논문 14-39B-01-01

한국통신학회논문지(J-KICS) '14-01 Vol.39B No.01
http://dx.doi.org/10.7840/kics.2014.39B.1.1

멀티코어 SoC의 테스트 시간 감축을 위한 테스트 Wrapper 설계

강 우 진◆, 황 선 영°

A Test Wrapper Design to Reduce Test Time for Multi-Core SoC

Woo-jin Kang◆, Sun-young Hwang°

요 약

본 논문은 멀티 코어 SoC의 전체 테스트 시간 감축을 위한 효율적인 테스트 wrapper 설계 방법을 제안한다. 제안된 알고리즘은 잘 알려진 Combine 알고리즘을 사용하여 멀티코어 SoC의 각 코어에 대해 초기 local wrapper 해 집합을 구성하고 가장 긴 테스트 시간을 소모하는 코어를 dominant 코어로 선택한다. Dominant 코어의 테스트 시간을 기준으로 다른 코어들에 대해 wrapper 특성인 TAM 와이어 수와 테스트 시간을 조정한다. Design space exploration을 위해 일부 코어들의 TAM 와이어 수를 줄이고 테스트 시간을 증가시킨다. 변경된 wrapper 특성을 기존 local wrapper 해 집합에 추가한다. 코어들의 기존 local wrapper 해 집합이 global wrapper 해 집합으로 확장되어 스케줄러에 의한 멀티코어 SoC의 전체 테스트 시간이 감소한다. 제안된 wrapper의 효과는 ITC'02 벤치마크 회로에 대해 B*-트리 기반의 테스트 스케줄러를 사용하여 검증된다. 실험 결과 기존의 wrapper를 사용하는 경우에 비해 테스트 시간이 평균 4.7% 감소한다.

Key Words : Multi-Core SoC, Test Wrapper, TAM, Test Time, Test Scheduling

ABSTRACT

This paper proposes an efficient test wrapper design that reduces overall test time in multi-core SoC. After initial local wrapper solution sets for all the cores are determined using well-known Combine algorithm, proposed algorithm selects a dominant core which consumes the longest test time in multi-core SoC. Then, the wrapper characteristics in the number of TAM wires and the test time for other cores are adjusted based on test time of the dominant core. For some specific cores, the number of TAM wires can be reduced by increasing its test time for design space exploration purposes. These modified wrapper characteristics are added to the previous wrapper solution set. By expanding previous local wrapper solution set to global wrapper solution set, overall test time for Multi-core SoC can be reduced by an efficient test scheduler. Effectiveness of the proposed wrapper is verified on ITC'02 benchmark circuits using B*-tree based test scheduler. Our experimental results show that the test time is reduced by an average of 4.7% when compared to that of employing previous wrappers.

◆ First Author : 서강대학교 전자공학과 CAD & ES 연구실, wj2201@sogang.ac.kr, 학생회원
° Corresponding Author : 서강대학교 전자공학과 CAD & ES 연구실, hwang@sogang.ac.kr, 종신회원
논문번호 : KICS2013-11-514, 접수일자 : 2013년 11월 27일, 심사일자 : 2013년 12월 10일, 최종논문접수일자 : 2013년 12월 12일

Ⅰ. 서 론

최근 멀티코어를 채택한 SoC의 사용이 증가하여 멀티코어 SoC의 테스트 시간을 효과적으로 줄이기 위한 연구가 필요하다. 코어 테스트를 위한 핵심 요소인 wrapper 설계 시 테스트 시간과 면적 오버헤드가 고려된다. 코어의 테스트 시간이 최소가 되도록 코어의 스캔 체인들을 한정된 TAM (Test Access Mechanism) 와이어에 partition하는 wrapper 설계는 NP-hard 문제로 분류된다. 주어진 TAM 와이어의 테스트 시간을 균등하게 분포시키는 LPT (Largest Processing Time)[1,2]와 테스트 시간을 제한조건으로 받아 최소한의 TAM 와이어 수를 사용하는 FFD (First Fit Decreasing)와 BFD (Best Fit Decreasing)[3,4], 그리고 LPT와 FFD를 선택적으로 사용하는 Combine 알고리즘[3]이 코어 wrapper 설계 알고리즘으로 제안되었다. 설계된 각 코어의 wrapper 구성은 멀티코어 SoC의 테스트 스케줄링에 해 집합으로 활용된다. 평면계획[5], Sequence Pair와 BSG (Bounded Slicing Grid)[6,7], 그래프와 3-D Bin Packing[8,9], 그리고 MILP (Mixed-Integer Linear Programming)[10] 등이 효율적인 멀티코어 SoC의 테스트를 위한 스케줄러로 제안되었다. 멀티코어 SoC는 모듈화 된 코어들로 구성되므로 전체 테스트 시간을 최소화하는 해 집합을 구성하는 wrapper 설계 방법이 요구된다.[11,13] 기존의 wrapper 설계 방법들에 의해 설계된 wrapper는 단일 코어에 치중하여 local 해 집합을 가지므로 이 해 집합을 활용한 멀티코어 SoC의 테스트 시간이 최소화되지 못한다. 본 논문에서는 테스트 스케줄링을 위해 코어들의 local 해 집합을 global 해 집합으로 확장시키는 코어 wrapper 설계 방법을 제안한다. 제안된 코어 wrapper 설계 방법은 코어들의 (TAM 와이어 수, 테스트 시간) 해 집합을 멀티코어 SoC 환경을 고려한 global 해 집합으로 확장하여 테스트 스케줄러의 design space exploration 후 멀티코어 SoC의 전체 테스트 시간을 감소시킨다.

논문의 구성은 다음과 같다. 2절에서는 코어 테스트의 기본 구성 요소와 기존의 wrapper 설계 방법과 멀티코어 SoC의 테스트 스케줄러에 대해 설명하고, 3절에서 본 논문에서 제안하는 알고리즘을 기술한다. 4절에서는 ITC'02 SoC 벤치마크 회로[14]를 이용한 실험 결과를 보이고, 5절에서 결론을 맺으며 추후과제를 논한다.

Ⅱ. 연구배경

본 절에서는 코어 테스트의 개념과 wrapper 설계에 대한 관련 연구를 기술하고, 멀티코어 SoC의 테스트 스케줄러에 대해 설명한다.

2.1 코어 테스트

코어 테스트를 위한 아키텍처는 테스트 패턴 소스와 싱크, TAM, 그리고 코어 테스트 wrapper로 구성된다. 소스는 테스트 패턴을 생성하고 TAM은 테스트 패턴을 wrapper로 전달하는 역할을 한다. wrapper는 DFT (Design For Testability)를 고려한 것으로 코어와 나머지 회로 환경 사이에 존재한다. 정상 모드에는 코어가 원래 설계된 기능대로 동작하게 하고, 테스트 모드에는 코어의 입출력 터미널들과 TAM을 연결하는 역할을 한다. 싱크는 테스트 패턴의 응답과 예상 응답을 비교한다. 그림 1은 이 세 가지 요소로 구성된 코어의 테스트 접근 방식을 보여준다.[13]

Wrapper의 내부와 외부 인터페이스의 하드웨어를 참고문헌[12]에서 제시한 바와 같이 그림 2에 보인다. Parallel TAM은 m개 와이어가 연결되어 WBR을 통해 일련의 테스트 패턴을 입력하고 코어의 테스트 응답을 전달받는다. Functional inputs은 코어에 따라 다른 특성을 가지고 코어의 동작을 결정하며, WBR (Wrapper Boundary Register)은 wrapper 가장자리에 위치하여 functional inputs과 parallel TAM에 의한

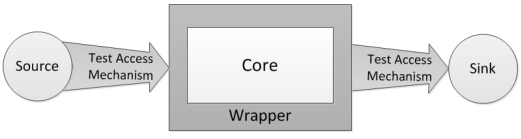

그림 1. 코어 테스트 접근 방식
Fig. 1. Overview of core test approach

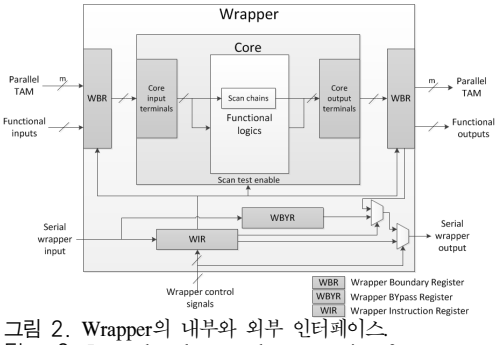

그림 2. Wrapper의 내부와 외부 인터페이스
Fig. 2. Internal and external wrapper interfaces

표 1. LPT, FFD 알고리즘의 pseudo 코드
Table 1. Pseudo code of LPT, FFD algorithm

```
LPT() // LPT algorithm
// m: number of TAM wires
// y: number of scan chains
// L(S_i): length of scan chain S_i
// W_i: TAM wire

(assumes m<y)
sort S_i such that L(S_1) ≥ L(S_2)-L(S_y);
for(i=1; i≤m; i++)   W_i = S_i
for(i=m+1; i≤y; i++)
  select k∈{j|L(W_j) = min_{1≤x≤m} L(W_x)};
    W_k = W_k ∪ S_i;
return max_{1≤x≤m} L(W_x);

FFD(C) // FFD algorithm
// C: constraint
sort S_i such that L(S_1) ≥ L(S_2)-L(S_y);
j=1;
for(i=1; i≤y; i++)
  while(1)
    if((L(W_j)+L(S_i)) ≤ C);
      W_j= W_j ∪ S_i;
      continue;
    else
      j++;
return max(j| W_j≠0);
```

표 2. Combine 알고리즘의 pseudo 코드
Table 2. Pseudo code of Combine algorithm

```
Combine() // Combine algorithm
A= Σ_{i=1}^{n} S_i/m;    X=LPT();
if X<1.5 · A
  then C_U=X;
  C_L=max(X/(4/3 − 1/3m), S_1, A);
    i= C_L;
    while i≤C_U ∧ FFD(i)>m
      i=i+1;
    return FFD solution
else
  then return LPT solution
```

테스트 패턴을 모드에 따라 입력받거나 코어 출력 터미널로부터 응답을 받아 저장한다. WIR (Wrapper Instruction Register)은 wrapper instruction을 입력받아 wrapper control 신호에 따라 코어의 스캔 테스트 여부와 WBR의 모드를 선택하며, WBYR (Wrapper BYpass Register)은 serial wrapper input을 다른 wrapper로 입력하기 위해 활용된다.

2.2 Wrapper 설계

일반적으로 wrapper 설계는 코어의 테스트 시간과 사용 TAM 와이어 수의 최소화를 목표로 하며 입력과 출력 셀 및 내부 스캔 체인의 수 정보를 바탕으로 진행된다. 입력 셀과 출력 셀도 테스트가 필요하므로 길이가 1인 스캔 체인과 동일하게 테스트된다. wrapper 설계 알고리즘들을 활용하여 주어진 TAM 와이어에 연결되는 내부 스캔 체인들이 결정되고 입력 셀들과 출력 셀들에 대해 반복된다.[1-4] TAM 와이어에 연결된 입력 셀과 스캔 체인, 출력 셀들은 순차적으로 테스트 된다. 그림 3은 3개의 입력 터미널과 1개의 출력 터미널, 그리고 각각의 길이가 9, 6, 5인 내부 스캔 체인 3개를 갖는 코어에 2개의 TAM 와이어를 연결하여 구성된 wrapper의 예를 보인다.[15]

TAM 와이어 #1은 9개의 플립플롭으로 구성된 내부 스캔 체인으로 구성되고, 입력 터미널과의 연결을 위한 wrapper 셀 2개와 출력 터미널과의 연결을 위한 wrapper 셀 1개와 연결되어 총 12개의 플립플롭으로 연결된다. TAM 와이어 #2는 6개와 5개의 플립플롭들로 구성된 내부 스캔 체인을 포함하고 입력 터미널과의 연결을 위한 wrapper 셀 1개를 거치므로 총 12개의 플립플롭으로 연결된다. TAM 와이어 #1, #2는 별도의 TAM 와이어이므로 코어 테스트는 병렬로 이루어진다. Wrapper의 테스트 시간 T는 식 (1)을 사용하여 계산된다.[3] 입력 스캔 체인 S_{in}은 TAM 와이어에 연결된 입력 셀과 스캔 체인의 길이, 출력 스캔 체인 S_{out}은 TAM 와이어에 연결된 스캔 체인과 출력 셀의 길이, P는 코어의 테스트 패턴 수이다. 식 (1)로부터 입력 스캔 체인의 길이와 출력 스캔 체인의 길이 중 최대값에 비례하여 테스트 시간이 결정되는 것을 알 수 있다.

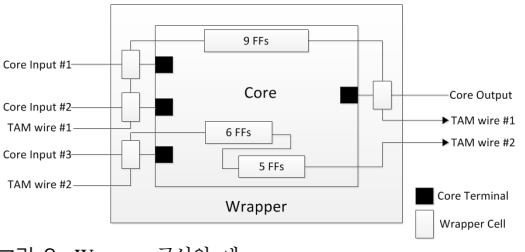

그림 3. Wrapper 구성의 예
Fig. 3. Example of wrapper configuration

$$T=\{1+\max(S_{in}, S_{out})\}P+\min(S_{in}, S_{out}) \quad (1)$$

LPT 알고리즘은 y개의 스캔 체인을 m개의 TAM 와이어에 균등하게 partition한다. 모든 스캔 체인은 길이의 내림차순으로 정렬되고, m개의 스캔 체인이 각 TAM 와이어에 한 개씩 연결된다. 나머지 스캔 체인은 연결된 스캔 체인 길이의 합이 가장 적은 TAM 와이어에 추가로 연결된다.[1] 주어진 TAM 와이어를 모두 사용하여 테스트 시간이 최소화된다. FFD 알고리즘은 스캔 체인 길이의 제한조건 C를 입력받고 TAM 와이어에 연결된 스캔 체인 길이의 합이 C를 초과하지 않는 경우 스캔 체인을 순차적으로 partition하는 알고리즘이며 사용되는 TAM 와이어의 수가 최소화된다.[3] FFD 알고리즘과 유사하지만 더 정교한 BFD 알고리즘은 FFD 알고리즘에서 스캔 체인이 연결될 때마다 TAM 와이어를 partition된 스캔 체인 길이의 합에 대해 내림차순 정렬하는 과정이 추가된 알고리즘이다.[3-4] 표 1은 LPT와 FFD 알고리즘의 pseudo 코드를 나타낸다.

최근까지 대표적으로 사용되고 있는 Combine 알고리즘은 LPT와 FFD 알고리즘을 결합하여 사용한다. 주어진 TAM 와이어 수에 이상적으로 partition된 스캔 체인의 길이를 A라 하면, LPT 알고리즘의 결과가 A의 1.5배 이상인 경우 그 결과로 wrapper가 설계된다. 그 외의 경우 FFD 알고리즘의 제한조건을 하한치 C_L부터 상한치 C_H까지 증가시키면서 반복 수행하여 사용되는 TAM 와이어 수를 줄인다. 사용되는 LPT와 FFD 알고리즘은 TAM 와이어와 스캔 체인들의 구성에 따라 다른 결과를 보이므로 combine 알고리즘은 단일 알고리즘 사용에 비해 최적화된 결과를 얻는다. 이 알고리즘의 pseudo 코드는 표 2와 같다.[3]

Combine 알고리즘을 P93791 ITC'02 벤치마크 회로의 코어 6에 사용하면 TAM 와이어 수와 테스트 시간 사이의 관계가 그림 4와 같이 나타난다.[13,15] 가로축은 가용 TAM 와이어 수, 세로축은 테스트 시간이다. 사용 가능한 TAM 와이어의 수를 $\#TAM_{Available\ j}$, 테스트 시간을 T_j라 할 때, $\#TAM_{Available\ i} \leq \#TAM_{Available\ j}$, $T_i \leq T_j$를 만족하는 해 ($\#TAM_{Available\ i}$, T_i)가 존재하지 않는 경우의 ($\#TAM_{Available\ j}$, T_j)를 pareto 최적 점으로 정한다.[16] 이때의 $\#TAM_{Available\ j}$는 사용 TAM 와이어 수인 $\#TAM_{Utilized\ j}$가 되고 pareto 최적 점은 코어의 해 집합이 된다. 멀티코어 테스트 스케줄러는 코어 6의 해

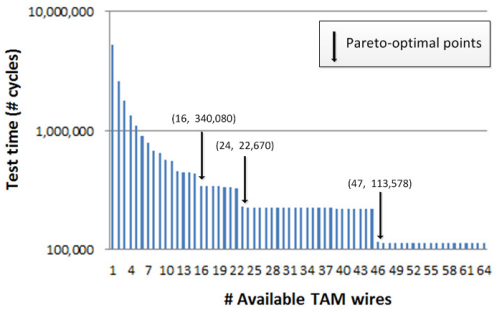

그림 4. P93791 ITC'02 벤치마크 회로 코어 6의 테스트 시간과 가용 TAM 와이어 수의 관계
Fig. 4. Relation between test time and number of available TAM wires for Core 6 in ITC'02 benchmark circuit P93791

집합 {(16, 340,080), (24, 226,720), (47, 113,578)}을 사용하여 design space exploration을 수행한다.

2.3 멀티코어 SoC의 테스트 스케줄러

테스트 스케줄러는 멀티코어 SoC을 구성하는 각 코어들의 해 집합으로 구성된 wrapper 정보를 기반으로 각 코어들의 테스트를 배열한다. 테스트 시간을 폭으로 TAM 와이어 수를 높이로 변환하는 B*-트리 기반의 평면계획 기법을 사용한 테스트 스케줄러는 다른 방법들보다 테스트 시간이 적다.[6-10,17] 이 방법은 회로를 구성하는 코어들에 대응되는 B*-트리 노드들을 구성한다. 두 노드의 교환과 노드의 다른 위치로의 이동, 각 코어의 해 변환의 세 가지 동작을 동일한 확률로 반복 수행하여 배열하는 design space exploration을 통해 해당 멀티코어 SoC의 테스트 시간을 최소화한다.[17] 이 테스트 스케줄러는 제안된 알고리즘의 멀티코어 SoC에 대한 효용성을 검증하기 위한 기준 솔루션으로 사용되었다. 그림 5는 ITC'02 벤치마크 회로 D695를 B*-트리 기반 평면계획 기법으로 테스트 스케줄링을 수행한 결과를 TAM 와이어 수 m과 테스트 시간 t에 대해 나타낸다.

Ⅲ. 제안하는 알고리즘

본 절에서는 멀티코어 SoC의 테스트 시간을 줄이기 위한 코어 wrapper 설계 알고리즘을 제안한다. $\#TAM_{Available}$은 스케줄러에 사용 가능한 TAM 와이어 수, $\#TAM_{Utilized\ j,k}$는 $\#TAM_{Available}$이 j일 때 코어 k의 사용 TAM 와이어 수, $S_{i,k}$는 코어 k의 i번째 스캔 체인, $L(S_{i,k})$는 $S_{i,k}$의 길이, $\min(L(S_{i,k}))$는 $L(S_{i,k})$의 최소값, $\max(L(W_{j,k}))$는 $\#TAM_{Available}$이

j일 때 코어 k의 각 $TAM_{Utilized\,j,k}$에 할당된 스캔 체인들의 총 길이 중 최대값, $\#pattern_k$는 코어 k의 테스트 패턴 수, $T_{j,k}$는 $\#TAM_{Available}$이 j일 때 $\max(L(W_{j,k}))$와 $\#pattern_k$를 곱한 코어 k의 테스트 시간, $Size_{j,k}$는 $TAM_{Utilized\,j,k}$와 $T_{j,k}$를 곱한 테스트 스케줄링이 수행되는 평면에서의 크기다. 기존의 코어 테스트 wrapper 설계를 위한 Combine 알고리즘은 단일 코어의 $\#TAM_{Available}$에 대한 $T_{j,k}$와 $\#TAM_{Utilized\,j,k}$ 감소에 집중하여 wrapper의 local 해 집합을 구성한다. 여러 코어들의 테스트를 평면에 배열하는 테스트 스케줄러의 design space exploration을 고려하면 global 해 집합이 필요하다.

제안된 알고리즘은 각 코어에 대해 Combine 알고리즘을 사용하여 $\#TAM_{Available}$에 대한 $\#TAM_{Utilized\,j,k}$와 $\max(L(W_{j,k}))$를 구한 후, $\max(L(W_{j,k}))$와 $\#pattern_k$를 곱하여 $T_{j,k}$를 구한다. 코어에 대한 wrapper 정보는 ($\#TAM_{Available}$, $\#TAM_{Utilized\,j,k}$, $\max(L(W_{j,k}))$, $T_{j,k}$)로 local 해 집합은 ($\#TAM_{Utilized\,j,k}$, $T_{j,k}$)로 표현된다. 각 코어들은 $\#TAM_{Available}$에 대한 $T_{j,k}$의 내림차순으로 정렬되고, $\#TAM_{Available}$에 대해 가장 긴 $T_{j,k}$를 가지는 코어가 dominant 코어로 선정된다. dominant 코어의 테스트 시간 $T_{j,dominant}$는 global 해 집합이 구성되는 기준으로 사용된다. 그 후 dominant 코어를 제외한 나머지 코어들의 $\max(L(W_{j,k}))$를 $\min(L(S_{i,k}))$만큼 더하여 combine 알고리즘에 의해 제한된 $T_{j,k}$를 최소한으로 증가시킨다. 증가된 $T_{j,k}$가 dominant 코어의 테스트 시간보다 적으면, 새로운 $\max(L(W_{j,k}))$는 BFD 알고리즘의 제한조건 C로 설정된다. BFD 알고리즘

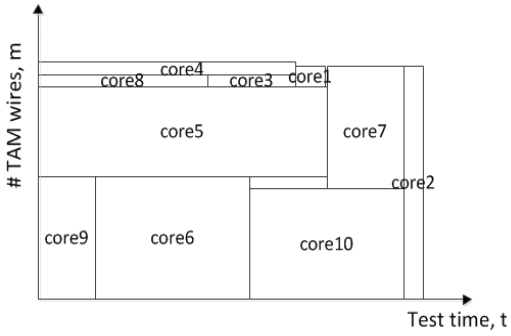

그림 5. D695 ITC'02 벤치마크 회로의 테스트 스케줄링에 대응되는 평면계획도
Fig. 5. Floorplan corresponding to test schedule of the ITC'02 benchmark circuit D695

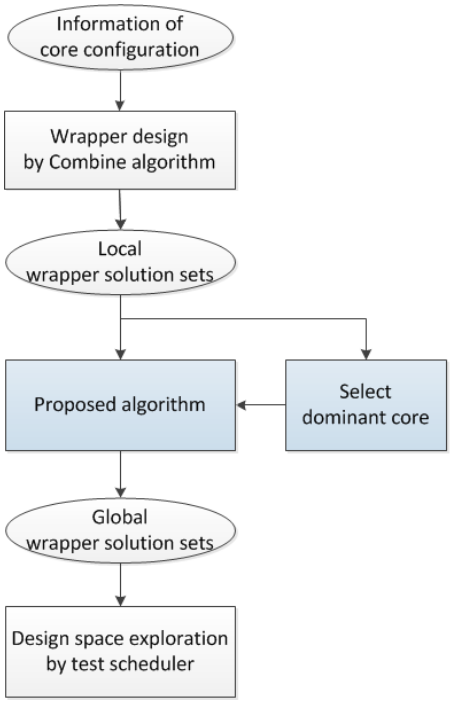

그림 6. 멀티코어 SoC의 테스트 스케줄링 흐름도
Fig. 6. Flowchart of test scheduling multi-core SoC

은 그 값을 입력받아 $\#TAM_{Available}$에 대한 $\#TAM_{Utilized\,j,k}$를 감소시킨다. 변경된 $Size_{j,k}$가 이전의 $Size_{j,k}$보다 작다면 기존의 local 해 집합에 (감소된 $\#TAM_{Utilized\,j,k}$, 증가된 $T_{j,k}$)가 추가된다. 이 과정은 dominant 코어를 제외한 나머지 코어들에 대해 동일하게 진행되며 테스트 스케줄러의 design space exploration을 위해 기존의 local 해 집합에 새로운 해들을 추가하여 global 해 집합으로 확장한다.

그림 6은 제안된 알고리즘이 활용된 멀티코어 SoC의 전체적인 테스트 스케줄링 과정을 개략적으로 표현한 것이다. 코어의 정보를 토대로 Combine 알고리즘을 통해 local 해 집합이 구성된다. dominant 코어의 테스트 시간을 기준으로 각 코어들에 대해 BFD 알고리즘이 조건부로 수행된다. 테스트 스케줄링을 고려한 해가 추가되어 다양해진 local 해 집합은 global 해 집합으로 확장된다. 그 결과 멀티코어 SoC의 테스트 스케줄러의 design space exploration의 효율성이 향상되어 총 테스트 시간이 감소된다.

Ⅳ. 실 험

제안된 알고리즘의 효용성을 검증하기 위해

표 3. ITC'02 벤치마크 회로에 대한 테스트 시간 결과 비교 (# cycles)
Table 3. Comparison of test time for ITC'02 benchmark circuits (# cycles)

ITC'02 Benchmarks	Combine[17] (for different # TAM wires)				Proposed (for different # TAM wires)				Comparison (%)			
	16	32	48	64	16	32	48	64	16	32	48	64
D695	39,094	19,575	13,513	9,786	37,797	18,922	12,881	9,303	-3.4	-3.5	-4.9	-5.2
P22810	434,233	214580	148,096	114,252	418,847	206860	142,396	108,847	-3.7	-3.7	-4.0	-5.0
P34392	926,293	539,133	539,133	539,133	892,744	516,811	516,811	516,811	-3.8	-4.3	-4.3	-4.3
P93791	1,764,246	881,191	603,484	447,723	1,690,990	834,392	572,117	421,422	-4.3	-5.6	-5.5	-6.2
U226	13,200	8,003	5,280	5,280	12,882	7,692	5,064	5,064	-2.5	-4.1	-4.3	-4.3
A586710	322,199,22	169,556,10	126,829,78	947,644,7	30,124,287	15,866,452	11,736,009	8,739,081	-7.0	-6.9	-8.1	-8.4
G1023	29,467	15,149	14,646	14,646	28,596	14,477	14,036	14,036	-3.0	-4.6	-4.3	-4.3
F2126	346,530	331,981	331,981	331,981	329,092	318,962	318962	318,962	-5.3	-4.1	-4.1	-4.1
D281	7,031	3,887	3,887	3,887	6,827	3,759	3,759	3,759	-3.0	-3.4	-3.4	-3.4
T512505	10,398,980	5,202,826	5,176,136	5,176,136	9,876,860	4,887,389	4,840,971	4,840,929	-5.3	-6.5	-6.9	-6.9

ITC'02 벤치마크 회로를 대상으로 총 테스트 시간을 측정하였다.[14] 제안된 방법은 C++로 구현하여 SUN-Sparc 워크 스테이션에서 수행되었다. 기존의 방법은 각 코어에 대해 Combine 알고리즘을 사용한 후 해 집합을 구성하는데 반하여[17], 제안된 알고리즘은 기존의 방법을 사용해 구한 해 집합을 확장한다. 구성된 해 집합을 입력으로 B*-트리 기반의 평면계획을 사용한 테스트 스케줄링을 수행한다. 표 3은 ITC'02 벤치마크 회로[14]마다 가용 TAM 와이어 수를 변화시키면서 기존 Combine 알고리즘의 해 집합과 제안된 알고리즘의 해 집합을 테스트 스케줄링 수행 후 테스트 시간과 감소율을 나타낸 것이다. 테스트 시간은 회로에 따라 2.5%~8.4% 감소하였고, 평균적으로 4.7% 감소하였다.

V. 결 론

코어 테스트 wrapper 설계는 코어 테스트 시 사용되는 TAM 와이어 수와 테스트 시간을 고려한다. 본 논문에서는 멀티코어 SoC의 테스트 스케줄링 측면에서 테스트 시간 감소를 위한 wrapper 설계 방법을 제안하였다. 기존 Combine 알고리즘을 사용한 단일 코어의 local 해 집합의 한계를 극복하기 위해, 제안된 알고리즘은 dominant 코어를 기준으로 나머지 코어의 테스트 시간을 증가시킨 후 BFD 알고리즘을 추가로 사용한다. 테스트의 크기가 감소하는 경우 (사용 TAM 와이어 수, 테스트 시간)을 테스트 스케줄러에서 사용되는 해 집합에 추가하여 global 해 집합이 구성된다. ITC'02 벤치마크 회로를 대상으로 실험한 결과 평균 4.7%의 전체 테스트 시간이 감소하였다. 추후에는 각 코어별 테스트 wrapper 설계 과정의 시간 오버헤드를 감소시키면서 멀티코어 SoC의 테스트 시간을 최소화하는 방법에 대한 추가 연구가 필요하다.

References

[1] R. Graham, "Bounds on multiprocessing timing anomalies," *SIAM J. Applied Mathematics*, vol. 17, no. 2, pp. 416-429, Mar. 1969.

[2] E. Coffman, M. Garey, and D. Johnson, "An application of bin-packing to multiprocessor scheduling," *SIAM J. Computing,* vol. 7, no. 1, pp. 1-17, Feb. 1978.

[3] E. Marinissen, S. Goel, and M. Lousberg, "Wrapper design for embedded core test," in *Proc. IEEE Int'l Test Conf.*, Atlantic City, NJ, pp. 911-920, Oct. 2000.

[4] V. Iyengar, K. Chakrabarty, and E. Marinissen, "Test wrapper and test access mechanism co-optimization for system on chip," in *Proc. IEEE Int'l Test Conf.*, Baltimore, MD, pp. 1023-1032, Oct. 2001.

[5] Y. Chang, Y. Chang, G. Wu, and S. Wu, "B*-Trees: A new representation for non-slicing floorplans," in *Proc. Design Automation Conf.*, Los Angeles, CA, pp. 458-463, Jun. 2000.

[6] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle-packing based module placement," in *Proc. Int'l Conf. Computer-Aided Design,* San Jose, CA, pp. 472-479, Nov. 1995.

[7] S. Nakatake, K. Fujiyoshi, H. Murata, and Y.

Kajitani, “Module placement on BSG-structure and IC layout application,” in *Proc. Int’l Conf. Computer-Aided Design,* San Jose, CA, pp. 484-491, Nov. 1996.

[8] C. Su and C. Wu, “A graph-based approach to power-constrained SoC test scheduling,” *J. Electronic Testing,* vol. 20, no. 1, pp. 45-60, Feb. 2004.

[9] Y. Huang, S. Reddy, W. Cheng, P. Reuter, N. Mukherjee, C. Tasi, O. Samman, and Y. Zaidan, “Optimal core wrapper width selection and SoC test scheduling based on 3-D bin packing algorithm,” in *Proc. IEEE Int’l Test Conf.*, Baltimore, MD, pp. 74-82, Oct. 2002.

[10] K. Chakrabarty, “Test scheduling for core-based system using mixed-integer linear programming,” *IEEE Trans. Computer-Aided Design*, vol. 19, no. 10, pp. 1163-1174, Oct. 2000.

[11] V. Immaneni and S. Raman, “Direct access test scheme – Design of block and core cells for embedded ASICs,” in *Proc. IEEE Int’l Test Conf.*, Washington, DC, pp. 488-492, Sept. 1990.

[12] E. Marinissen, Y. Zorian, R. Kapur, T. Taylor, and L. Whetsel, “Toward a standard for embedded core test: An example,” in *Proc. IEEE Int’l Test Conf.*, Atlantic City, NJ, pp. 616-627, Sept. 1999.

[13] Y. Zorian, E. Marinissen, and S. Dey, “Testing embedded core-based system chips,” in *Proc. IEEE Int’l Test Conf.*, Washington, DC, pp. 130-143, Oct. 1998.

[14] E. Marinissen, V. Iyengar, and K. Chakrabarty, *ITC’02 SoC Test Benchmarks* (2002), Retrieved Sep., 10, 2013, from http://itc02socbenchm.pratt.duke.edu/#Benchmarks.

[15] J. Jung, “Efficient test wrapper design in SoC,” *J. Academia-Industrial Technology*, vol. 10, no. 6, pp. 1191-1195, Jun. 2009.

[16] V. Iyengar, K. Chakrabarty, and E. Marinissen, “On using rectangle packing for SoC wrapper/TAM co-optimization,” in *Proc. VLSI TEST Symp.*, Monterey, CA, pp. 253-258, Apr. 2002.

[17] J. Wuu, T. Chen, and Y. Chang, “SoC test scheduling using the B*-tree based floorplanning technique,” in *Proc. Asia South Pacific Design Automatic Conf.*, Shanghai, China, pp. 1188-1191, Jan. 2005.

강 우 진 (Woo-jin Kang)

2012년 2월 : 서강대학교 전자공학과 졸업

2012년~현재 : 서강대학교 전자공학과 CAD & ES 연구실 석사과정

<관심분야> Embedded System Design, ASIP Design

황 선 영 (Sun-young Hwang)

1976년 2월 : 서울대학교 전자공학과 학사

1978년 2월 : 한국과학원 전기 및 전자공학과 공학석사

1986년 10월 : 미국 Stanford 대학 전자공학 박사

1976~1981년 : 삼성반도체(주) 연구원, 팀장

1986~1989년 : Stanford 대학 Center for Integrated System 연구소 책임연구원 및 Fairchild Semiconductor Palo Alto Research Center 기술 자문

1989~1992년 : 삼성전자(주) 반도체 기술 자문

1989년 3월~현재 : 서강대학교 전자공학과 교수

<관심분야> SoC 설계 및 framework 구성, CAD 시스템, Embedded System 설계, Computer Architecture 및 DSP System Design 등