Journal of Digital Contents Society Vol. 9 No. 3 Sept. 2008(pp. 471-481)

시스템 온 칩(system-on-a-chip) 내부 코어들의 전력소모 변화를 고려한 새로운 테스트 스케쥴링 알고리듬 설계

이재민*, 이호진**, 박진성***

요 약

전력소모를 고려한 테스트 스케줄링은 회로의 복잡도가 높은 SoC 시스템을 테스트할 경우 제한된 전력 소모량 내에서 고장 검출율을 높일 수 있고 테스트 시간을 단축 할 수 있는 효과적인 방법이다. 본 논문에서는 제한된 전력소모량 내에서 효율적으로 테스트를 수행하기 위한 테스트 자원의 모델링 방법 및 테스트 스케줄링 알고리듬을 제안하고 그 유효성을 검증한다. 테스트 자원의 모델링 방법으로는 전력사용량의 최고점과 차고점을 이용한 방법 및 소모 전력의 변화량에 따라 테스트 자원을 분할하는 방법을 제시한다. 또한 테스트 자원과 코어의 상관관계를 이용하여 동시 사용가능한 최대 코어 수를 생성하는 확장나무성장 그래프 생성 알고리듬 및 전력의 최적화가 가능한 전력 소모량 변이 그래프 생성 알고리듬으로 구성된 휴리스틱(heuristic) 테스트 스케줄링 알고리듬을 제안하고 이전의 알고리듬과 비교한다.

A Novel Test Scheduling Algorithm Considering Variations of Power Consumption in Embedded Cores of SoCs

Jae-Min Lee*, Ho-Jin Lee**, Jin-Sung Park***

Abstract

Test scheduling considering power dissipation is an effective technique to reduce the testing time of complex SoCs and to enhance fault coverage under limitation of allowed maximum po wer dissipation. In this paper, a modeling technique of test resources and a test scheduling al gorithm for efficient test procedures are proposed and confirmed. For test resources modeling, two methods are described. One is to use the maximum point and next maximum point of po wer dissipation in test resources, the other one is to model test resources by partitioning of th em. A novel heuristic test scheduling algorithm, using the extended-tree-growing-graph for gen eration of maximum embedded cores usable simultaneously by using relations between test res ources and cores and power-dissipation-changing-graph for power optimization, is presented an d compared with conventional algorithms to verify its efficiency.

Keywords : System-on-a-chip, Test scheduling, Test resources modeling, Conflicts of power dissipation, Variation of power dissipation

1. 서론

최근 고기능 소형화의 소비자 욕구와 반도체 공정 기술의 발전으로 인하여 기 개발된 IP를 사용하여 신규 시스템을 One Chip화하는 SoC (System-on-Chips) 기술이 등장하게 되었다[1][2][3][4][5][6].

SoC는 기 개발된 IP, 예를 들어서 CPU, SRAM, DSP, 아날로그 회로 또는 User-Defined -Lo

※제일저자(First Author): 이재민
접수일자:2008년07월30일, 심사완료:2008년09월05일
* 관동대학교 전자정보통신공학부
leejm@kd.ac.kr
** 관동대학교 전자정보통신공학부
*** (주)라이벌코리아
▣ 이 논문은 2006년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음(KRF-2006-521-D00341).

gics(UDLs)등으로 조합하여 구성하게 된다. 기 개발된 IP의 사용으로 신규 시스템 개발에 따른 Time-to-Market 기간은 짧아지고 성능향상, 크기감소, 생산비용 감소 등의 다양한 효과를 얻을 수 있다. 그러나 SoC 기술은 성능향상에 따른 복잡도의 증가로 인하여 단일 코어로 구성되어 있는 시스템의 테스트에 비하여 다양하고 복잡한 테스트 방법을 요구한다. 이러한 문제점은 개발 및 양산 과정에서 테스트시간 증가로 인한 비용 증가로 이어진다. 또한 일반적인 테스트보다 복합적인 테스트 패턴으로 종합적인 테스트가 이루어지기 때문에 통상적인 동작상태보다 많은 전력을 소모하게 되고 IR-drop현상으로 인하여 정상 상태임에도 불구하고 불량으로 판명되는 경우가 발생한다[2][3][4].

테스트 시간의 감소를 위하여 SoC 내에 내장된 각 코어를 병렬적으로 테스트할 필요가 있다. 동 시간에 테스트하는 코어의 수를 늘리면 소비전력의 문제점이 더욱 커질 수 있다. 소비전력의 증가는 내부 코어의 과열을 야기하고 이로 인해 시스템의 손상이 유발되므로 최근에는 SoC 의 구성 코어들을 테스트하는 과정에서 소비전력을 고려하는 것이 점차 중요해지고 있다[3][4][5][6][7].

따라서 제한된 소비전력 내에서 IP와 함께 제공되는 테스트 패턴을 이용하여 테스트 스케줄링의 기본단위인 테스트 자원의 효과적인 모델링 기술이 요구된다.

또한 SoC 설계 자체에는 하드웨어적으로는 부담을 주지 않으면서 테스트 수행 시간을 감소시켜 테스트 비용을 줄일 수 있는 코어들의 소비 전력을 고려한 테스트 스케줄링 기술의 연구가 병행되어 진행 되고 있다.

본 논문에서는 SoC 테스트를 위하여 테스트 스케줄링 알고리즘의 효율을 높일 수 있는 테스트 자원의 모델링 방법 및 테스트의 수행시간을 최소화하는 휴리스틱 테스트 스케줄링 알고리듬을 제안한다.

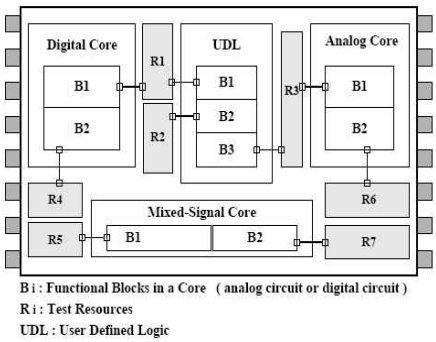
테스트 자원의 병렬 사용 가능성을 높이기 위하여 2가지의 모델링 방법을 제안한다. 첫째는 테스트 자원의 소비전력 사용량의 최고점과 차고점을 이용하여 테스트 자원을 모델링 하는 방법이고, 둘째는 테스트 자원의 소모 전력의 변화량에 의한 테스트 자원의 분할 모델링 방법이다. 제안하는 테스트 스케줄링 시 발생하는 무위 테스트 시간 (Idling Test Time)을 최소화하기 위하여 각 분할된 테스트 자원의 우선순위를 결정하고 확장나무 성장 그래프 (Expanded Tree Growing Graph : ETGG) 와 시간에 따른 전력 소모량 변이 그래프를 이용하여 충돌이 없는 테스트 자원을 시간 진행에 따라서 배치한다.

본 알고리듬은 테스트 시간의 변화에 따른 테스트 자원의 병렬 처리 능력을 높이고 총 전력 사용량 내에서 테스트 시간을 최소화하는 장점을 갖는다.

2. 테스트 스케줄링을 위한 SoC 모델링

일반적인 SoC는 (그림 1)과 같이 다양한 기능적 특성 및 물리적인 특성을 지니는 IP코어들로 구성되어 있다.

각 코어는 특성에 따라서 기능적 블록(B_i)의 조합으로 구성되어 있으며 각 블록은 테스트 벡터에 의한 테스트의 자원(R_i)으로 구성 된다.



(그림 1) SoC 모델의 기능블록도 및 테스트 자원

테스트의 목적 또는 테스트의 방법에 따라서 다양한 가능성이 존재하지만 (그림 1)의 R1 테스트 자원처럼 2개 이상의 기능 블록과 연관될 수 있고, R6 자원과 같이 하나의 테스트 자원이 단일 블록으로만 종속될 수도 있다.

각 IP 코어들은 높은 제품 신뢰도와 요구되는 일정한 고장 검출율을 얻기 위해 기능적인 성능 평가와 코어 및 입출력 단자 사이의 데이터 및 컨트롤 라인과 같은 물리적인 연결 확인을 위하

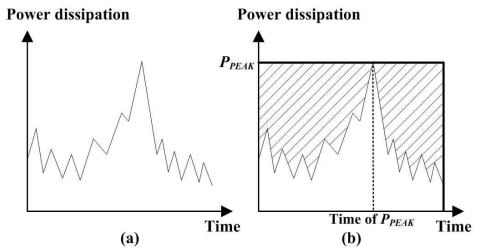
여 다양한 테스트 기법들이 사용된다.

최근에는 테스트 기술의 개발로 인하여 코어 자체적으로 고장을 검출하는 BIST(Built-in self test)기법 및 외부의 ATE(Automatic Test Equipment)를 사용하여 고장을 검출하는 기법 등이 복합적으로 사용되기 때문에 여러 가지의 테스트 기법을 필요로 한다[1][2][3][4][8].

3. 테스트 스케줄링을 위한 테스트 자원 모델링

IP와 함께 제공되는 테스트 벡터를 사용하여 개별 코어(IP)의 테스트를 진행하고 테스트 수행 과정의 전력사용량을 측정하면 (그림 2)(a)와 같은 테스트 소비전력을 측정하여 데이터화 할 수 있다[1-4].

소비전력을 고려한 테스트 자원 그래프는 (그림 2)(b)와 같이 최고전력(P_{PEAK})을 기준으로 테스트 자원을 직사각형의 형태로 모델링 한다.



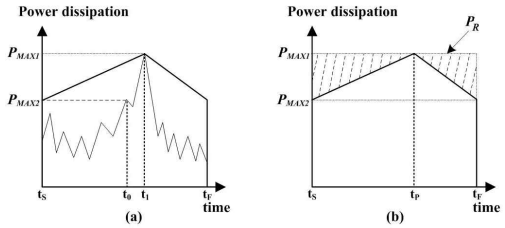
(그림 2) (a) 코어의 소비전력량 변화 그래프,
(b) 일반적 테스트자원의 모델링 방법

전력을 고려한 테스트 자원 모델링의 이전 연구에서는 최고 전력만을 고려함으로써 (그림 2)(b)의 사선 부분에 속하는 리던던트한 공간이 포함되어 있었다. 최적화된 테스트 자원을 모델링하는 방법은 테스트 수행 이전에 각 IP의 테스트 자원을 재구성하는 방법으로서 테스트의 수행시간에 영향을 주지 않으며 알고리듬의 효율을 높이는 효과적인 방법이기 때문에 본 논문에서는 알고리듬의 효율을 높일 수 있는 새로운 2가지의 테스트 자원 모델링 방법을 제안한다.

3.1 최고점과 차고점을 이용한 테스트 자원 모델링

(그림 3)의 (a)와 같이 개별 코어에 대한 테스트를 진행하고 시간별 소비전력을 측정하여 코어의 테스트 수행시간(t_n) 내의 전력의 최고점(P_{MAX1})과 차고점(P_{MAX2})을 조사한다.

테스트 자원을 (그림 3)(b)와 같이 차고점(P_{MAX2})의 전력 값을 테스트 시작 시각(t_S)과 종료 시각(t_F)의 전력값으로 변환하여 모델링한다.



(그림 3) 전력의 최고점과 차고점을 이용한
테스트 자원의 모델링

(정의 1) 코어의 테스트 수행시간(t_n) : 독립된 코어(N)에 대하여 테스트 벡터로 테스트를 진행할 경우 테스트 시작시각(t_S)부터 종료시각(t_F)까지의 총 수행 기간(t_n)을 의미하며 식 (1)과 같다.

$$t_n = t_F - t_S \quad (1)$$

(정의 2) 코어의 테스트 수행시간(t_n) 내의 전력의 최고점(P_{MAX1})과 차고점(P_{MAX2}) : 전력 사용의 최고점을 나타내는 시각을 최고점 P_{MAX1}이라 하고, 전력 사용의 두 번째로 높은 시각을 차고점 P_{MAX2}라 한다.

(정의 3) 전력 최고점의 시각(t_P) : 식 (2)를 만족하며 소비전력이 최고점(P_{MAX1})이 되는 시각을 전력 최고점의 시각이라 한다.

$$t_S \le t_P \le t_F \quad (2)$$

(정의 4) 리던던트(Redundant)한 공간의 누적 전력량(P_R) : 기존의 연구에서 사용된 직사각형 형태의 테스트 자원 모델링과 비교하여 감소될 수 있는 모델링 자원 전력의 총 합을 리던던트한 공간의 누적 전력량 P_R라 하고 식 (3)과 같이 나타낸다.

$$P_R = \frac{1}{2}(P_{MAX1} - P_{MAX2})t_n \qquad (3)$$

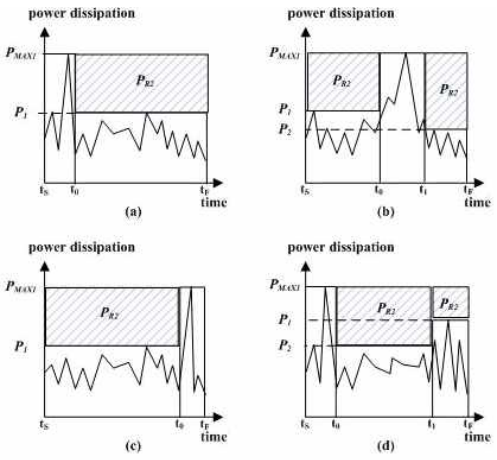
테스트 자원의 모델링을 위해 (그림 3)(b)와 같이 최고점과 차고점을 이용하여 테스트 자원을 5각형의 형태로 나타낸다. 최고 전력사용량만을 고려한 (그림 2)(b)와 같은 종래의 테스트 자원 모델링 방법과 비교하였을 경우, (그림 3)(b)의 사선 만큼의 공간감소를 얻는다.

이러한 공간감소 효과는 식 (3)과 같이 최대 P_R 만큼의 전력소비량이 리던던트(redundant)되어 모델링 될 수 있기 때문에 테스트 자원의 병렬 처리 능력이 향상된다[9].

3.2 소모 전력 변화량에 의한 테스트 자원 분할 모델링

소모 전력의 변화량에 의한 테스트 자원의 분할 모델링 방법은 (그림 4)와 같이 테스트의 자원을 병렬 처리 능력이 향상될 수 있도록 2개 이상의 테스트 자원으로 분할하는 것이다[10].

테스트 자원을 분할하는 시각은 테스트 수행 과정에서 측정되는 소비전력의 변화량이 급격하게 상승되었다가 하강하는 시간을 확인하여 결정한다. 주로 최고점 및 차고점이 결정의 주요한 요소가 된다. 단, 테스트 자원을 기준 수 이상의 다수개로 분할하는 경우에는 테스트 수행시간은 감소될 수 있으나 테스트 스케줄링 알고리듬의 연산과정에서 과도한 연산시간으로 부담을 줄 수 있다. 따라서 본 제안에서는 테스트 자원의 분할 기준 수를 3개로 정하여 모델링한다.



(그림 4) 2분할 또는 3분할 모델링 방법

(그림 4)(a)와 (그림 4)(c)는 테스트 초기의 전력사용량이 높고 나머지 부분의 변화량은 크지 않은 경우(H-L: High-Low Type) 또는 테스트 후기의 특정 부분만 소비전력 사용량이 높고 나머지 부분의 변화량이 크지 않은 경우(L-H: Low -High Type)이며 2 분할한다.

(그림 4)(b)와 같이 한부분의 특정 부분만 소비전력이 높은 경우(L-H-L: Low-High-Low Type)는 3분할하며, (그림 4)(d)와 같이 시작 및 종료 부분의 소비전력이 높고 한부분의 특정 부분만 소비전력이 낮은 경우도 나머지 중간 부분의 전력 값이 일정한 경우(H-L-H: High-Low -High Type)도 3분할한다. 단, 전체적으로 전력 값의 변화가 크지 않은 경우에는 분할하지 않는다.

(그림 2)(b)와 같이 최대 전력사용량만을 고려한 직사각형의 테스트 자원 모델링 방법과 비교하면 본 테스트 모델링 제안은 (그림 4)에서 사선으로 표시된 P_{R2}를 제거하여 병렬사용 능력을 높일 수 있으며 식 (4)와 같이 최대 P_{R2}만큼의 전력소비량이 감소된 테스트 자원으로 모델링 할 수 있다.

$$P_{R2} = (P_{MAX1} - P_1)(t_0 - t_S) + (P_{MAX1} - P_2)(t_F - t_1) \qquad (4)$$

(그림 4)(a) H-L 2분할 모델링의 P_{R2} 적용 수식, (그림 4)(b) L-H-L 3분할 테스트 모델링의 P_{R2} 적용 수식, (그림 4)(c) L-H 2분할 테스트 모델링의 P_{R2} 적용 수식, 그리고 그림 4(d) H-L-H 3분할 테스트 모델링의 P_{R2} 적용 수식은 식 (5), 식 (6), 식 (7) 그리고 식 (8)과 같이 다르게 적용된다.

$$P_{R2H-L} = (P_{MAX1} - P_1)(t_F - t_0) \qquad (5)$$

$$P_{R2L-H-L} = (P_{MAX1} - P_1)(t_0 - t_S) + (P_{MAX1} - P_2)(t_F - t_1) \qquad (6)$$

$$P_{R2L-H} = (P_{MAX1} - P_1)(t_0 - t_S) \qquad (7)$$

$$P_{R2H-L-H} = (P_{MAX1} - P_1)(t_1 - t_0) + (P_{MAX1} - P_2)(t_F - t_1) \qquad (8)$$

본 모델링 방법은 테스트 자원의 분할 시점인 t_0 및 t_1 의 시각에서 테스트 입력 패턴 및 출력 패턴 데이터를 저장하는 기능이 하드웨어적으로 구현 가능해야 하며, 테스트 스케줄링 과정에서

순차적인 테스트 진행이 보장되어야 한다야 한다[2][9].

4. 테스트 스케줄링 알고리듬

테스트 스케줄링의 기본 목적은 모든 제한 조건들(constraints)을 만족하면서 테스트 자원들이 서로 충돌하지 않도록 테스트 자원들을 병렬로 배치하여 총 테스트 시간이 최소가 되게 하는 것이다. 테스트 시간을 최소화하기 위하여 외부 테스트로 인한 테스트 자원과 BIST 엔진의 테스트 자원 등이 여러 코어들에게 적절히 배분되어 병렬 수행 수를 늘리기 위한 코어의 테스트들은 최적으로 스케줄 되어야 한다.

안정된 회로 동작을 위하여 전력 소비 또한 반드시 고려되어야만 한다. SoC는 테스트 모드에서 일반 동작 모드일 경우에 사용되어지는 전력보다 다수의 코어가 테스트를 수행함으로써 많은 전력이 사용된다. 따라서 코어 내의 스위칭 동작이 일반모드보다 테스트 모드일 경우 더욱 많아서 코어의 손상을 줄 수 있다.

알고리듬을 설명하기 위한 용어를 정의 한다.

(정의 5) 테스트 자원의 충돌(Conflict of test resources) : 동일 테스트 데이터라인을 사용하거나, 코어에 종속되어 있는 2개 이상의 테스트 자원들이 동시에 테스트를 수행하고자 하는 경우는 병렬 처리가 불가능하며 이것을 테스트 자원 충돌이라 한다.

(정의 6) 테스트 전력 충돌(Conflict of testing power dissipation) : 서로 다른 코어의 사용으로 자원의 충돌이 발생하지 않는 테스트 자원의 그룹이라 할지라도 각 자원들의 총 전력 사용량이 시간 변화에 따라서 제한된 전력사용량 최고점을 초과하면 병렬 처리가 불가능하며 이것을 테스트 전력 충돌이라 한다.

제안하는 휴리스틱 테스트 스케줄링은 제안된 2가지 방법의 테스트 자원의 모델을 이용하여 테스트 자원의 충돌 및 테스트 전력 충돌이 없고 무위 테스트 시간이 최소화된 테스트 스케줄링을 생성하는 알고리듬으로 최소화된 총 테스트 시간을 얻는 장점을 가진다.

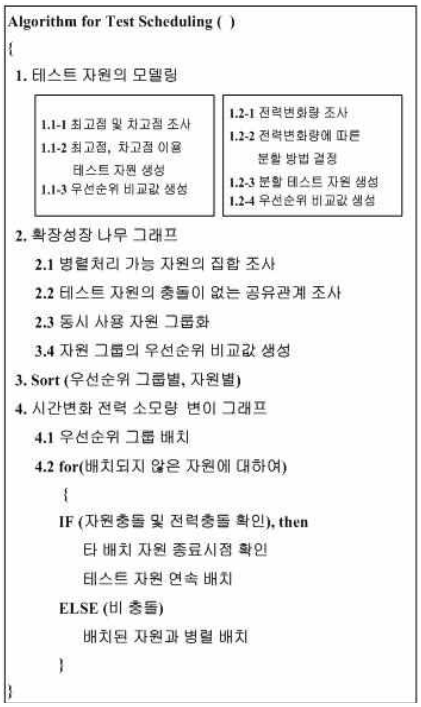
4.1 최고점과 차고점을 이용한 테스트 자원 모델링을 적용한 테스트 스케줄링 알고리듬

제안하는 알고리듬은 (그림 5)와 같은 순서로 진행된다.

첫째, 확장나무성장 그래프를 이용하여 병렬처리가 가능한 테스트 자원의 집합을 조사하고 테스트 자원의 충돌 없는 공유관계를 찾아내어 동시 사용가능한 자원을 그룹화한다.

둘째, 각 자원의 전력 및 시간의 곱을 이용한 상대적 크기를 기준으로 우선순위를 결정한다.

셋째, 우선순위에 따라서 자원을 그룹화 하여 그래프에 배치하며 전력소모량 변이 그래프를 이용하여 테스트 전력 충돌을 확인한다.



(그림 5) 제안하는 휴리스틱 테스트 스케줄링 알고리듬

마지막으로 제한된 소비전력을 초과하면 다른 테스트 자원을 대치하는 과정을 반복 하면서 최적화된 테스트 스케줄링을 수행한다.

4.2 확장나무성장 그래프

확장나무성장 그래프는 (그림 6)과 같이 노드(NODE), 연결라인(LINE)과 레벨(LEVEL)로 구성되어 있다. 각 노드는 코어와 자원에 따른 요

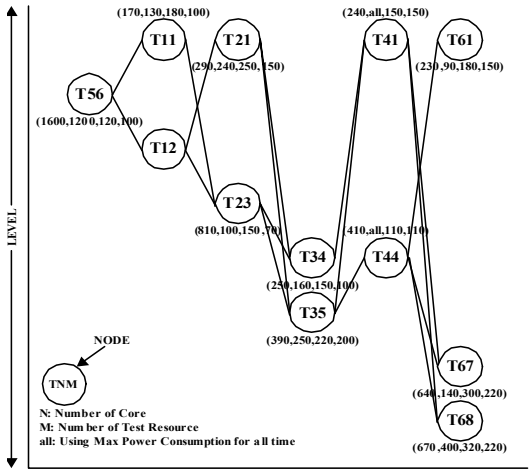
소를 포함하고 있으며 테스트 자원의 총 테스트 시간 t_n, 전력사용량의 최고점일 경우의 테스트 시각 t_p, 최고점의 전력 사용량 P_{MAX1}, 시작과 종료 시각의 전력사용량 P_{MAX2} 즉, 차고점의 전력사용량을 포함한다. 또한 노드는 원형의 TNM으로 표시되는데 여기서 T는 노드, N은 코어의 번호, 그리고 M은 자원의 번호를 의미한다. 테스트 자원의 요소를 표시하는 경우 최고점의 전력소모량 P_{MAX1}과 차고점의 전력소모량 P_{MAX2} 값이 일치하는 경우는 최고점의 시각 t_p를 all로 표시한다.

각 노드를 연결하는 연결라인은 코어를 기준으로 한 병렬처리의 가능성을 나타낸다. 또한 각 레벨은 사용되는 코어 기준의 병렬처리 불가능성을 나타낸다. 따라서 연결라인으로 연결되지 않았거나 같은 레벨의 노드는 그룹화 할 수 없다. 그룹화는 병렬처리 자원의 집합을 의미한다.

(그림 6)의 확장나무성장 그래프는 <표 1>의 예제회로인 가상 SoC-X 회로의 요소 자원을 입력하여 나타난 것이다. <표 1>에서와 같이 SoC-X는 6개의 개별 코어로 구성되어 있다. 각 코어는 Cn으로 표시한다. 각 코어 Cn은 1개 이상의 테스트 자원과 종속되어 있다.

각 노드는 LEVEL 별로 서로 다른 위치에 놓이게 된다. LEVEL은 코어에서 사용되는 동일 테스트 자원을 나타낸다. 각 노드는 실선의 라인으로 연결되어 있다. 각 라인은 상기에서 설명한 것과 같이 각 노드의 병렬 처리 가능성을 나타낸 것이다.

확장 나무 성장 그래프를 이용하면 <표 1>을 (그림 6)과 같이 표시할 수 있으며 총 12개의 노드를 가지고 있다. 12개의 노드를 가진 예제회로 SoC 는 각 노드를 원소로 하는 총 $2^{12}-1$개의 독립집합으로 구성된다. 그러나 각 코어는 종속적인 1개 이상의 테스트 자원과의 상관관계인 테스트 자원의 충돌로 인하여 (그림 6)과 같이 한정된 라인으로 연결될 수 있다. 예를 들어, 가장 작은 단일 집합인 $\{T11\}, \{T12\}, \cdots, \{T68\}$의 12개의 단일 조합으로부터 최대 집합인 $\{T56, T11, T23, T35, T44, T67\}$ 또는 $\{T56, T11, T23, T35, T44, T68\}$로 그룹화 될 수 있다. 확장 나무 성장 그래프는 노드 및 연결 라인을 이용하여 동시 사용 가능한 최대 집합 그룹을 생성 한다.



(그림 6) 확장나무 성장 그래프

<표 1> 예제 SoC-X 회로의 테스트 자원 요소 데이터

Cores (N)	Resource Number (M)	t_n / t_p	P_{MAX1} / P_{MAX2}
C1	1	170/130	180/100
	2	1530/950	210/160
C2	1	290/240	250/150
	3	810/100	150/70
C3	4	250/160	150/100
	5	390/250	220/200
C4	1	240/ALL	150/150
	4	410/ALL	110/110
C5	6	1600/1200	120/100
C6	1	230/90	180/150
	7	640/140	300/220
	8	670/400	320/220

4.3 전력 소모량 변이 그래프

모델링된 테스트 자원은 확장나무 성장 그래프를 이용하여 병렬 사용가능 테스트 자원의 집합으로 그룹화 된다. 시간변화 전력 소모량 그래프 상에 테스트 자원의 배치를 위하여 각 테스트 자원 및 테스트 집합은 우선순위 결정 비교값(SUR)을 생성한다. 각 SUR 값은 테스트 자원 모델링 과정에서 생성하며 테스트 자원의 그룹의 SUR 값은 확장 나무 성장 그래프를 생성하여 그룹화 된 이후 생성한다.

(정의 7) 우선순위 결정 비교 값(SUR: Space of Using Resource) : 테스트 자원의 시간진행에 따른 전력사용량의 누적 값이며, 테스트 자원 모델 n에 관한 SUR_n값의 일반식은 식 (9)와 같다.

$$SUR_n = P_{MAX1} \times t_F - \frac{1}{2}(P_{MAX1} - P_{MAX2})t_F \quad (9)$$
$$= \frac{1}{2}(P_{MAX1} + P_{MAX2})t_F$$

테스트의 자원집합 그룹은 식 (10)과 같이 집합 내의 테스트 자원의 우선순위 결정 비교 값(SUR)의 합이며 SUR_T로 나타낸다. m은 테스트의 자원집합 그룹 내의 테스트 자원의 수이다.

$$SUR_T = \sum_{n=1}^{m} SUR_n \quad (10)$$

우선순위 결정 비교 값에 따라서 각 그룹은 전력 소모량 변이 그래프의 배치 우선 순서가 결정되며, 그룹 내에서도 테스트 자원의 우선순위 결정 비교 값에 따라서 테스트 자원의 배치 우선 순서가 결정된다.

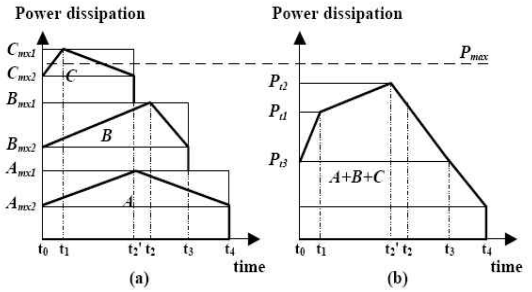
우선순위가 높은 자원은 시간변화 전력 소모량 그래프에서 x, y 축의 0(zero) point에 가깝게 배치된다.

우선순위에 따라서 테스트 자원 그룹을 배치할 때 테스트의 전력 충돌 여부를 확인하고 배치한다. 테스트 자원 N 은 테스트 종료시각 (t_F)과 테스트 자원의 최고 전력사용량 ($N_{MX1}: N = A, B, C$)으로 모델링 된 형태이며 (그림 7)(a)는 직사각형의 테스트 자원 N 을 이용하여 자원의 충돌이 없는 테스트 자원 그룹을 배치한 경우를 나타낸 것이다.

(그림 7)(a)와 같이 테스트 자원 A, B, C로 구성된 테스트 자원그룹 K(K={A, B, C})는 확장 나무 성장 그래프를 통해서 하나의 병렬 사용 그룹으로 그룹화 되었다고 가정한다. 테스트 자원그룹 K는 우선순위 결정 비교 값 SUR에 따라 우선순위가 높은 순서인 A, B, C 순서로 0(zero) point에 가깝도록 배치한다.

그러나 배치된 결과 전체 전력소모량이 시스템에서 제한하는 $P_{\max}$ 를 초과한다. 제한 전력을 초과하기 때문에 A, B, C 의 테스트 자원은 t_0 시각에서 동시에 테스트 될 수 없고, 우선순위가 낮은 테스트 자원 C 는 A 또는 B 테스트 자원이 종료된 이후 배치되어야 한다. 따라서 (그림 7)(a)에 표시된 테스트 자원그룹 K는 최종 테스트 종료 시각이 t4 시각을 초과하게 된다.

반면에 (그림 7)(b)와 같이 제안하는 최고점 및 차고점을 이용한 테스트 자원 모델링 방법을 이용하여 테스트 자원을 모델링하여 배치한다면 배치된 자원의 소비전력 값을 누적할 경우에도 전력 소비량 변이 그래프에 배치된 자원들은 제한하는 $P_{\max}$ 내에서 종료 테스트 시간이 t4 시각이 됨을 알 수 있다.



(그림 7) 테스트 자원 모델의 전력 소모량 변이 그래프

$$P_{tn} = \sum_{N=A}^{m} N + \sum_{N=A}^{m} \left(\frac{N_{mx1} - N_{mx2}}{t_{N1st} - t_{NS}}\right)t_n \quad (11)$$

P_{tn} : t_n 시각의 총 전력누적 값
N : 테스트 자원의 명칭
N_{mx1} : 테스트 자원 N의 최고점 전력사용량
N_{mx2} : 테스트 자원 N의 차고점 전력사용량
t_{N1st} : 테스트 자원 N의 최고점 시각
t_{NS} : 테스트 자원 N의 테스트 시작시각

테스트 자원 그룹의 누적 전력량은 테스트 시작부터 종료까지의 지속적인 연산을 필요로 하지 않으며 각 테스트 자원의 테스트 시작 시각, 최고점의 시각 그리고 종료시각에서만 연산하여 총 전력소비량과 그 값을 비교한다. 그룹화 된 테스트 자원의 각 시각(t_n)에 대한 누적 전력량 P_{tn} 의 일반식은 식 (11)과 같다.

그러므로 A, B, C의 테스트 자원 그룹에 대한 각 시각 (t_n)의 누적 전력량(P_{tn})은 식 (12)과 같다.

$$P_{tn} = \sum_{N=A}^{m} N_{mx2} + \sum_{N=A}^{m} \frac{N_{mx1} - N_{mx2}}{t_{N1st} - t_{NS}} \times t_n \qquad (12)$$
$$(N = A, B, C, \cdots, m)$$

전력 소모량 변이 그래프는 확장나무 성장 그래프의 최종결과인 테스트 자원 그룹을 이용하여 각 테스트 자원의 테스트 시작점, 최고점의 시각 그리고 종료시각에서 P_{tn} 을 연산하고 총 전력소비량 $P_{\max}$ 와 비교하며, 이 값에 따라서 병렬 수행을 결정하여 테스트 자원을 배치하고 각 테스트 자원의 테스트 시작 시각, 최고점의 시각 그리고 종료시각의 전력사용량의 합 P_{tn}을 연산한다.

만약 $P_{tn} < P_{\max}$ 인 경우는 테스트 자원을 그래프에 배치하고 $P_{tn} > P_{\max}$ 인 경우는 테스트 자원 중 우선순위가 낮은 순으로 그룹 내의 테스트 자원을 그룹에서 제외하여 배치한다. 제안된 알고리듬은 최종적으로 테스트 자원의 충돌 및 테스트 전력 충돌이 없는 테스트 자원을 시간변화 전력 소모량 그래프를 이용하여 각 시점에서 제한된 전력 소모량 값과의 비교를 반복적으로 수행하고 최소화된 테스트 총 시간을 산출하게 된다.

4.4 전력변화량에 의한 테스트자원 분할 모델링을 적용한 테스트 스케줄링

테스트 자원의 분할 모델링을 적용한 예제회로 SoC의 테스트 자원은 <표 2>와 같이 나타낼 수 있다[11]. 코어는 6개로 이전 방법과 동일하지만 각 테스트 자원을 전력 소모량의 변화에 따라서 2개 또는 3개로 자원을 분할하여 모델링한다. 상기에 설명된 확장 나무성장 그래프 및 전력 소모량 변이 그래프는 동일하게 사용되지만 동일 자원 내에서 시간적인 전력의 소모량에 따라 분할한 것이기 때문에 스케줄링 알고리듬 과정에서 분할된 테스트 자원내의 소자원은 순서가 바뀔 수는 없다. 예를 들어서 <표 1>의 C1 코어의 1번 자원인 T11은 <표 2>에서와 같이 T11-1, T11-2로 분할될 수 있는데 스케줄링 과정에서 T11-2를 T11-1보다 먼저 수행할 수는 없다.

각 코어 및 테스트 자원에 따라서 각 자원은 (그림 6)에 의하여 총 4가지 방법의 최소자원으로 분할한다.

T23, T61, T67은 (그림 4)(a)와 같은 H-L Type 2 분할하였으며, T12, T35는 (그림 4)(b)와 같은 L-H-L Type 3 분할한다. T11, T21, T34, T56, T68은 (그림 4)(c)와 같은 L-H Type 2 분할하였으며, T41은 (그림 4)(d) H-L-H Type 3분할한다. T44는 전체가 동일한 타입이다.

여기서 제안하는 테스트 스케줄링 알고리듬은 앞서 기술한 2가지의 테스트 자원 모델링 방법과는 상관없이 수행할 수 있도록 설계되었기 때문에 (그림 4)의 1단계인 테스트 자원의 모델링 세부 내용을 테스트 자원의 분할 모델링 방식으로 변경하여 진행하고, 이하의 방식은 (그림 4)에 나타낸 알고리듬의 순서와 동일하게 진행한다.

<표 2> SoC-X 회로의 분할 자원 요소 데이터

Cores (N)	Resource Number (M)	t_n / t_p	P_{MAX1} / P_{MAX2}
C1	1-1	90/ALL	100/100
	1-2	80/40	180/100
	2-1	650/ALL	100/100
	2-2	550/300	210/160
	2-3	330/ALL	160/160
C2	1-1	190/ALL	150/150
	1-2	100/50	250/150
	3-1	200/100	150/70
	3-2	610/ALL	70/70
C3	4-1	110/ALL	100/100
	4-2	140/50	150/100
	5-1	170/ALL	100/100
	5-2	120/60	220/210
	5-3	100/ALL	150/150
C4	1-1	40/ALL	150/150
	1-2	130/ALL	100/100
	1-3	70/ALL	150/150
	4-1	210/ALL	110/110
	4-2	200/ALL	110/110
C5	6-1	800/ALL	100/100
	6-2	800/400	120/100
C6	1-1	170/90	180/150
	1-2	60/ALL	150/150
	7-1	300/140	300/220
	7-2	340/ALL	220/220
	8-1	150/ALL	220/220
	8-2	520/250	320/220

5. 실험 결과

5.1 공간 누적 전력량 감소 실험

제안한 두 가지의 테스트 모델링의 감소될 수 있는 공간의 누적 전력량(P_R)을 실험한다.

첫 번째 모델링 기법인 전력의 최고점과 차고점을 이용한 테스트 자원의 모델링 방법의 감소된 공간의 누적 전력량(P_R)는 식 (3)과 같다.

식 (3)에서 t_n은 테스트 수행시간으로 고정된 값을 가진다. 또한 최고점인 P_{MAX1}은 일반적인 테스트 자원의 모델링 방법에서도 같은 값을 사용함으로 고정된 값으로 가정할 수 있다. 기 사용되었던 일반적인 테스트 자원의 모델링 방법에서는 P_{MAX1}값과 P_{MAX2}의 값이 동일한 값이기 때문에 P_R는 '0(zero)'의 값을 갖는다.

반면에 전력의 최고점과 차고점을 이용한 테스트 자원의 모델링 방법으로는 P_R은 측정되는 P_{MAX2}의 값에 따라서 (그림 8)의 실선과 같이 감소되는 공간 전력 누적량을 통하여 테스트 자원의 병렬 수행 능력을 높일 수 있다.

(그림 8)의 실선은 $P_{MAX2} \leq P_{MAX1}$ 조건에서 P_{MAX1}=100, t_n=200으로 가정한 경우 P_{MAX2}의 변화량에 따른 식 (3)의 P_R를 계산한 결과 그래프이다.

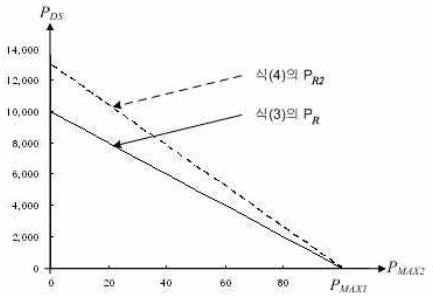
두 번째 모델링 기법인 소모 전력의 변화량에 의한 테스트 자원의 분할 모델링 방법의 감소된 공간의 누적 전력량(P_{R2})는 식 (4)와 같다.

식 (3)과의 동일한 비교를 위하여 식 (4)의 P_1 =P_2=P_{MAX2}라 가정하고, t_S =0, t_F = t_n, t_0 와 t_1은 고정된 일정 값을 가진다고 가정한다.

(그림 8)은 P_1=P_2=$P_{MAX2} \leq P_{MAX1}$ 조건에서 P_{MAX1}=100, t_F = t_n =200, t_0=80, t_1=150 으로 가정한 경우 P_1,P_2 즉, P_{MAX2}의 변화량에 따른 식 (4)의 P_{R2}를 계산한 결과 그래프 및 식 (3)의 P_R 를 계산한 결과를 각 각 점선 및 실선으로 나타낸 그래프이다. 기 사용되었던 직사각형 형태의 테스트 자원 모델링에서는 P_{MAX2}의 변화량이 존재하지 않기 때문에 어떠한 경우에서도 리던던트한 공간이 모두 다 적용되었으나 제안하는 두 가지의 모델링 방법은 P_{MAX1}과 P_{MAX2}의 차이만큼의 리던던트 공간을 최소화 하여 모델링함으로 테스트 자원의 병렬 수행 능력을 효과적으로 높일 수 있음을 보인다.

5.2 테스트 스케줄링 알고리듬 실험

생성된 테스트 자원의 데이터는 앞에서 설명한 테스트 자원의 모델링 방법에 따라서 첫째



(그림 8) 공간의 누적전력량 비교 그래프

일반적인 테스트 자원 모델링 방법에 의한 테스트 자원, 둘째 최고점 및 차고점을 이용한 모델링 방법에 의한 테스트 자원 그리고 마지막으로 테스트 자원의 분할 모델링 방법에 의한 테스트 자원으로 각각 생성한다.

첫 번째 방법으로 생성된 자원은 기 연구된 테스트 스케줄링 알고리듬인 DP-PCTS[4] 기법으로 스케줄링 한다. 또한 두 번째와 세 번째 방법으로 생성된 제안하는 테스트 모델링 자원은 여기서 제시하는 휴리스틱 테스트 스케줄링 알고리듬으로 스케줄링을 진행한다.

총 3가지의 방법으로 테스트 스케줄링을 진행한 후 도출되는 최종 테스트 수행 시간을 비교하여 결과를 얻는다. 테스트의 전력 사용량 최고값(P_{max})을 변화시켜가며 실험을 진행하고 나머지의 실험 조건은 동일하게 진행한다.

테스트 자원은 <표 1>과 같이 테스트 자원을 최고점 및 차고점을 이용한 방법 및 <표 2>와 같이 테스트 자원의 분할 모델링 방법으로 자원을 모델링한다.

본 실험에서는 전력 사용량 최고값(P_{max})을 800, 1,000, 1,200으로 3가지로 설정하여 각 테스트 스케줄링을 구현하고 최종 테스트 시간을 도출하였다.

기존에 제안된 DP-PCTS 테스트 스케줄링 방법으로 테스트 스케줄링 및 총 테스트 시간을

실험하고, 모델링된 테스트 자원을 통하여 확장 나무 성장 그래프를 생성하고 시간변화 전력 소모량 그래프를 생성하는 휴리스틱 테스트 스케줄링 알고리즘을 통하여 테스트 스케줄링 및 총 테스트 시간을 실험한 결과 <표 3>과 같은 실험 결과를 얻을 수 있다.

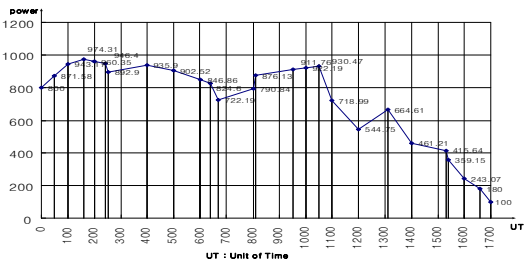
<표 3> 실험 결과 (테스트 시간)

제한전력	800	1,000	1,200	평균
제안 알고리듬1*	3,770	1,700	1,600	2,357
제안 알고리듬2**	3,650	1,620	1,550	2,274
DP-PCTS	5,240	4,990	3,130	4,453
제안 알고리듬1에 대한 감소율	28%	65%	51%	48%
제안 알고리듬2에 대한 감소율	30%	67%	51%	49%

* : 최고점과 차고점을 이용한 테스트 자원 모델링 적용한 테스트 스케줄링 알고리듬
** : 테스트 자원의 분할 모델링 방법을 적용한 테스트 스케줄링 알고리듬

제안한 테스트 스케줄링 알고리듬은 DP-PCTS 알고리듬과 비교하여 약 30%이상의 테스트 시간 감소 및 3배 이상의 병렬 수행 능력이 향상되었음을 알 수 있다. (그림 9)는 예제 SoC 회로를 대상으로 제한 전력 1,000 인 경우 최고점과 차고점을 이용한 테스트 자원 모델링 기법으로 테스트 자원을 생성하고 제안하는 테스트 스케줄링 알고리즘으로 수행한 테스트 스케줄링 결과를 나타내는 전력 소모량 변이 그래프이다. (그림 9)의 결과 그래프는 제한전력인 1,000 이내에서 테스트 스케줄링의 최종 결과 최종테스트 시간이 1700ut(ut: unit of time)을 나타낸다.

제한 전력과 시간단위 ut는 실질적인 시간이나 전력량과는 다른 알고리즘 수행상의 기준단위이다.



(그림 9) 최종 결과 그래프(전력 제한량 1,000)

6. 결론

본 논문에서는 복잡도가 높은 SoC 회로를 과도한 발열로 인한 시스템의 손상 없이 안전하면서도 최소의 시간으로 테스트하기 위한 방법으로서 소비전력 내에서 최적화된 테스트를 진행할 수 있는 테스트 자원의 모델링 방법과 테스트의 수행시간을 최소화하는 휴리스틱 테스트 스케줄링 알고리듬을 제안하였다.

2가지의 모델링 방법을 제안하였는데 첫 번째 모델링 방법은 SoC 내 각 코어들의 테스트 전력 소모량을 조사하여 전력사용량의 최고점 및 차고점을 검출하고 최고점과 차고점의 시각을 기준으로 테스트 자원을 5각형의 형태로 모델링하는 것이다. 두 번째 모델링 방법은 테스트 과정의 전력 소모량 변화의 크기에 따라서 단일 테스트 자원을 무위 테스트 시간을 감소시킬 수 있도록 다수개의 테스트 자원으로 분할하여 모델링하는 것이다.

제안된 2가지의 모델링 방법은 파라미터의 변화에 따른 공간 누적 전력량 감소 실험을 통하여 2배 이상의 테스트 자원의 병렬 사용 능력이 증가됨을 확인할 수 있었다. 이러한 결과는 본 제안이 제한적인 전력사용량 내에서 테스트를 진행할 수 있는 동시에 테스트 시간 단축 효과를 얻을 수 있음을 보인다.

제안된 모델링 방법을 적용하는데 적합한 휴리스틱 테스트 스케줄링 알고리듬은 그 복잡도가 크게 증가하지 않도록 단계별로 구성하였다. 제안한 테스트 모델링 방법에 의하여 테스트 자원을 모델링하고 알고리듬에 입력하면 확장나무 성장 그래프 단계에서는 자원의 충돌이 없는 조합을 효과적으로 구성하고, 다음으로 시간변화 전력 소모량 그래프 단계에서는 전력의 충돌이 없는 자원 조합을 조건에 적합하도록 반복적으로 배치함으로 테스트 자원을 효과적으로 스케줄링 할 수 있다.

실험을 통하여 모델링의 변경을 통한 알고리듬의 효율 증가를 확인하였으며, 제안된 휴리스틱 테스트 스케줄링 알고리듬은 다양한 제한 소비전력을 가정한 테스트 스케줄링 실험을 통하여 기 개발된 타 알고리즘과 최종 테스트 시간을 비교하여 보았다.

제안된 휴리스틱 테스트 스케줄링 알고리듬은

알고리듬의 스케줄링 생성 시간도 이전에 비하여 단축되었으며 제한 전력 소모량 내에서 병렬 사용능력 향상 및 무위 테스트 시간 단축을 통한 최적의 테스트 시간 결과를 스케줄링 할 수 있는 장점 및 각 단계별 시각적으로 확인할 수 있는 기능적 향상을 보였다.

참 고 문 헌

[1] Paul Rosinger, Bashir M. Al-Hashimi, and Krishnendu Chakrabarty, "Thermal-Safe Test Scheduling for Core-Based System-on-Chip Integrated Circuits", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 25, No. 11, pp. 2502-2511, 2006.

[2] Zhiyuan He; Zebo Peng; Eles, P., "Power Constrained and Defect-Probability Driven SoC Test Scheduling with Test Set Partitioning" Design, Automation and Test in Europe, 2006. Proceedings, pp. 291-296, 2006.

[3] Vikram Iyenger and K. Chakrabarty, "Precedence-Based, Preemptive, and Power-Constrained Test Scheduling for System-on-a-chip" 19th IEEE Proceedings on VLSI Test Symposium, pp. 368-374, 2001.

[4] Zhao D. and S. Upadhyaya, "Dynamically Partitioned Test Scheduling for SoCs Under Power Constraints," 11th IEEE North Atlantic Test Workshop, Montauk, NY, 2002.

[5] Dan Zhao and Shambhu Upadhyaya, "A Resource balancing approach to SOC Test Scheduling". Circuits and Systems, 2003. ISCAS'03. Proceedings of the 2003 International Symposium on, Volume 5, 2003.

[6] Krishnendu Chakrabarty, "Test Scheduling for Core-based Systems Using Mixed-Integer Linear Programming," IEEE Trans. On computer-Aided Design of Integrated circuits and Systems, vol. 19, No. 10, pp. 1163-1174, 2000.

[7] Zhao D., S. Upadhyaya and M. Margala, "Minimizing Concurrent Test Time in SoCs by Balancing Resource Usage," Proceedings of the 12th ACM Great Lakes Symposium on VLSI, pp. 77-82, 2002.

[8] M. Sugihara, H. Data, and H. Yasuura, "Analysis and minimization of test time in a combined BIST and external test approach," in Design, Automation and Test in Europe Conf.2000, 2000.

[9] Erik Larsson and Zebo peng, "System-on-Chip Test Parallelization Under Power Constraints," European Test workshop, Stockholm Sweden, 2001.

[10] Zhao D. and S. Upadhyaya, "Dynamically Partitioned Test Scheduling for SoCs Under Power Constraints," 11th IEEE North Atlantic Test Workshop, Montauk, NY, May 2002.

[11] E.J. Marinissen, V. Iyengar and K. Chakrabarty. ITC 2002 SOC test bench-mark initiative. http://www.extra.research. philips.com/itc02socbenchm

이 재 민

1979년 : 한양대학교 전자공학과 (공학사)
1981년 : 한양대학교 대학원 전자공학과(공학석사)
1987년 : 한양대학교 대학원 전자공학과(공학박사)
1990년～1991년 : UIUC Beckman Institute 연구원
2001년～2002년 : SUNY(Buffalo), Visiting Professor
1986년～현 재 : 관동대학교 전자정보통신공학부(교수)
관심분야 : 멀티미디어 SoC(System-on-Chip) 설계, 알고리듬 및 CAD등

이 호 진

2004년 : 관동대학교 대학원 전자정보통신공학부(공학사)
2006년 : 관동대학교 대학원 전자공학과(공학석사)
2006년～현 재 : 관동대학교 전자공학과 박사과정
관심분야 : 멀티미디어 회로 설계, 제어회로 설계

박 진 성

2000년 : 관동대학교 전자공학과 (공학사)
2003년 : 관동대학교 대학원 전자공학과(공학석사)
2004년～현 재 : 라이벌코리아 핵심기술연구소 선임연구원
관심분야 : 멀티미디어 회로 설계 및 알고리듬 개발