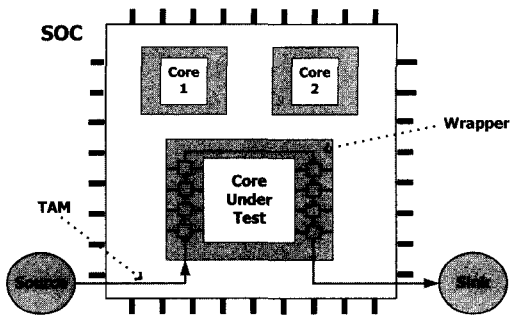


그림 1 SOC 테스트를 위한 구조

있다. 코어 테스트 wrapper는 IEEE 1149.1[13]과 형태가 유사하며, 4가지의 테스트 필수모드를 가진다[7].

- 기능 모드(functional mode) : 정상 모드(normal mode)라고도 하며 코어들이 설계시에 기술된 상태로 동작하도록 한다.
- 코어 테스트 모드(core test mode) : 테스트 대상 코어에 테스트 패턴을 인가할 수 있도록 설정한다.
- 연결선 테스트 모드 (interconnect test mode) : 코어와 코어 사이에 존재하는 연결선과 로직(logic)을 테스트한다.
- 테스트 분리 모드 (test isolation mode) : 다른 코어 및 로직의 테스트 시에 해당 코어의 입출력을 불능시킨다.

이처럼 코어 테스트 wrapper는 기능 모드 시에 코어의 입출력 터미널들을 SOC의 기능 회로부분에 연결하고, 테스트 모드 시에는 코어의 입출력 터미널들과 TAM을 연결하는 스윗칭 인터페이스 역할을 책임진다.

코어 테스트 wrapper를 설계할 때 고려해야하는 주요 파라미터들은 코어의 입력터미널의 개수, 출력 터미널의 개수, 그리고 내부 스캔 체인의 개수 등으로 구성된다. 코어 테스트 wrapper에 테스트 패턴을 운송하는 역할을 수행하는 TAM 라인 수는 소스와 싱크의 대역폭에 따라 결정되며, 코어의 터미널 개수와 반드시 동일할 필요는 없다. 또한 각각의 TAM 라인의 테스트는 병렬로 동시에 수행되어지며, 한 TAM 라인에 포함된 스캔 체인들은 순차적으로 테스트된다. 그림 2는 이러한 코어 테스트 wrapper의 설계에 필요한 파라미터들과 TAM 할당의 예를 나타내고 있다.

그림 2의 예에서 보이고 있는 코어는 3개의 입력 터미널과 1개의 출력 터미널, 그리고 각각의 길이가 5, 6, 10인 내부 스캔 체인 3개를 가지고 있다. 입력 터미널에 테스트 패턴을 인가하거나 출력 터미널에서 테스트 응답을 추출하기 위해 wrapper 셀(cell)을 각 터미널 앞에 배치한다. 주어진 TAM 라인 수가 2비트라고 가정하면,

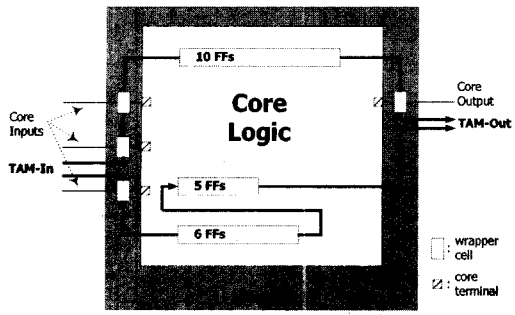


그림 2 코어 테스트 wrapper 구성의 예

그림 2와 같이 할당해야만 코어를 최소한의 시간으로 테스트할 수 있다. 위의 구성으로 코어 테스트 wrapper를 설계하면 두 TAM 라인의 테스트 패턴 입력 시간은 모두 12, 그리고 테스트 응답 시간 11로써 어느 한 쪽의 TAM 라인의 지연 없이 동일한 시간에 코어 테스트를 병렬로 수행할 수 있다. 다시 말해서, 각 TAM 라인의 테스트 시간이 균등하게 분배되도록 코어 테스트 wrapper를 설계함으로써, 주어진 TAM 라인 수 이내에서 코어의 총 테스트 시간을 최소화할 수 있다.

코어 테스트 wrapper 설계 알고리즘은 크게 3단계로 이루어진다.

- 내부 스캔 체인 할당 단계 - 내부 스캔 체인을 각 TAM 라인에 할당한 후 최장 TAM 라인의 길이가 최소화되도록 구성한다.
- 코어 입력 터미널 할당 단계 - 내부 스캔 체인 할당이 완료된 후 wrapper 입력 셀을 TAM 라인에 분배하여 할당한다. 단, TAM 라인의 최장 스캔 입력 시간이 최소화되도록 구성한다.
- 코어 출력 터미널 할당 단계 - 내부 스캔 체인 할당과 코어 입력 터미널 할당이 모두 완료된 후 wrapper 출력 셀을 TAM 라인에 분배하여 할당한다. 단, TAM 라인의 최장 스캔 출력 시간이 최소화되도록 구성한다.

코어 테스트 wrapper의 총 테스트 시간 및 면적 오버헤드는 테스트 대상 코어의 wrapper 체인의 구성에 전적으로 의존한다. 테스트 패턴의 스캔 입력 시간을 Tin, 테스트 응답의 스캔 추출 시간을 Tout이라고 하고, 해당 코어에 대한 테스트 패턴의 개수를 P라고 했을 때, 코어를 위한 총 테스트 시간 T는 식 1과 같이 표현할 수 있다.

$$T = \{1 + \max(Tin, Tout)\} \times P + \min(Tin, Tout) \quad (1)$$

코어를 테스트하기 위해서는 각 테스트패턴에 대해서 이를 인가하고 테스트응답을 추출해야 한다. 코어테스트를 위한 입력의 구성요소는 wrapper 입력 셀과 내부스캔채인이며 출력의 구성요소는 wrapper 출력 셀들과 내부스캔채인이다. 코어테스트를 짧은 시간에 완료하기 위해서 다음 테스트패턴을 인가시키고 현재 테스트응답을 추출하는 과정을 동시에 진행시켜야 한다. 즉, 각각의 테스트 패턴이 인가되었을 때 스캔 테스트에 소요되는 시간은 한 클럭이며, P개의 스캔 테스트 패턴이 각각 인가되는 시간은 입력구성요소와 출력구성요소 모두가 한번에 적재, 출력되는 조건을 만족해야하므로 전체 테스트 시간 T는 식 1과 같이 나타내어진다. 예를 들어, 그림 2의 코어를 위한 테스트 패턴의 개수를 10개라고 가정하면, $T=\{1+max(12,11)\}\times 10+min(12,11)$이므로 코어의 총 테스트 시간은 141이 된다. 코어 테스트 wrapper의 테스트 시간 비용은 코어 테스트 wrapper의 설계가 완료된 후, 모든 TAM 라인 중 가장 긴 라인의 테스트 시간으로 결정되며, 면적 오버헤드는 할당에 소요된 TAM 라인 수로 결정된다. 이와 같이 본 논문에서는 코어 테스트에 소요되는 테스트 시간과 면적 오버헤드의 최적화를 위한 효율적인 테스트 코어 wrapper 설계 알고리즘을 제안하고자 한다. 다음 절에서는 기존의 테스트 코어 wrapper 설계 알고리즘에 대해 설명한다.

3. 기존의 연구 기법들

3.1 COMBINE 알고리즘

테스트 코어 wrapper의 설계를 위해서는 우선 내부 스캔 체인들을 주어진 TAM 라인에 균등하게 분배한 후 입출력 터미널을 할당하는 방식을 취한다. TAM 라인의 최장 테스트 시간이 최소화되도록 스캔 체인들을 할당하는 문제는 기존의 멀티 프로세서 스케쥴링 문제, 빈 할당(bin-packing) 문제와 동일하며, 이는 NP-난해(NP-hard) 문제로 분류된다[9]. 이러한 문제들을 위해 개발된 대표적인 알고리즘인 LPT(Largest Processing Time) 기법[14]과 MULTIFIT 기법[15]을 조합한 알고리즘이 1988년에 개발되고 널리 사용되었다[16]. COMBINE 기법[9]은 최근에도 대표적으로 사용되고 있는 이러한 알고리즘에 기초를 두어 개발된 테스트 코어 wrapper 설계 기법으로써, 멀티 프로세서 스케쥴링 문제에서의 프로세서를 테스트 코어 wrapper 설계 문제의 TAM 라인으로, 프로세서에 할당하는 작업은 스캔 체인으로 각각 일대일 매핑하였다. 이러한 COMBINE 기법의 의사 코드(pseudo code)는 아래와 같다. 단, y개의 코어 내부 스캔 체인의 집합 $S=\{S_1, S_2, ..., S_y\}$라고 하며 스캔 체인 S_i의 길이를 $Length(S_i)$로 표현하기로 한다. 주어진 m비트 TAM 라인의 집합 $TL=\{TL_1, TL_2, ..., TL_m\}$라고 하며 TL_i의 길이를 $Length(TL_i)$로 표현하기로 한다.

```
COMBINE
do      S descending sort        // Length(S1)≥Length(S2)≥...≥Length(Sy)로 S를 정렬
A ← 0
for i ← 1 to y
        do      A ← A + Si/m
X ← LPT()
if X < 1.5*A
        then    CU ← X
                CL ← ⌊max(X/(4/3-1/3m), Si, A)⌋
                i ← CL
        while   i ≤ CU ∧ FFD(i) > m
                do      i ← i + 1
        return the FFD solution
else    then
        return the LPT solution
```

COMBINE 기법은 LPT와 FFD 함수를 특정 조건에 따라 선택적으로 사용한다. 우선 LPT 함수를 수행하여 최장 테스트 시간 X가, 주어진 스캔 체인 길이를 TAM 라인의 개수로 나눈 TAM 라인의 이상적인 길이 A의 1.5배 이상일 경우, LPT 함수의 수행 결과 얻어진 테스트 시간이 최적이라 결정하여 프로그램을 종료한다. 만약 1.5배 미만일 경우에는 2차적으로 FFD함수를 반복적으로 수행하여 코어 테스트 wrapper의 설계를 완료한다. 하위 함수로 사용되는 LPT 함수와 FFD 함수의 의사코드는 다음과 같다.

```
LPT
do      S descending sort
do      TLj = NULL for all j
for i ← 1 to m
        do      TW ← Si
for i ← m+1 to y
        do      select k ∈ {j | Length(TLj) = min1≤x≤m Length(TLx)}
                TLk ← TLk ∪ {Si}
return  max1≤x≤m Length(TLx)

FFD(C)
do      S descending sort
do      TLj = NULL for all j
do      j ← 1
for i ← 1 to y
        while (1)
                if  Length(TLj) + Si ≤ C
                        then
                        do      TLj ← Si;       break;
                else
                        then    j ← j + 1
return  j
```

LPT 함수는 항당 대상인 내부 스캔 체인들을 길이의 내림차순으로 정렬한 후, 각각의 스캔 체인을 현재까지 할당된 스캔 체인 길이의 합이 가장 작은 TAM 라인에 할당하는 간단한 함수이다. FFD(First Fit Decreasing) 함수는 TAM이 할당받을 수 있는 스캔 체인의 총 길이 합의 한계치(bound) C를 인자로 받아, C를 초과하지 않는 TAM 라인을 순차적으로 검색하여 내림차순으로 정렬된 각각의 스캔 체인을 할당한다. 만약 한계치 C를 초과할 경우 새로운 TAM 라인에 해당 스캔 체인을 할당한다. COMBINE 기법에 활용된 FFD 함수는, TAM이 가질 수 있는 테스트 길이의 한계치 C의 하한치 C_L과 상한치 C_H를 결정한 후에, C_L에서부터 C_H까지 순차 검색(linear search)을 통하여 코어 테스트 wrapper를 위한 최소의 테스트 시간 C를 찾아낸다. 즉, C_L에서부터 C_H까지 해당 C값을 TAM이 가질 수 있는 테스트 길이의 한계치로 가정하고, 매번 FFD를 수행하여 그 결과 주어진 TAM 라인 수에 스캔 체인을 모두 할당 가능하다면 알고리즘을 종료한다.

이와 같이, COMBINE 기법은 테스트 시간만을 고려했을 경우, [16]에서 증명한 위의 조건을 만족할 경우 LPT 함수의 수행 결과가 최적이라는 점에 기반한 기법이다. 이처럼 COMBINE 기법은 테스트 시간을 우선적으로 고려하여 테스트 코어 wrapper의 설계를 수행하였다. 하지만 동일한 테스트 시간이 인자로 주어지고 이를 FFD 함수를 수행할 경우, LPT 함수보다 더 적은 TAM 라인을 소모하면서도 동일한 테스트 시간을 산출하는 결과를 얻는 경우가 존재한다. 이 경우, 최장 테스트 시간이 동일하다면 더 적은 TAM 라인을 소비하는 할당 구조를 따르는 것이 효율적인 코어 테스트 wrapper 설계 방법이다. 본 논문에서는 이러한 점에 기반하여 테스트 시간과 면적 오버헤드의 동시 최적화를 위한 기법을 제안한다.

3.2 Design_wrapper 알고리즘

Design_wrapper 기법[10]은 기존의 COMBINE 기법이 동일한 최장 wrapper 테스트 시간에도 불구하고 필요 이상의 TAM 라인을 소비한다는 단점을 수정한 기법이다. 알고리즘은 다소 간단하며 의사코드는 다음과 같다.

```
Design_wrapper
do      S descending sort
do      TLj = NULL for all j
for i ← 1 to y
        do      select k ∈ {j | Length(TLj) = min1≤x≤m Length(Smax)-(Length(Si)+Length(TLx))}
                TLk ← TLk ∪ {Si}
        if      no such TL
                then
                if      no available TL
                        then
                        do      select k ∈ {j | Length(TLj) = min1≤x≤m Length(TLx)}
                                TLk ← TLk ∪ {Si}
                else
                        then
                        do      TLmcd+1 ← TLucd+1 ∪ {Si}
return the solution
```

Design_wrapper 기법은 기존의 전통적인 알고리즘 체계를 따르지 않고 함수를 단순화시켰다. 우선 할당할 스캔 체인들을 길이의 내림차순으로 정렬한다. 그리고 스캔 체인이 할당되었을 때 해당 TAM 라인의 길이와 현재의 최장 TAM 라인의 길이와의 차이가 최소화되는 TAM 라인을 검색하여 해당 스캔 체인을 할당한다. 단, 어느 TAM 라인에 할당해도 현재의 최장 TAM라인의 길이를 초과할 경우에는 새로운 TAM 라인에 할당한다. 이 기법은 가능한 최소한의 TAM 라인만을 사용한다는 장점이 있지만 알고리즘이 단순하여 역으로 최장 테스트 시간이 기존 기법들보다 더 길다는 단점이 존재한다. 이는 최소한의 테스트 시간이 소요되는 것을 우선으로 하는 코어 테스트 wrapper 설계 알고리즘에 있어서 치명적인 오류라 할 수 있다.

4. 제안된 코어 테스트 Wrapper 설계 기법

4장에서는 본 논문에서 제안하는 효율적인 코어 테스트 wrapper 설계 기법에 대해 설명한다. 본 논문에서 제안하는 기법은 NP-난해로 분류되어지는 코어 테스트 wrapper 설계 문제를 위해 LPT 함수와 BFD(Best Fit Decreasing) 함수[17]를 병렬적으로 사용한다. 기존의 COMBINE 기법을 기반으로 하지만, H/W 오버헤드의 최소화를 위해 핵심 알고리즘을 FFD 함수에서 BFD 함수로 교체하였고, 또한 최소한의 코어 테스트 wrapper 테스트 시간을 위해 기존의 방식과 달리 BFD 함수를 무조건적으로 수행한다. 그리고 미리 보존된 LPT 함수의 결과와 BFD 함수와의 최종 비교를 통하여 최소한의 비용을 필요로 하는 코어 테스트 wrapper의 설계 구조를 결정하였다. 1절에서는 BFD 함수에 대해 설명하고 2절에서 LPT 함수와 BFD 함수의 병렬 사용에 대해 설명한다.

4.1 내림차순 최량 적합(Best Fit Decreasing) 알고리즘

COMBINE 기법에서는 특정 조건을 만족하지 않을 경우 FFD 함수를 선택 수행하여 TAM 길이의 한계치 C를 만족하는 최적의 TAM 라인 수 m을 결정한다. 그러나 본 논문에서는 FFD 함수가 아닌 BFD 함수를 사용한다. BFD 함수는 주어진 TAM 길이의 한계치 C에 대하여 최적의 TAM 라인 수 m을 결정한다는 면에서 FFD 함수와 유사하지만, 각각의 스캔 체인을 할당할 때 TAM 라인을 검색하는 방법에서 차이가 존재한다. 우선 스캔 체인 집합 S를 내림차순으로 정렬한다. 그리고 각각의 스캔 체인을 할당할 때마다 검색할 TAM 라인을 길이에 대해 내림차순으로 정렬한다. 그리고 나서 FFD 함수와 마찬가지로 한계치 C를 초과하지 않는 TAM 라인을 검색하여 스캔 체인을 할당한다. BFD 함수의 의사 코드는 다음과 같다.

```
BFD
do      S descending sort
do      TL_j = NULL for all j
do      j ← 1
for i ← 1 to y
        do      TL descending sort
        while (1)
                if  Length(TL_j) ≤ C and (TL_j + S_i) ≥ (TL_i + S_i)
                        then
                        do      TW_j ← S_i;     break;
                else
                        then    j ← j + 1
return  j
```

즉 FFD 함수에, 각각의 스캔 체인을 할당할 때마다 각 TAM 라인에 할당된 스캔 체인 길이의 합에 대해 내림차순으로 TAM 라인을 정렬하는 부분만을 추가한 것이다. BFD 함수는 FFD 함수보다 경험적으로 더 나은 할당 결과를 보이며, 특히 스캔 체인 길이의 분포 범위가 넓은 경우 뛰어난 성능을 보인다[17]. 한 예를 들어, 코어의 내부 스캔 체인의 개수가 6개이고 각각의 길이가 20, 11, 10, 5, 3, 3, 그리고 주어진 TAM 길이의 한계치 C가 26으로 주어질 경우, FFD 함수와 BFD 함수를 각각 수행한 결과는 다음의 표 1와 같다.

표 1 FFD의 TAM 할당 결과

TAM	FFD	BFD
TAM[0]	{20, 5}	{20, 3, 3}
TAM[1]	{11, 10, 3}	{11, 10, 5}
TAM[2]	{3}	

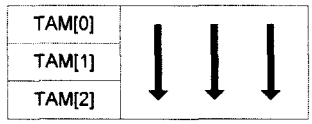
FFD 함수는 길이 20과 11에 해당하는 스캔 체인을 순차적으로 TAM[0]과 TAM[1]에 할당하고 길이 10에 해당하는 스캔 체인을 TAM[1]에 할당한 후, 길이 5의 스캔 체인을 검색 순위가 높은 TAM[0]에 할당한다. BFD 함수는 20, 11, 10에 해당하는 스캔 체인을 FFD 함수와 동일하게 할당한 후, 길이 5에 해당되는 스캔 체인을 TAM[1]에 할당한다. 이는 스캔 체인을 할당할 때 TAM을 길이의 내림차순으로 정렬하면 TAM[0]보다 TAM[1]을 우선적으로 검색하기 때문이다. 모든 스캔 체인에 대한 할당이 종료되면 FFD 함수는 3개의 TAM 라인을 사용하고, BFD 함수는 2개의 TAM 라인만을 사용하게 된다. 본 논문에서는 경험적으로 더 나은 성능을 보이는 BFD 함수를 코어 테스트 wrapper 설계를 위한 핵심 알고리즘으로 사용한다.

4.2 큰 처리 시간(Largest Processing Time) 알고리즘

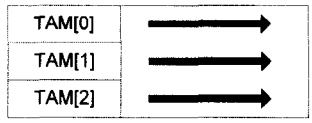
COMBINE 기법이 특정 조건에 따라 LPT와 FFD, 둘 함수 중 하나의 함수만을 선택 수행한 것과 달리, 본 논문에서 제안하는 테스트 코어 wrapper 설계 기법에서는 LPT 함수의 수행 이후 BFD 함수를 무조건적으로 수행한 다음, 두 결과를 비교하여 최종 할당 결과를 선택한다. 즉 BFD 함수와 LPT 함수를 병렬적으로 사용한다. 그 이유는 두 가지로써 첫째, FFD 함수와 BFD 함수보다 LPT 함수의 결과가 더 우수한 경우가 존재한다. BFD 함수는 주어진 스캔 체인을 TAM에 할당하기 위해 각 TAM 길이의 한계치 C를 가정한다. 본 논문에서는 COMBINE 기법과 마찬가지로 LPT 함수의 수행 결과로 얻어진 테스트 시간을 BFD 함수의 C의 상한치, 즉 CH로 놓는다. 다시 말해서 LPT 함수의 수행 결과를 최소한으로 얻을 수 있는 테스트 시간으로 가정하는 것이다. 하지만 LPT 함수가 결정한 테스트 시간이 FFD 함수나 BFD 함수에 동일하게 주어진다 해도, 주어진 개수의 TAM 라인에 주어진 스캔 체인 셋을 할당

하지 못하는 경우가 존재하고 결과적으로 주어진 TAM 라인 수를 초과하여 사용하게 된다[16]. COMBINE 기법은 LPT 함수와 FFD 함수를 선택적으로 사용하기 때문에 FFD 함수를 단독 사용하는 경우가 발생하고, 이와 같이 주어진 TAM 라인 수를 초과하여 사용할 경우 테스트 코어 wrapper 설계 문제에 치명적일 수 있다.

BFD 함수와 LPT 함수를 병렬적으로 사용하는 두 번째 이유는, 두 함수가 모두 동일한 테스트 시간에 주어진 스캔 체인 셋을 할당할 수 있는 경우, LPT 함수는 항상 FFD나 BFD 함수보다 같거나 많은 수의 TAM 라인 수를 소모하기 때문이다. 그 이유는 그림 3에 나타난 바와 같이 두 함수의 알고리즘이 스캔 체인을 TAM에 할당하는 순서의 방향성 때문이다.



(a) LPT



(b) BFD

그림 3 LPT 알고리즘과 BFD 알고리즘의 비교

FFD 함수나 BFD 함수가 TAM의 테스트 시간의 한계치 C를 초과하지 않는 한 새로운 TAM 라인을 사용하지 않고 동일한 TAM 라인에 연속적으로 스캔 체인을 할당하는 것과는 달리, LPT 함수는 주어진 모든 TAM 라인에 우선적으로 1개씩의 스캔 체인을 배분한다. 즉 LPT 함수는 스캔 체인의 개수가 TAM 라인의 개수보다 같거나 클 경우, 주어진 모든 TAM 라인을 소비하게 된다. 따라서 LPT 함수는 동일한 테스트 시간을 가지는 FFD 혹은 BFD 함수의 할당 결과보다 항상 같거나 더 많은 수의 TAM 라인을 소비한다. 기존의 COMBINE 기법은 TAM 라인 수 할당에 관계없이 최소의 테스트 시간을 우선시하였기 때문에, LPT 함수의 테스트 시간이 최적인 경우 필요 이상의 TAM 너비를 사용하는 LPT 함수의 할당 결과를 그대로 테스트 코어 wrapper 설계의 구성으로 결정하였다.

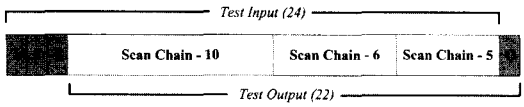
이와 같은 두 가지 문제점을 해결하기 위해, 본 논문에서 제안하는 wrapper 설계 기법은 최소한의 H/W 오버헤드를 위해 BFD 함수를 무조건적으로 수행하며, 최소한의 테스트 시간 소모를 위해 BFD 함수의 결과를 LPT 함수의 결과와 비교하여 최종 할당 구조를 결정한다. 만일 BFD 함수가 주어진 TAM 라인 수를 초과 사용한 경우, LPT 함수의 결과를 복원하여 TAM 할당 구조를 결정한다. 다음은 본 논문에서 제안하는 코어 테스트 wrapper 설계 알고리즘의 의사코드이다.

```
COMBINE_Extended
do      S descending sort
A ← 0
for i ← 1 to y
        do      A ← A + Si/m
X ← LPT()
save the LPT solution
CU ← X
CL ← ⌈ max(X/(4/3-1/3m), S1, A) ⌉
do      i ← CL
while   i ≤ CU ∧ BFD(i) > m
        do      i ← i + 1
if      i > CU
        then
        do      restore the LPT solution
return the solution
```

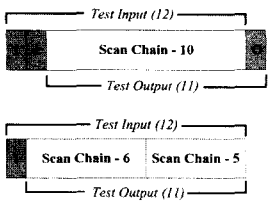
제안된 알고리즘 결과에 따라 주어진 TAM 너비에 맞게 내부스캔 채인을 할당한 후 입력터미널과 출력터미널의 wrapper 셀을 할당한다. 이는 별도의 알고리즘을 사용하지 않고 wrapper 셀을 각각의 길이가 1인 내부스캔채인으로 가정하고 동일하게 알고리즘을 수행하여 완료한다.

예를 들어 그림 2에서 소개된 코어 예제를 위해 본 알고리즘을 이용하여 wrapper를 설계할 경우 그림 4와 같이 이루어진다. 코어 예제는 입력터미널 3개, 출력터미널 1개, 그리고 길이가 각각 10, 6, 5인 3개의 내부스캔채인을 가진다.

그림 4의 괄호 안의 숫자는 테스트패턴의 입력과 출력에 소요되는 클럭 수를 나타낸다. 그림 4(a)는 한 개의 TAM 비트를 사용하는 경우, 그리고 그림 4(b)는 두 개의 TAM 비트를 사용하는 예이다. 그림 4(b)에서 각각의 TAM 비트에 소요되는 테스트입력(출력) 시간 중



(a) TAM 너비가 1인 경우



(b) TAM 너비가 2인 경우

그림 4 제안된 기법을 이용한 Wrapper 설계의 예

가장 큰 수치가 전체 TAM 테스트 입력(출력) 시간을 결정한다. 본 예제에서는 각 TAM의 테스트 입력시간과 출력시간이 동일하므로 각각 12와 11로 결정되어진다.

5장에서는 테스트 코어 wrapper 설계를 위한 기존의 COMBINE 기법과 Design_wrapper 기법, 그리고 본 논문에서 제안하는 기법을 벤치마크 회로에 적용하여 실험하고 결과를 분석한다.

5. 실험 결과

5장에서는 본 논문에서 제안하는 코어 테스트 wrapper 설계 알고리즘의 우수성을 입증하기 위해 ITC'02 SOC 테스트 벤치마크 회로를 실험에 이용하였다. 실험에 사용된 코어 회로는 [10]에서의 실험과 마찬가지로, ITC'02 SOC 테스트 벤치마크 회로 중 p93791의 모듈 6 코어를 사용하였다. 모듈 6 코어의 특성은 표 2에 나타난 바와 같다.

표 2 p93791의 모듈 6 코어 특성

입출력 단자			스캔 체인	
입력	출력	양방향	길이	수
417	324	72	500	7
			520	30
			521	9

모듈 6 코어는 417개의 입력터미널, 324개의 출력터미널, 그리고 72개의 양방향터미널을 가지고 있으며, 총 46개의 내부 스캔 체인을 포함하고 있다. 표 3은 COMBINE 기법을 벤치마크 회로에 적용한 결과이다.

첫 번째 열에는 가정으로 주어진 TAM 라인 수를 나타낸다. TAM 라인 수가 1비트부터 64비트까지 주어지는 경우를 실험한다. 두 번째 열은 주어진 TAM 라인 수를 초과하지 않도록, 코어의 스캔 체인과 입출력 터미널들을 알고리즘에 따라 할당한 후 최종 사용된 TAM 라인 수를 나타낸다. 세 번째 열은 전체 할당 과정이 완료된 후, 사용된 TAM 라인 중 최장 TAM 라인의 테스트 시간, 즉 식 1에서 설명된 $max(T_{in}, T_{out})$값을 나타낸다.

표 3에서 나타나듯이 주어진 TAM 라인 수가 넓어질수록 코어의 총 테스트 시간이 점차 줄어드는 것을 알 수 있다. 이는 코어의 내부 스캔 체인이 더 많은 수의 TAM 라인에 분배될 경우 각 TAM 라인에 할당되는 스캔 체인의 양이 줄어들기 때문이다. 주어진 TAM 라인 수가 좁은 경우에는 보통 모든 TAM 라인을 사용하지만, 주어진 TAM 라인 수가 넓어질수록 모든 TAM 라인을 소요하지 않으면서도 동일한 최장 테스트 시간을 얻는 경우가 다수 존재한다. 예를 들어 주어진 TAM 라인 수가 14인 경우 모든 TAM 라인을 소비하지 않고 13개만을 할당에 소비하면서 주어진 라인 수가 13일 때와 동일한 최장 테스트 시간 결과를 얻을 수 있다.

표 3 COMBINE 기법의 p93791의 모듈 6 코어 wrapper 설계 결과

Available TAM width	Utilized TAM width	Longest wrapper scan chain	Available TAM width	Utilized TAM width	Longest wrapper scan chain
1	1	24278	22	22	1500
2	2	11840	23	23	1056
3	3	8180	24-34	24	1040
4	4	6120	35		1040
5	5	5060	36		1040
6	6	4140	37		1040
7	7	3620	38		1040
8	8	3100	39	39	1020
9	9	3000	40		1020
10	10	2580	41		1020
11	11	2560	42		1020
12	12	2080	43	43	1000
13-14	13	2060	44		1000
15	15	2000	45		1000
16-19	16	1560	46	46	528
20-21	20	1540	47-64	47	521

표 4 Design_wrapper 기법의 p93791의 모듈 6 코어 wrapper 설계 결과

Available TAM width	Utilized TAM width	Longest wrapper scan chain	Available TAM width	Utilized TAM width	Longest wrapper scan chain
1	1	24278	13	13	
2	2	12139	14	14	2060
3	3		15	15	
4	4	6202	16-19	16	1560
5	5		20-21	20	1540
6	6		22	22	
7	7		23	23	1056
8	8		24-38	24	1040
9	9		39-42	39	1020
10	10		43-45	43	1000
11	11		46	46	528
12	12	2080	47-64	47	521

하지만 COMBINE기법은 표 3의 음영 처리된 부분의 경우와 같이, 결과로 얻어진 최장 테스트 시간에 비해 필요 이상의 TAM 라인 수를 낭비하는 결과를 보인다. 예를 들어 주어진 TAM 라인 수가 35인 경우, 최장 테스트 시간이 1040으로써 TAM 라인 수가 34일 때의 테스트 시간 결과와 동일하지만 TAM 라인을 한 비트 더 소비하고 있는 것을 볼 수 있다. 또한 세 번째 열의 밑줄 그어진 경우와 같이, COMBINE기법은 주어진 TAM 라인 수를 초과 사용하여 코어 테스트 wrapper를 설계하는 오류를 지니고 있다. 다음의 표 4는 모듈 6 코어에 Design_wrapper 기법을 적용시킨 결과이다.

Design_wrapper의 경우도 주어진 TAM 라인 수가 넓어질수록 코어의 총 테스트 시간은 점차 줄어드는 것을 알 수 있다. 또한 최장 테스트 시간이 동일한 경우, COMBINE 기법과는 달리 모든 경우에서 최소한의 TAM 라인 수만을 사용하는 것을 알 수 있다. 하지만 표 4의 음영 처리된 부분의 경우와 같이, Design_wrapper 기법은 최장 테스트 시간의 결과에 있어서 COMBINE 기법에 비해 좋지 않음을 알 수 있다. 표 5

표 5 제안된 기법의 p93791의 모듈 6 코어 wrapper 설계 결과

Available TAM width	TAM width utilized	Longest wrapper scan chain	Available TAM width	TAM width utilized	Longest wrapper scan chain
1	1	24278	13-14	13	2060
2	2	12139	15	15	2000
3	3	8180	16-19	16	1560
4	4	6202	20-21	20	1540
5	5	5060	22	22	1500
6	6	4140	23	23	1056
7	7	3620	24-38	24	1040
8	8	3100	39-42	39	1020
9	9	3000	43-45	43	1000
10	10	2580	46	46	528
11	11	2560	47-64	47	521
12	12	2080			

는 본 논문에서 제안하는 기법을 모듈 6 코어에 적용한 결과를 보인다.

본 논문에서 제안하는 코어 테스트 wrapper 설계 기법은 표 5와 같이 기존의 기법들이 가지는 단점들을 모두 보완하고 이를 더욱 발전시켰다. 코어 테스트 wrapper의 최장 테스트 시간을 나타내는 세 번째 열과 같이, 테스트 시간에 있어서 기존의 기법들에 비해 동일하거나 더 나은 결과를 산출한 것을 알 수 있다. 이와 동시에 사용된 TAM 라인 수를 나타내는 두 번째 열을 비교해 볼 때, 본 논문에서 제안하는 기법은 코어 테스트 wrapper 설계의 핵심 알고리즘으로써 BFD를 사용하여 전체적으로 기존의 기법들과 동일하거나 더 적은 TAM 라인 수의 사용을 보였다. 또한 두 번째 열의 굵은체로 쓰여진 부분과 같이, BFD 함수와 LPT 함수를 병렬적으로 사용함으로써 COMBINE 기법이 가지고 있던 TAM 라인 수의 초과 사용의 오류를 제거하였다. 이와같이 할당된 결과를 이용하여 주어진 TAM 너비에 따라서 Wrapper를 설계할 수 있다.

객관적인 검증을 위해 ITC'02 SOC 테스트 벤치마크 회로 중 다른 회로를 추가적으로 무작위 검출하여 실험에 사용하였다. 회로이름은 p22810의 모듈 26 코어이다. 모듈 26 코어의 특성은 표 6과 같으며, 이 회로에 대한 COMBINE, Design_wrapper, 그리고 제안된 기법에 기반하여 실험한 결과는 각각 표 7, 8, 9와 같다.

첫 번째 실험결과와 마찬가지로 COMBINE기법은 표

표 6 p22810의 모듈 26 코어의 특성

입출력 단자			스캔 체인	
입력	출력	양방향	길이	수
66	33	98	400	6
			399	16
			334	2
			333	3
			279	2
			278	1
			198	1

표 7 COMBINE 기법의 p22810의 모듈 26 코어 wrapper 설계 결과

Available TAM width	Utilized TAM width	Longest wrapper scan chain	Available TAM width	Utilized TAM width	Longest wrapper scan chain
1	1	11609	14	14	904
2	2	5805	15-21	15	798
3	3	3870	22	22	733
4	5	2862	23	23	732
5	5	2328	24		732
6	6	1935	25	25	678
7	7	1745	26	26	666
8	8	1530	27	27	612
9	10	1400	28	28	571
10	10	1197	29	29	517
11	11	1132	30	30	436
12	12	1131	31-64	31	400
13	14	1038			

표 8 Design_wrapper 기법의 p22810의 모듈 26 코어 wrapper 설계 결과

Available TAM width	Utilized TAM width	Longest wrapper scan chain	Available TAM width	Utilized TAM width	Longest wrapper scan chain
1	1	11609	14	14	
2	2		15	15	
3	3		16-21	15	798
4	4	2943	22	22	733
5	5		23-24	23	732
6	6		25	25	678
7	7		26	26	666
8	8	1530	27	27	612
9	9	1411	28	28	571
10	10		29	29	517
11	11	1132	30	30	436
12	12	1131	31-64	31	400
13	13	1066			

표 9 제안된 기법의 p22810의 모듈 26 코어 wrapper 설계 결과

Available TAM width	Utilized TAM width	Longest wrapper scan chain	Available TAM width	Utilized TAM width	Longest wrapper scan chain
1	1	11609	13	13	1066
2	2	5805	14	14	904
3	3	3870	15-21	15	798
4	4	2943	22	22	733
5	5	2328	23-24	23	732
6	6	1935	25	25	678
7	7	1745	26	26	666
8	8	1530	27	27	612
9	9	1411	28	28	571
10	10	1197	29	29	517
11	11	1132	30	30	436
12	12	1131	31-64	31	400

7의 밑줄 그어진 부분과 같이 주어진 TAM 라인 수를 초과 사용하는 것을 볼 수 있으며, 음영 처리된 부분의 경우와 같이 필요 이상의 TAM 라인 수를 낭비하는 결과를 보인다. 또한 Design_wrapper기법은 최대한 적은 수의 TAM 라인 수를 사용하지만 표 8의 음영 처리된 부분과 같이 최장 테스트 시간이 COMBINE 기법보다 더 길게 나타나는 것을 알 수 있다. 표 9는 본 논문에서 제안하는 기법이 이러한 두 기법의 모든 단점을 모두 극복하여 최대한 적은 TAM 너비를 사용하고 짧은 최장 테스트 시간을 사용함을 알 수 있으며, 이러한 결과를 다른 모든 ITC'02 SOC 테스트 벤치마크 회로에서

도 확인할 수 있다.

본 논문의 기법은 기존 기법들의 장점들만을 추출하여 병합하고 이를 더욱 확장시킴으로써 알고리즘 상의 복잡도는 다소 증가하였다. 하지만 본 논문의 실험이 이루어진 시스템 환경(Intel Pentium-4 1GHz CPU)에서의 전체 수행시간은 불과 1초 미만이었다. 따라서 본 논문에서 제안하는 효율적인 코어 테스트 wrapper 설계 기법은 NP-난해 문제인 테스트 코어 wrapper 설계 문제에 실제적으로 적용되어 사용되어질 수 있을 것으로 기대된다.

6. 결 론

본 논문에서는 SOC 테스트를 위한 비용을 최소화하는 효율적인 코어 테스트 wrapper 설계 기법을 제안하였다. 본 논문의 제안 기법은 기존의 기법들이 각기 가지고 있던 장점들을 모두 취합하고 이를 더욱 발전시킴으로써 최소한의 테스트 시간과 H/W 오버헤드를 동시에 만족시킨다. 본 논문에서 제안하는 코어 테스트 wrapper 설계 알고리즘은 기존의 핵심 알고리즘으로 사용된 FFD보다 경험적 우수성이 입증된 BFD를 사용하며, 최종적으로 LPT의 결과와 조합하여 코어 테스트 wrapper를 설계하였다. 본 논문에서 제안하는 효율적인 코어 테스트 wrapper 설계 기법은 테스트 시간과 H/W 오버헤드 비용의 동시 최소화를 만족시키고 무시해도 좋을 만큼의 시간만을 필요로 하므로, NP-난해 문제인 테스트 코어 wrapper 설계 문제에 실제적으로 적용될 수 있을 것으로 기대되어진다.

참 고 문 헌

[1] Y. Zorian, E.J. Marinissen, and S. Dey, "Testing embedded core-based system chips," *IEEE Computer*, Volume 32, Number, 6, pp.52-60, June 1999.

[2] Erik Jan Marinissen, Yervant Zorian, Rohit Kapur, Tony Taylor, Lee Whetsel, "Towards a Standard for Embedded Core Test: An Example," *Proc. International Test Conference*, pp.616-627, 1999.

[3] P1500 Scalable Architecture Task Force Members, "Preliminary Outline of the IEEE P1500 Scalable Architecture for Testing Embedded Cores," *Proc. VLSI Test Symposium*, 1999.

[4] Immaneni, V., and S.Raman, "Direct Access Test Scheme - Design of Block and Core Cells for Embedded ASICS," *Proc. International Test Conference*, pp.488-492, 1990.

[5] Peter Harrod, "Testing Reusable IP - A Case Study," *Proc. International Test Conference*, pp.493-498, 1999.

[6] Lee Whetsel, "Addressable Test Ports An Approach to Testing Embedded Cores," *Proc. International Test Conference*, pp.1055-1064, 1999.

[7] IEEE P1500 General Working Group website, *"IEEE P1500 Standards For Embedded Core Test,"* http://grouper.ieee.org/groups/1500/

[8] Yervant Zorian, Erik Jan Marinissen, and Sujit Dey, "Testing Embedded-Core Based System Chips," *Proc. International Test Conference*, pp.130-243, 1998.

[9] Marinissen, E.J., Goel, S.K., Lousberg, M., "Wrapper Design for Embedded Core Test," *Proc. International Test Conference*, pp.911-920, 2000.

[10] Iyengar. V., Chakrabarty. K., Marinissen, E.J., "Test wrapper and test access mechanism co-optimization for system-on-chip," *Proc. International Test Conference*, pp.1023-1032, 2001.

[11] Prab Varma, Sandeep Bhatia, "A Structured Test Re-Use Methodology for Systems on Silicon," *Proc. International Test Conference*, pp.294-302, 1998.

[12] E. J. Marinissen, R, Arendsen, G. Bos, H. Dingemanse, M. Lousberg, and C. Wouters, "A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores," *Proc. International Test Conference*, pp.284-293, 1998.

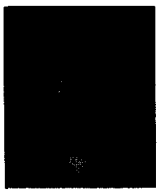
[13] IEEE Computer Society, *IEEE Standard Test Access Port and Boundary-Scan Architecture-IEEE Std. 1149.1-1990*, IEEE, New York, 1990.

[14] R.L. Graham, "Bounds on Multiprocessing Anomalies," *SIAM Journal of Applied Mathematics*, Volume 17, pp.416-429, 1969.

[15] E.G. Coffman Jr., M.R. Garey, D.S. Johnson, "An Application of Bin-Packing to Multiprocessor Scheduling," *SIAM Journal of Computing*, Volume 7, Number 1, pp.1-17, 1978.

[16] Lee, C.Y., D. Massey, "Multiprocessor Scheduling: Combining LPT and Multifit," *Discrete Applied Mathematics*, Volume 20, pp.233-242, 1988.

[17] H. Chao, M. P. Harper, R. W. Quong, "A Tighter Lower Bound for Optimal Bin Packing," *Operations Research Letters*, Volume 18, pp.133-138, 1995.



김 문 준

2000년 숭실대학교 컴퓨터학부 졸업(학사). 2002년 숭실대학교 대학원 컴퓨터학과 졸업(석사). 2002년~현재 숭실대학교 대학원 컴퓨터학과 박사과정. 관심분야는 VLSI 설계 및 테스트, 컴퓨터구조, VLSI CAD

장 훈

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. 관심분야는 VLSI 설계 및 테스트, 컴퓨터구조, VLSI CAD