# 저 전력을 고려한 스캔 체인 수정에 관한 연구

박수식[0] 김인수 정성원 민형복
성균관대학교
{ sspark[0], iskim, swjung, min }@ece.skku.ac.kr

# Scan Chain Modification for Low Power Design

Susik Park[0] Insoo Kim Sungwon Jung Hyoung Bok Min
Sungkyunkwan university

요 약

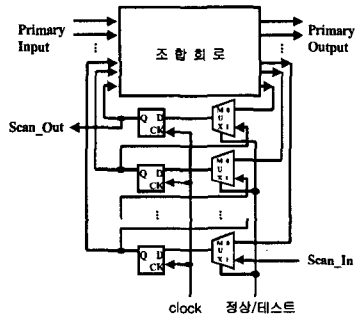
이동기기들이 늘어가고 있는 추세에서 기기들의 구성품인 디지털 회로들의 테스트 시간과 전력소모는 성능에 상당한 영향을 미친다. 테스트 시간을 줄이는 방법은 병렬 코어 테스트 방법으로 줄일 수 있으나, 다양한 코어들이 동시에 테스트 되면 많은 전력 소모를 야기 시킨다. 스캔 구조를 기반으로 한 회로에서 전력 소모는 테스트 데이터의 불필요한 천이에 의해 많이 발생한다. 그러므로 스캔 체인을 수정함으로 인해 입력 값과 출력 값의 천이를 줄일 수 있다. 제안하는 스캔 체인의 수정은 스캔 셀의 재배치와 특정한 회로의 추가로 이루어진다. 또한 회로의 추가는 그에 적합한 그룹화를 시킴으로써 최소의 수를 결정한다. 천이 주기를 해석하여 효과적인 알고리즘을 세움으로써 최적의 스캔 체인구조와 그룹을 구함으로써 전력 소모를 최소화할 수 있다.

## 1. 서론

SOC 코어들의 테스트는 테스트 시간을 줄이기 위해 병렬적으로 이루어진다. 이런 병렬 테스트는 코어들에 무리한 전력을 인가하여 손상을 줄 수도 있다. 물리적인 손상을 막기 위해서는 평균 전력과 최대 전력을 신중하게 고려하여야 한다. 전력 소모를 줄이는 방법으로는 코어가 정확한 동작을 할 수 있는 범위 내에서 clock 속도를 조절하는 방법이 많이 쓰인다. 하지만 clock 속도를 낮추는 방법을 선택하기 때문에 테스트 시간이 증가하는 딜레마에 빠지게 된다[4]. 본 논문에서는 시간을 늘이지 않고 전력 소모를 줄이는 방법을 제안한다. 테스트 모드동안 과도한 이벤트는 테스트 데이터가 전달되는 동안 각 플립플롭이 그 앞의 값을 받기 때문에 회로의 모든 플립플롭이 모두 동작하게 되는 일이 발생할 수 있다. 하지만 코어가 테스트 모드가 아닌 실질적인 동작에서는 몇몇 플립플롭만이 매 주기에 동작한다. 그러므로 테스트 전력의 과도한 소모 문제는 저 전력으로 설계된 노트북이나 휴대폰과 같은 이동 장비에서는 중요한 문제가 되고 있다. 전력 소모 문제의 요인은 스캔 체인에서 천이이지만 가장 큰 요소는 CUT(Circuit Under Test)에서 이루어지는 switching activity 이다. 테스트 데이터가 스캔 체인에 전달되어 흘러갈 때 불필요한 switching 이 CUT 에서 발생하기 때문이다. CUT를 테스트하기 위해서는 각 스캔 셀에 그에 해당하는 테스트 벡터 값들이 있어야 한다. 하지만 스캔 셀들이 재정렬되고 스캔 경로에 회로가 추가되면 의도한 값들이 스캔 셀에 입력되지 않는다. 이것을 고려하여 테스트 벡터들을 조정하고 이 벡터들의 천이주기를 이용하여 스캔 셀들을 재 정렬하고 스캔 체인 경로에 회로를 추가할 수 있다[4]. 이렇게 스캔 체인내의 셀들을 재 정렬하고 몇몇 부분에 회로를 추가하는 것으로 스캔 체인에서의 천이를 줄여 테스트 데이터가 전달되는 동안 전력 소모를 줄이는 것을 본 논문에서 제안한다. 또한 추가회로를 설치할 위치와 제어벡터 별로 그룹화하여 더욱 전력 소모를 줄일 수 있다. 제안된 스캔 체인의 수정은 회로의 테스트 동작이 아닌 실 동작시의 성능 저하를 일으키지는 않는다..

## 2. 기본적인 개념

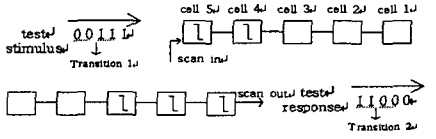
예전에는 조합회로로 이루어져 있어서 주입력과 주출력 부분만을 확인하여 테스트를 할 수 있었으나 현재에는 일반적인 디지털 회로들이 조합회로와 순차회로로 이루어져 있어 순차회로에 의한 변화를 예측하기 어렵다. 이를 용이하게 하기위해 쓰이는 한 방법이 스캔 테스트 기법[1][2][6]이다. 스캔 테스트는 순차회로의 기본을 이루는 기억소자(Memory element)들을 shift register 형식으로 연결하여 테스트 모드에서 외부로부터 쉽게 제어할 수 있도록 한 테스트의 방법이다. 즉 CUT에 인가될 순차회로의 값들을 외부에서 인위적으로 인가할 수 있다[1][2][6].




[그림1] 완전스캔설계기법을 적용한 순차회로

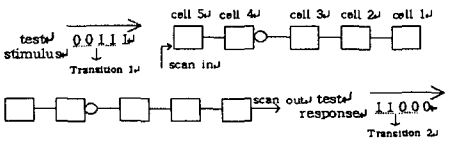
## 3. 이전 연구

각 스캔 셀들은 CUT를 테스트하기 위해 테스트 벡터라고 하는 정해진 값들을 가져야 한다. 그리고 이 값들이 적용되어 출력 핀으로 나오는 값을 비교하여 회로가 정확한 동작을 실행하는지를 판별한다. 이와 같은 테스트 벡터들을 생성해주는 것이 ATPG(Automate Test Pattern Generator) 이다. 이런 연관성 때문에 스캔 체인을 수정할 경우 입력 값과 출력 값을 변환시켜야 한다.




[그림 2] 테스트벡터천이 중 스캔셀내의 값 변화

[그림 2]에서 스캔 체인 내에서 값이 전달될 때 나타나는 현상을 보여주고 있다. "00111"이 정해진 위치에 머물기까지 셀4와 5에서 천이가 일어나는 것을 볼 수 있다.




[그림 3] 스캔셀내의 값변화를 줄이기위한 스캔체인 수정

[그림 3] 과 같이 셀4와 셀3 사이에 인버터를 추가하면 입력 벡터는 "00000"로 변환되고 CUT 로 천이가 전달되지 않는다. 또한 출력값의 변화도 보여주고 있다. 각 셀들 사이의 평균적인 천이 주기를 계산하고 가장 높은 천이 주기를 가지는 셀들 사이에 인버터를 사용하면 천이 주기를 줄일 수 있는 것을 위에서 보았다. 여기에 인접한 다른 셀과 교환하고 위의 방법을 적용하면 그림 4와 같다. 왼쪽에서는 셀1과 2 사이에서 천이 주기가 높으므로 이곳에서 반전을 할 경우 0.2가 줄어드는 것을 볼수 있다. 하지만 오른쪽에서는 2와 3의 위치를 변화시킨 후 반전하면 0.6이 줄어드는 것을 볼 수 있다.[4] 이와 같이 스캔 셀들을 재 정렬하고 인버터를 추가하면 천이를 상당량 줄일 수 있다.

## 4. 동기

### 4.1 알고리즘

두개의 특정한 비트가 있고 이 두 비트가 충돌이 일어나는 경우를 백분율로 나타낸 것을 $S_{Tr}$ 이라고 하고 동일한 비트값을 가지는 경우는 $S_{NoTr}$ 이라 한다.

i 번째와 j 번째 비트의 천이 주기는[4]

$$S_{Tr}(i,j) = \frac{1}{|T|}\sum_{k\in T} t_k[i] \text{ 쮤 } t_k[j]$$

$$S_{NoTr}(i,j) = \frac{1}{|T|}\sum_{k\in T} (t_k[i] \text{ 쮤 } t_k[j])'$$

이와같이 표현된다.

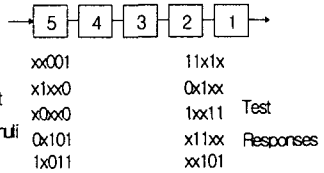
여기서 $t_k[i]$ 는 k 번째 테스트 벡터의 I 번째 비트를 나타낸다.

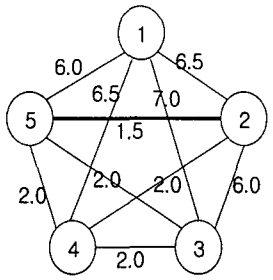
$$\begin{aligned} r_{Tr}(i,j) &= \frac{1}{|T|}\sum_{k\in T}\{\,cont(t_k[i],0)\times cont(t_k[j],1) \\ &\quad + cont(t_k[i],1)\times cont(t_k[j],0)\,\} \\ &= \frac{1}{|T|}\sum_{k\in T}\{\,cont(t_k[i],0)+cont(t_k[j],0) \\ &\quad - 2\times cont(t_k[i],0)\times cont(t_k[j],0)\,\} \\ &= \frac{1}{|T|}\sum_{k\in T}\{\,cont(t_k[i],1)+cont(t_k[j],1) \\ &\quad - 2\times cont(t_k[i],1)\times cont(t_k[j],1)\,\} \end{aligned}$$

위 식에서 rtr( i, j ) 는 k 번째 테스트 벡터가 적용될 때 v에서 I 번째 출력값의 controllability 기댓값을 cont ($t_k[i]$, v) 가 만들어내는 동안 i 번째와 j 번째 출력 비트사이의 천이주기를 표현한다. $r_{NoTr}$ 은 i 와 j 사이에 천이주기가 없는 것을 말한다. 그러므로 $r_{NoTr}$ 과 $r_{Tr}$ 이 합은 항상 '1' 이다[4].

$$ETC(i,j) = \min(S_{Tr}(i,j)\times(N-I) + r_{Tr}(i,j)\times I,\; S_{NoTr}(i,j)\times(N-I) + (r_{NoTr}(i,j))\times I)$$

위 식은 i 와 j 번째 셀이 인접한 위치에 옮겨질 확률상의 값을 계산하는 식이다[4]( N: 스캔 체인의 길이, I: en 스캔 셀이 옮겨질 위치). 두 스캔 셀 사이의 관계성을 보면 어느 셀들 사이에 인버터를 추가할지 결정할 수 있다. 하지만 stimuli에서 don't care가 있을 경우 controllability 값이 변경될 것이고 stimuli와 response들의 천이 주기를 다시 계산해야 된다.



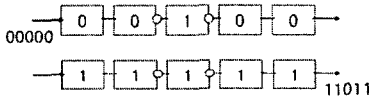




[그림 4] transition cost 그래프 구조

위 알고리즘의 예를 들면 [그림 4]에서 가장 최소의 연결은 2와 5 노드사이가 되고 이것이 첫 번째 묶음이 된다. 그 다음은 3과 5이고 이 두 묶음들 사이도 적용될 수 있다. 하지만 1과 3, 1과 5 사이에는 상당히 높은 역관계가 성립되므로 이들 사이에 인버터를 삽입하여 좀 더 높은 성능을 얻을 수 있다. 결론적으로 SCAN_IN -> 4 -> 3 -> 인버터 -> 1 -> 인버터 -> 5 -> 2 -> SCAN_OUT 의 순서를 가지는 수정된 스캔 체인을 얻을 수 있다.

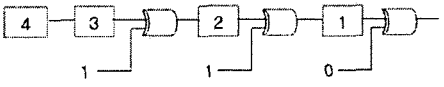
### 4.2 그룹화

앞에서 언급한 방법은 스캔 체인의 순서를 바꾸고 벡터를 반전시켜서 스캔 체인 내에서 최소의 천이를 가지도록 하는 방법이었으나 이 방법은 소수의 테스트 벡터를 가지거나 패턴들이 서로 비슷한 경우 위 알고리즘을 통해서 최적화시킬 수 있으나 상당량의 테스트 벡터를 가지거나 패턴들이 상당한 다른 경우 불필요한 반전에 의한 천이를 발생시켜 오히려 천이량을 늘이는 상황을 초래한다.




[그림 5] 불필요한 반전 예

위 그림에서 셀들에 { 00000 } 이 위치하기 위해서는 입력값으로 { 00100 } 인가된다. 이것은 인버터를 추가하지 않은 경우와 비교하여 1이 자신의 자리를 찾아가는 동안 2번의 불필요한 천이를 발생시킴을 볼 수 있다. 이에 테스트 벡터를 그룹화 시켜서 더 세밀한 부분까지 제어하려고 한다. 즉 인버터대신 Exclusive-OR( ⊕ )를 사용하여 인버터로 사용할 수도 있고 단순히 통과시킬 수도 있는 제어회로를 추가하여 그룹에 따라 Exclusive-OR를 제어함으로써 CUT로의 천이를 막을 수 있다.




[그림 6] 제안하는 회로

위 그림이 제안하고자 하는 회로모델이다. Exclusive-OR의 제어값으로 '1'을 가지면 인버터동작을 하고 '0'인 경우 통과시킨다. 이렇게 인버터에 의해 불필요한 천이를 하는 부분들 중 비슷한 테스트 벡터를 그룹화하여 새롭게 인버터 연결을 맺어줌으로서 좀 더 세밀한 부부까지 천이 수를 줄일 수 있다.

### 4.3 그룹화 알고리즘

본 논문에서 가장 중심이 되는 그룹화는 ETC 알고리즘의 회귀적인 방법으로 이루어진다. 앞서 소개한 ETC 수식을 알고리즘화하여 수행하면 제안한 방법을 적용하기 전 구조가 되지만 이것은 상당량의 불필요한 transition을 발생시키게 되는 것을 보았다. 이에 ETC 알고리즘을 수행 후 알맞지 않은

패턴들을 수집하여 알고리즘을 재 수행하는 것의 반복으로 그룹화하는 방식을 채택하였다.

```
RE_ETC(group_mumber){
  FOR( I = 0 ; I <= TEST_VECTOR_NUMBER ; I++){
    IF (ETC() != TEST_VECTOR( I ) ) {
      RE_ETC(group_number + 1); }
    GROUP(group_number)<=ETC(); }
```

제안하는 방법은 위와 같이 간략화 해서 나타낼 수 있다. 먼저 ETC 알고리즘에 의해 최적의 셀 위치와 인버터의 추가 위치를 결정한다. 인버터의 추가 위치에서 천이가 일어나지 않는 벡터들을 추출하고 추출된 패턴들로 알고리즘을 재 수행하는 것이 IF문으로 표현되어 있다. 이런 알고리즘의 반복에 의한 방법은 수행시간이 상당히 늘어나는 단점이 있으나 비교적 그룹화하기 쉽고 패턴의 수가 많은 경우 그룹화하는 것이 효율적이다.

### 5. 실험결과

제안한 테스트 이벤트 수 감소에 관한 구조는 ISCAS89 회로에 풀-스캔 방식으로 적용하였다. 회로의 합성은 SYNOPSYS™의 Design_Analyzer 를 사용하였고 테스트 벡터는 SYNOPSYS™ 의 TetraMax 에서 추출하였다. 그리고 CADENCE™의 VerilogXL 에서 시뮬레이션 결과를 확인하였다.

| | | 기존방법 | 2개그룹 | 3개그룹 | 4개그룹 | 하프그룹 |
|---|---|---|---|---|---|---|
| S208 (10571) 패턴수:30 | 이벤트 수 | 10490 (0.8%) | 10304 (2.5%) | 10269 (2.8%) | 9755 (7.1%) | 8562 (19%) |
| | CPU time | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 |
| | 추가 게이트 수 | 0 | 3 | 4 | 4 | 4 |
| S510 (43693) 패턴수:67 | 이벤트 수 | 43536 (0.4%) | 43025 (1.5%) | 41328 (5.4%) | 38712 (11.4%) | 34086 (22%) |
| | CPU time | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 |
| | 추가 게이트 수 | 0 | 2 | 2 | 3 | 3 |
| S1488 (122588) 패턴수:105 | 이벤트 수 | 119645 (2.4%) | 114252 (6.8%) | 109471 (10.7%) | 102361 (16.5%) | 92618 (24.4%) |
| | CPU time | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 |
| | 추가 게이트 수 | 0 | 3 | 4 | 5 | 4 |
| S1423 (1832473) 패턴수:69 | 이벤트 수 | 1806818 (1.4%) | 1753676 (4.3%) | 1605246 (12.4%) | 1530114 (16.5%) | 1304720 (28.8%) |
| | CPU time | 1.5 | 1.5 | 1.3 | 1.3 | 1.1 |
| | 추가 게이트 수 | 0 | 44 | 52 | 39 | 49 |
| S5378 (24759989) 패턴수:213 | 이벤트 수 | 24487629 (1.1%) | 23967669 (3.2%) | 23150589 (6.5%) | 20798490 (15.9%) | 18315284 (26%) |
| | CPU time | 16.9 | 16.9 | 16.9 | 16.9 | 12.9 |
| | 추가 게이트 수 | 0 | 93 | 121 | 85 | 96 |
| S5378 (24759989) 패턴수:213 | 이벤트 수 | 24487629 (1.1%) | 23967669 (3.2%) | 23150589 (6.5%) | 20798490 (15.9%) | 18315284 (26%) |
| | CPU time | 16.9 | 16.9 | 16.9 | 16.9 | 12.9 |
| | 추가 게이트 수 | 0 | 93 | 121 | 85 | 96 |
| S9234 (13768138) 패턴수:136 | 이벤트 수 | 13396398 (2.7%) | 12763063 (7.3%) | 12198570 (11.4%) | 11287221 (18%) | 10036972 (27.1%) |
| | CPU time | 7.8 | 7.3 | 7.0 | 7.0 | 6.7 |
| | 추가 게이트 수 | 0 | 145 | 196 | 168 | 174 |

[표 1] 테스트 이벤트 수와 CPU time 비교

위 결과 [표 1]을 보면 기존의 방법을 채택하여 하나의 그룹만 가졌을 경우는 별다른 효과를 기대하기 힘들다는 것을 볼 수 있다. 차츰 그룹의 수를 높여감에 따라 이벤트의 수들은 줄어가고 하프 그룹에서는 상당량이 줄어 든 것을 볼 수 있다. 여기서 하프 그룹이라고 명명한 그룹은 테스트 벡터의 수의 절반만큼 그룹을 나누는 것을 의미한다. 하프 그룹보다 많은 그룹을 채택할 경우 이벤트의 수는 줄어들지만 그 양이 평이하여 그룹의 제어벡터를 저장하는 메모리의 면적만 증가 시킬 뿐이다. 그래서 하프 그룹이 가장 최적화 시킬 수 있는 그룹의 수임을 확인하였다. 또한 내부 회로를 많이 가지는 Chip에서는 그룹의 수를 늘려감에 따라 CUT에서의 동작이 줄어들어 CPU time도 줄어드는 것을 볼 수 있다.

### 6. 결 론

본 논문에서는 기존에 제안된 스캔 셀의 위치 조정과 스캔 체인내에 추가적인 회로를 부가하여 저전력 테스트를 하는 방법을 개선하여 기존 알고리즘을 회귀적인 방법으로 적용하여 그룹별로 테스트 벡터를 나눈 것에 의하여 저전력 테스트를 실행하였다. 테스트 벡터들을 세부 그룹들로 나누어 그 벡터들에 적합한 인버터동작을 하도록 제어회로를 추가하고 이를 동작시키는 제어벡터를 생성시킨다. 이 방법을 CUT가 많은 조합회로로 구성된 S5378 회로에 적용하여 실험한 결과 이벤트 수는 26%, CPU time은 23.6%의 감소를 얻을 수 있었다.

### 참 고 문 헌


[1] A. Shen, A. Ghosh, S.Devadas, and K. Keutzer, "On average power dissipation and random pattern testability of COMS combinational logic networks", in Proc. ACM/IEEE Int. Conf. Computer Aided Design, 1992, pp. 402-407.

[2] A. Ghosh, S. Devadas, K. Keutzer, and J. White, "Estimation of average switching activity in combinational and sequential circuits", in Proc. ACM/IEEE 29th Design Automation Conf. , 1992, pp. 253-259.

[3] O. Sinanoglu, I. Bayraktaroglu and A. Orailoglu, "Test Power Reduction Through Minimization of Scan Chain Transitions", in VTS, 2002, pp. 155-161.

[4] Ozgur Sinanoglu, I. Bayraktaroglu and A. Orailoglu, "Scan Power Reduction Through Test Data Transition Frequency Analysis", in ITC. 2002, pp. 844-850.

[5] V. Dabholkar, S. Chakravarty, I. Pomeranz and S. M. Reddy, "Techniques for minimizing power dissipation in scan and combinational circuits during test application", IEEE TCAD, 1998, vol. 17, n. 12, pp. 1325-1333.

[6] M. Abramovici, M. Breuer, and A. Friedman, "Digital Systems Testing and Testable Design.", New York: Computer Sciencce Press, 1990.