論文2003-40SD-11-7

제어 회로를 위한 효율적인 비주사 DFT 기법

(An Efficient Non-Scan DFT Scheme for Controller Circuits)

沈 載 憲 *, 金 文 俊 *, 朴 在 興 *, 梁 善 雄 *, 張 勳 **

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

요 약

본 논문에서는 완벽한 고장 효율을 보장하는 제어 회로를 위한 효율적인 비주사 DFT(design for testability) 기법을 제안한다. 제안된 비주사 DFT 기법은 순차 회로 모델이 아닌 조합 회로 모델에 대하여 ATPG(automatic test pattern generation)를 수행함으로써 짧은 테스트 패턴 생성 시간과 항상 완벽한 고장 효율을 보장한다. 본 논문에서 제시된 기법은 완전 주사 기법 및 기존의 비주사 DFT 기법들과 비교하여 적은 면적 오버헤드를 가지며 테스트 패턴을 칩의 정상동작속도로 인가할 수 있고, 또한 테스트 패턴의 재배열과정을 통해 테스트 패턴을 최소한의 시간으로 인가할 수 있도록 하였다. 제안된 기법의 효율성을 검증하기 위해 MCNC'91 FSM 벤치마크 회로들을 이용하여 실험을 수행하였다.

Abstract

In this paper, an efficient non-scan design-for-testability (DFT) method for controller circuits is proposed. The proposed method always guarantees a short test pattern generation time and complete fault efficiency. It has a lower area overhead than full-scan and other non-scan DFT methods and enables to apply test patterns at-speed. The proposed method also shortens the test application time through a test pattern re-ordering procedure. The efficiency of the proposed method is demonstrated using well known MCNC'91 FSM benchmark circuits.

Keyword : Non-Scan DFT, Fault Efficiency, FSM, Controllability, Obversability

Ⅰ. 서 론

VLSI 칩의 복잡도 증가로 테스트 비용이 크게 증가하고 있다. 테스트 비용의 증가는 테스트 패턴 생성의 어려움에 기인하며, 이러한 문제점을 해결하기 위하여 다양한 DFT 기법이 적용되고 있다. 특히 조합 회로가 아닌 순차 회로로 구성되어 있는 제어회로의 경우, ATPG에 의해 완벽한 고장 효율을 보장하는 테스트 패턴을 생성할 수 없기 때문에 주사 기법을 적용하는 것이 일반적이다[1-3]. 그러나 주사 기법이 짧은 시간에 높은 고장 검출율을 갖는 테스트 패턴을 생성하기에는 적합하지만, 회로내의 기억 소자를 주사 가능한 소자로 변경하는 과정에서 많은 면적 오버헤드가 발생하고, 테

* 正會員, 崇實大學校 大學院 컴퓨터學科
(Department of Computing, Graduate School, Soongsil University)
** 正會員, 崇實大學校 컴퓨터學部
(School of Computing, Soongsil University)
※ 본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음
接受日字:2003年1月29日, 수정완료일:2003年11月6日

스트 패턴을 칩에 인가하는데 많은 시간이 소요된다는 점, 그리고 칩의 정상 동작 속도로 테스트 패턴을 인가할 수 없다는 점 등의 단점들이 존재한다[4, 5]. 동일한 고장 검출율을 갖는 테스트 패턴을 주사 모드와 정상 동작 속도로 인가했을 때 정상 동작 속도로 인가하는 경우의 실제 고장 검출률이 더 높게 측정된다는 것은 잘 알려진 사실이다[6, 7]. 이러한 사실에 기반하여 제어 회로 테스트를 위한 비주사 DFT 기법에 관한 연구가 많이 이루어졌다[6, 8, 9].

기억 소자의 앞단에 멀티플렉서를 삽입하고 테스트 모드시에 입력으로부터 직접 기억 소자에 원하는 패턴을 인가할 수 있는 테스트 포인트 삽입 방법이 제안되었다[8]. 회로의 입력이 기억 소자의 개수보다 같거나 많을 경우, 모든 기억 소자는 회로의 입력으로부터 제어가 가능하기 때문에 조합회로에 대한 테스트 패턴 생성 알고리즘이 적용될 수 있다. 그러나 기억 소자에 비해 회로의 입력의 개수가 적을 경우 일부 기억 소자는 회로의 주입력에서 제어가 불가능하기 때문에 순차회로를 위한 테스트 패턴 생성 알고리즘을 적용해야 하므로 테스트 포인트 삽입 방법에서 제안하고 있는 방법은 항상 완벽한 고장 효율을 보장하지 못한다. 이를 보완한 기법으로써 입력의 개수에 관계없이 항상 완벽한 고장 효율을 보장하는 기법이 제안되었으나, 이 기법은 추가된 멀티플렉서들 외에 그 멀티플렉서들을 제어하는 회로가 추가되어 완전 주사 기법과 테스트 포인트 삽입의 방법보다 면적 오버헤드가 월등히 크다는 단점이 있다[9].

본 논문에서는 기존의 주사, 비주사 DFT 기법들에 비해 적은 면적 오버헤드로 완벽한 고장 효율을 보장할 수 있는 기법을 제안한다. 제안된 기법은 조합회로를 위한 테스트 패턴 생성 알고리즘을 적용하여 완벽한 고장 효율을 보장하는 테스트 패턴을 생성한다. 본 논문에서 제안한 기법의 효율성을 검증하기 위해 MCNC'91 FSM 벤치마크 회로들을 이용하여 실험하였다.

본 논문의 전체적인 구성은 다음과 같다. Ⅱ장에서는 제안된 비주사 DFT 기법 적용의 전체흐름과 제안된 기법을 적용하기 위한 제어 회로 모델에 대해 설명하고 Ⅲ장에서는 제안된 기법의 각 절차에 대한 기술을 상세히 한다. Ⅳ장에서는 제안된 기법의 효율성을 검증하기 위해서 MCNC'91 FSM 벤치마크 회로들을 이용한 실험을 기술하고 Ⅴ장에서 결론을 맺는다.

Ⅱ. 제어 회로를 위한 비주사 DFT 기법

Ⅱ장에서는 본 논문이 제안하는 비주사 DFT 기법이 적용되기 위해 필요한 제어 회로의 모델과 제안하는 기법의 전체 흐름도에 대해 설명한다.

1. 제어 회로 모델

제어 회로는 유한 상태 머신(finite state machine), 즉 순차 회로로 구현된다. 이러한 순차회로의 경우 귀환 (feedback) 루프를 내장하고 있기 때문에 순차회로를 위한 테스트 패턴 생성 알고리즘을 적용하여야 하지만 순차회로를 위한 테스트 패턴 생성 알고리즘은 테스트 패턴을 생성하는 데 오랜 시간이 소요되고 완벽한 고장 효율을 보장하는 테스트 패턴 생성을 항상 보장할 수는 없다. 따라서 본 논문에서 제안하는 비주사 DFT 기법은 제어 회로를 조합회로 모델로 변경하여 테스트 패턴을 생성한다. 본 논문에서는 항상 완벽한 고장 효율을 보장하는 조합회로를 위한 테스트 패턴 생성 알고리즘을 적용하기 위해 순차회로의 기억 소자 입력과 출력을 각각 회로의 출력과 입력으로 변경한 조합회로 모델을 사용하였다. <그림 1>은 일반적인 순차회로의 구조와 테스트 패턴 생성을 위해 변경된 조합회로 모델을 보여준다.

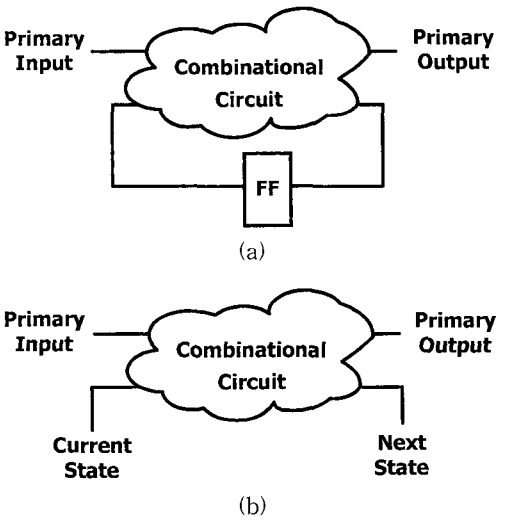


그림 1. 순차회로 모델과 변경된 조합회로 모델 (a) 순차회로 모델 (b) 조합회로 모델

Fig. 1. Sequential Circuit Model and Modified Circuit Model. (a) Sequental Circuit Model (b) Combinational Circuit Model

본 논문에서는 상태 할당이 이진 부호화 (binary encoding) 되었으며, 상태 x를 S_x라고 표현하고 m개의 상태를 표현하기 위해 S_0, S_1, …, S_{m-2}, S_{m-1}을 사용한다. 최상위 상태는 상태들의 이진 부호화된 값 중 가장 큰 값을 갖는 상태를 의미하고 최하위 상태는 이진 부호화된 값 중 가장 작은 값, 즉 $0_{(2)}$값을 갖는 상태를 의미한다. 이 순차회로가 $2^{n-1}+1 \sim 2^n$ 개의 상태를 갖는 경우에는 이 상태를 구현하기 위해 최소 n 비트의 기억 소자를 가져야만 한다. 예를 들어, $5(2^{3-1}+1)$개 이상 $8(2^3)$개 이하의 상태를 사용하는 상태 머신의 경우 3비트의 기억소자를 필요로 한다. <그림 2>는 S_0부터 S_5까지의 6개 상태만을 사용하는 상태 머신으로 3개의 기억 소자를 사용하며 2개의 정의되지 않은 상태(S_6, S_7)를 갖는 예를 보여주고 있다.

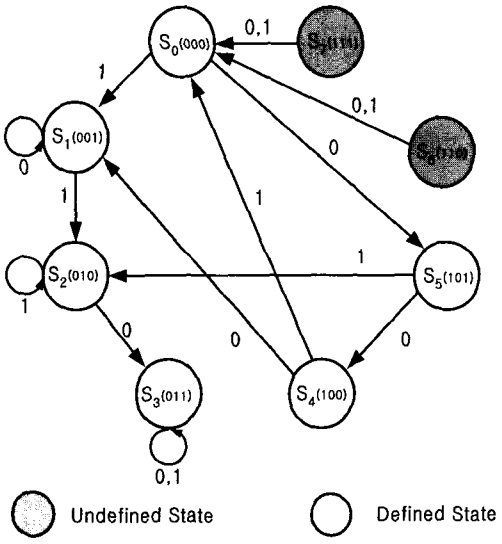


그림 2. 상태 머신의 예
Fig. 2. Example of Finite State Machine.

2. 제안된 비주사 DFT 기법

본 논문에서 제안하는 비주사 DFT 기법은 제어회로가 [조건 1]을 만족하는 경우 기존의 주사, 비주사 기법들에 비해 최소한의 면적 오버헤드로 완벽한 고장 효율을 보장한다.

[조건 1] 1보다 큰 자연수 m개의 정의된 상태를 가지는 제어 회로를 가정할 때, 기억 소자의 개수 $n = \lceil \log m \rceil$ 이며 최상위 상태의 이진값은 $2^{n-1} \le B(S_{m-1}) \le 2^n-1$ 에 속한다.

```
Procedure Non-Scan_Controller (Circuit C)
{
    // 주어진 순차회로 모델을 조합회로 모델로 변경
    C = Make_Combinational_Circuit_Model(S);
    // 조합회로에 대한 ATPG 수행
    TP = Test_Pattern_Generation (C);
    // 제안된 비주사 DFT 기법의 적용
    S' = Apply_Non-Scan_DFT_Method (S);
    // 제안된 테스트 패턴 재배열 알고리즘
    RTP = Arranging_Test_Pattern (TP);
    return (S', RTP);
}
```

그림 3. 제안된 제어 회로를 위한 비주사 DFT 기법 절차
Fig. 3. Proposed Non-scan DFT technique precedure for Sequential Circuit.

예를 들어 5개 이상 8개 이하의 상태를 가지는 제어 회로의 경우, m은 최소한 3비트의 기억소자가 할당되어야 한다. <그림 2>의 상태 머신은 3비트의 기억 소자를 가지고 있으며 정의 가능한 8가지 상태 중에서 S_0부터 S_5까지, 즉 6개의 상태만을 정의하여 설계한 예를 보여주고 있다. <그림 3>은 본 논문에서 제안하는 비주사 DFT 기법의 전체 절차를 보여주고 있다. 3장에서 제안된 비주사 DFT 기법의 각 단계에 대해 상세하게 알아본다.

Ⅲ. 비주사 DFT 기법 절차

Ⅲ장에서는 본 논문에서 제안한 제어 회로를 위한 비주사 DFT 기법의 절차에 대해 설명한다. 즉 테스트 패턴 생성, DFT 하드웨어의 추가 여부의 결정, 생성된 테스트 패턴의 인가 시간을 최소화 하기 위한 테스트 패턴의 재배열에 대해 기술한다.

1. 조합 회로 모델을 위한 테스트 패턴 생성

제어 회로로부터 조합 회로 모델을 생성하고 생성된 조합 회로 모델을 위한 테스트 패턴을 생성한다. <표 1>은 <그림 2>의 상태 머신을 구현한 회로에 대해 조합 회로 모델을 생성하고 테스트 패턴을 생성한 결과와 그에 따른 테스트 응답을 보여준다. 조합 회로 모델로부터 생성된 테스트 패턴은 순차 회로의 주입력 값과 순차 회로 기억 소자들의 현재 상태값으로 구성되어진다. 그리고 테스트 응답은 순차 회로의 주출력과 다음 상태 값에 해당하는 기억 소자의 입력값으로 구성되어진다. <표 1>의 음영 처리된 부분의 현재 상태 값은 <그림 2>의 상태 머신에 정의되어 있지 않은 상태를 나타낸다.

표 1. 생성된 조합회로 모델을 위한 테스트 패턴과 테스트 응답
Table 1. Test Pattern and Test Response for the Generated Combinational Circuit Model.

테스트 패턴		테스트 응답	
현재상태	주입력	다음상태	주출력
S_1(001)	1	S_2(010)	01
S_5(101)	0	S_4(100)	11
S_0(000)	0	S_5(101)	10
S_4(100)	0	S_1(001)	01
S_2(010)	0	S_3(011)	10
S_3(011)	0	S_3(011)	10
S_5(101)	1	S_2(010)	11
S_7(111)	**1**	S_0(000)	01
S_6(110)	**0**	S_0(000)	11

2. 제어 회로를 위한 비주사 DFT 하드웨어 추가

본 논문에서 제안한 방법은 완벽한 고장 효율을 보장하는 테스트 패턴을 생성하기 위해 제어 회로 기억 소자의 입력과 출력을 각각 회로의 출력과 입력으로 가정한 조합회로 모델에 대한 테스트 패턴을 생성하였다. 제어 회로의 완벽한 고장 효율을 보장하기 위해서는 테스트 모드에서 조합회로 모델로부터 생성된 모든 테스트 패턴이 인가되어야만 한다. 이를 위해 테스트 패턴에 주어진 상태로 제어 회로의 상태를 이동하고, 테스트 패턴에 나타난 주입력을 인가하여야 한다. 그러나 제어 회로의 현재 상태에 해당하는 기억 소자의 출력을 회로의 입력으로 가정하고 테스트 패턴을 생성하였기 때문에 순차회로에 정의되지 않은 상태를 포함하는 테스트 패턴이 생성될 수 있다. 이러한 경우 순차회로의 기능적인 상태 이동만을 이용해서는 모든 테스트 패턴의 인가가 불가능하다. 따라서 본 논문에서는 정의되지 않은 상태를 포함하는 테스트 패턴을 인가하기 위해 적은 면적 오버헤드만으로 정의되지 않은 상태로 이동할 수 있는 기법을 제안하며 구현 방법은 다음과 같다. n 비트로 표현할 수 있는 모든 상태의 이진값은 $(0 \sim 2^n-1)$의 범위를 갖고, 이들은 다시 최상위 비트가 0인지와 1인지에 따라 $(0 \sim 2^{n-1}-1)$과 $(2^{n-1} \sim 2^n-1)$의 범위를 갖는 두 그룹, B_L과 B_H로 나눌 수 있다. 각각의 그룹에는 최상위 비트(most significant bit)를 제외한 나머지 비트들이 서로 일치하는 각각의 쌍들이 존재한다. 이 때, [조건 1]을 만족하는 회로의 경우 $2^n \cdot m$ 개의 정의되지 않은 상태들은 최상위 비트가 1인 그룹 B_H에 속하게 된다. 따라서 정의되지 않은 상태들은 각각의 쌍인 정의된 상태의 최상위 비트만 반전시켜줌으로써 만들어낼 수 있다. <그림 4>는 제안된 비주사 DFT 기법의 구조를 보여준다.

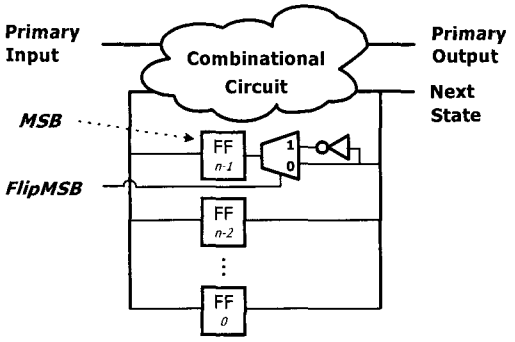


그림 4. 제안된 비주사 DFT 기법의 구조
Fig. 4. Proposed Structure of Non-scan DFT technique.

예를 들어 <표 1>의 경우 <그림 2>의 순차회로에서 정의하지 않은 상태 S_6(110)과 S_7(111)이 테스트 패턴에 포함되어 있다. 이러한 테스트 패턴은 제어 회로의 상태 천이를 이용해서 생성할 수 없는 테스트 패턴이다. 그러나 상태 S_6(110)은 상태 S_2(010)의 최상위 비트를, 상태 S_7(111)은 상태 S_3(011)의 최상위 비트를 반전시킴으로써 생성할 수 있다. <그림 5>는 정의되지 않은 상태를 생성키 위해 최상위 비트를 반전시키는 과정의 시뮬레이션 파형을 보여준다. 예를 들어 정의되지 않은 상태인 S_7(111)에 포함된 테스트 패턴을 제어 회

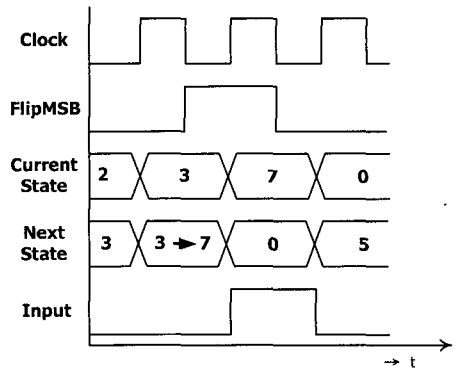


그림 5. 정의되지 않은 상태 생성
Fig. 5. Generation of Undefined States.

로에 인가하기 위해서는 제어 회로의 상태를 $S_3(011)$으로 천이시키면서 최상위 비트를 반전시키면 된다.

3. 테스트 패턴의 재배열과정

현재 조합회로 모델에 의해 생성된 테스트 패턴은 ATPG 툴에 의해 무작위로 추출된 상태이다. 본 논문에서는 테스트 패턴을 순차회로에 인가하는 시간을 최소화 시킬 수 있도록 테스트 패턴에 순차회로에서 정의된 상태 천이도를 반영시킨다. 이는 현재의 테스트 패턴을 재배열함으로써 가능하며 계략적인 테스트 패턴 재배열 알고리즘의 의사코드가 <그림 6>에 나타나 있다. 우선 주어진 테스트 패턴 셋을 정렬한다. 정렬은 테스트 패턴의 현재상태를 오름차순으로 정렬한 후 각 현재상태 그룹 내에서 다시 테스트 응답의 다음상태를 오름차순으로 정렬한다. 단, 현재상태와 다음상태가 같은 테스트 패턴이 각 그룹의 선두에 오도록 정렬한다. 또한 앞, 뒤 패턴의 상태 천이가 연결되지 않는 경우에는 원하는 상태로 이동하기 위한 의사(psuedo) 패턴을 삽입함으로써 완벽한 테스트 인가를 가능하게 하였다. <그림 7>은 조합회로 모델에 의해서 생성된 테스트 패턴을 <그림 2>의 상태 천이도를 이용하여 테스트 패턴을 재배열하는 과정을 보여주고 있다. <그림 7>에서 음영은 들어간 노드는 상태 머신에 정의되지 않은 상태를 의미한다. <그림 7>의 경우에 $S_0 \rightarrow S_5 \rightarrow S_2 \rightarrow S_3 \rightarrow (S_3)S_7 \rightarrow S_0 \rightarrow S_5 \rightarrow S_4 \rightarrow S_1 \rightarrow (S_2)S_6$ 순서로

```
Procedure Arrange_Test_Patterns(Test_Pattern_Set TP)
{
    RTP = NULL; //정렬이 완료된 테스트 패턴 셋 저장소 초기화
    STP = Sort_Test_Pattern_Set (TP); //테스트 패턴 셋 정렬
    while (STP != NULL) // 인가될 테스트 패턴이 존재하는 동안
    {
        Select_a_Test_Pattern (Random_Value, STP) ; // 시작 테스트 패턴 P 선택
        Search_State_Transition (CS); // P에 포함된 현재상태까지의 상태천이도탐색
        Place_Dummy_Patterns_in_RTP (); //필요한경우 의사패턴 삽입
        Push (P, RTP); Pop (P, STP); SS = NS;
        while (1)
        {   // P의 다음상태와 동일한 현재상태를 포함하는
            //테스트 패턴의 순차적 선택
            Select_a_Test_Pattern (SS, STP) ;
            if (SS== CS) Push (P, RTP); Pop (P, STP); SS=NS;
            else break;
        }
    }
    return RTP; // 재배열 과정 종료
}

Procedure Select_a_Test_Pattern(int Group, Test_Pattern_Set STP)
{
    P = STP[Group][Top];
    CS = Current_State_of_P; // 선택된 패턴의 현재 상태
    NS = Next_State_of_P; // 선택된 패턴의 다음 상태
}
```

그림 6. 테스트 패턴 재배열 알고리즘
Fig. 6. Test Pattern Reordering Algorithm.

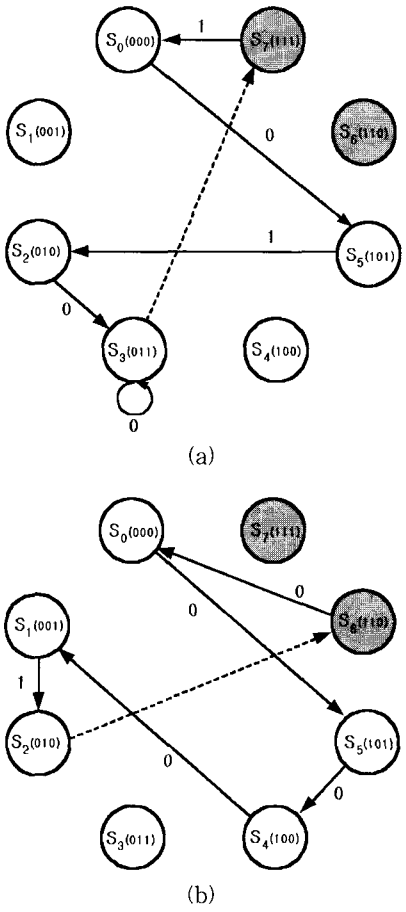


그림 7. 테스트 패턴 재배열 과정의 상태천이도
Fig. 7. State Transitino Diagram for Test Pattern Reordering.

표 2. 재배열 과정을 완료한 순차회로를 위한 테스트 패턴
Table 2. Test Patterns after Reordering Procedure.

현재상태	테스트 패턴			다음상태	테스트 응답
	주입력	FlipMSB	Reset		주출력
XXX	x	0	1	000	00
000	0	0	0	101	10
101	1	0	0	010	11
010	0	0	0	011	10
011	0	1	0	111	10
111	1	0	0	000	01
000	0	0	0	101	10
101	0	0	0	100	11
100	0	0	0	001	01
001	1	1	0	110	01
110	0	0	0	000	11

생성된 테스트 패턴을 재배열함으로써 완벽한 고장 효율을 보장하는 테스트 패턴을 모두 순차회로에 인가할 수 있다. 밑줄이 그어진 상태는 원하는 상태로 이동하기 위해 추가된 의사 테스트 패턴을 의미한다. $(S_3)S_7$은 FlipMSB라인을 활성화시켜 S_3에서 S_7로 상태가 천이한 것을 의미한다.

<표 2>는 <표 1>에서 나타난 테스트 패턴으로부터 재배열 과정을 수행한 테스트 패턴을 보여준다. 음영이 들어간 부분은 제안된 방법에 의해 정의되지 않은 상태 S_6과 S_7로 상태가 천이하는 부분과 제어 회로의 상태 천이를 이용하여 원하는 상태로 이동하기 위하여 추가된 테스트 패턴을 의미한다.

Ⅳ. 실험 결과

본 논문에서 제안한 방법의 성능을 검증하기 위해 MCNC'91 FSM 벤치마크 회로들을 이용하여 실험을 수행하였다. 벤치마크 회로들은 Synopsys를 사용하여 합성하였다[10]. 그리고 조합 회로 모델에 대한 테스트 패턴의 생성 및 최종 테스트 패턴에 대한 제어 회로의 고장 시뮬레이션은 Syntest를 이용하여 수행하였다[11]. <표 3>은 실험에 사용된 회로의 특징을 보여준다. <표 3>의 면적의 단위는 Synopsys의 Design Analyzer 툴의 결과리포트에서 사용되는 2-입력 NAND 게이트의 수이다.

<표 4>는 테스트 패턴을 순차회로에 인가하는 데 필요한 총 클럭 사이클의 수를 보여주고 있다. 완전 주사 기법의 경우 1개의 스캔 체인만을 가정하여 합성하였으므로 클럭 수는 "*테스트 패턴의 수* * (*기억 소자의 수* + 1) + 기억 소자의 수" 에 의하여 계산될 수 있다. [8]과 [9]는 [9]에서 언급한 데로 각각 "*테스트 패턴의 개수* * 2 + 1", "*정의된 상태에 도달하는 클럭 수* + *테스트 패턴에 포함된 정의되지 않은 상태의 수* + *테스트 패턴의 수* +2"로 계산될 수 있다. 또한 제안된 기법은 "*테스트 패턴에 포함된 상태에 도달하는 클럭 수* + *테스트 패턴의 수* + 1"로 계산될 수 있다. <표 4>의 "회로명" 열에서 심볼 "*"는 주입력 핀의 수가 플립플롭의 개수보다 같거나 크다는 것을 뜻한다. [8]의 기법은 회로의 주입력 핀의 수가 플립플롭의 개수보다 같거나 많을 경우 테스트 패턴이 포함하는 모든 상태를 직접 생성할 수 있고 [9]의 기법은 추가로 설계된 제어 회로에서 정의되지 않은 상태를 직접 생성시켜 주고, 또한 그 상태를 유지시킬 수 있는 플립플롭을 추가했기 때문에 상태에 도달하는 클럭 수가 적게 나타났다.

표 3. 벤치마크 회로의 특성
Table 3. Characteristics of Benchma가 Circuits.

회로명	면적	주입력핀 개수	주출력 핀개수	플립플롭 수	정의된 상태의 수
bbtas	41.5	2	2	3	6
bbara	89	4	2	4	10
beecount	53.5	3	4	3	7
ex1	305	9	19	5	20
ex2	147	2	2	5	19
ex3	93	2	2	4	10
ex4	88.5	6	9	4	14
ex5	85	2	2	4	9
dk16	203.5	2	3	5	7
dk27	43	1	2	3	27
dk512	87	1	3	4	7
opus	104.5	5	6	4	10
planet	418.5	7	19	6	48
pma	206	8	8	5	24
s1	277.5	8	6	5	20
s27	36	4	1	3	6
s1488	420	8	19	6	48
s1494	419	8	19	6	48
styr	357.5	9	10	5	30

표 4. 테스트 패턴 적용 시간
Table 4. Test Pattern Application Time.

회로명	완전주사기법	[8]	[9]	제안된 기법
bbtas	55	93	21	40
*bbara	194	77	50	85
*beecount	87	43	26	36
*ex1	461	153	88	220
ex2	311	965	78	100
ex3	184	432	53	92
*ex4	129	51	33	121
ex5	154	341	44	82
dk16	359	952	81	85
dk27	67	159	23	24
dk512	124	331	39	41
*opus	154	61	41	73
*planet	769	219	197	506
*pma	492	163	103	262
*s1	539	179	113	160
*s27	47	23	14	16
*s1488	755	215	150	292
*s1494	741	211	156	289
*styr	713	237	152	238

본 논문에서 제안한 기법과 기존 기법들과의 면적 오버헤드를 비교한 결과의 백분율 그래프가 <그림 8>에 나타나 있다. [8]에서 제안된 기법은 완전 주사 기법보다 오버헤드가 적지만 [9]에서 제안된 기법은 주입력이 플립플롭의 수보다 적을 경우 정의되지 않은 상태를 생성하기 위한 추가회로로 인해 완전 주사 기법보다 훨씬 많은 오버헤드를 갖는 경우도 있음을 알 수 있다. 제안된 기법은 어떠한 회로에도 멀티플렉서와 인버터가 각각 1개만이 추가되므로 면적 오버헤드를 최소화하였다.

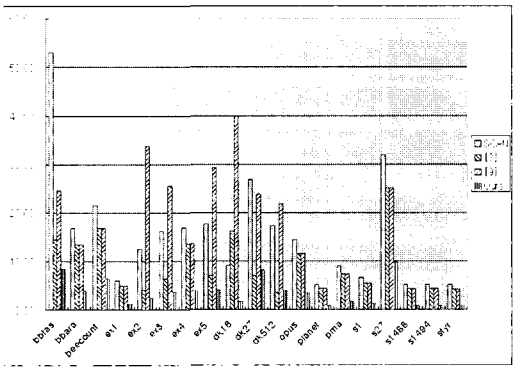


그림 8. 면적 오버헤드 비교
Fig. 8. Area Overhead Comparison.

V. 결 론

본 논문에서는 순차회로를 위한 효율적인 비주사 DFT 기법을 제안하였다. 일반적으로 순차회로를 위한 테스트 패턴 생성 시에는 순차회로를 위한 테스트 패턴 생성 알고리즘을 적용하여야 하지만 순차회로를 위한 테스트 패턴 생성 알고리즘은 패턴 생성에 오랜 시간이 소요되고 완벽한 고장 효율을 항상 보장할 수는 없다. 본 논문에서 제안한 기법은 순차회로에 대해 조합회로 모델을 생성하여 조합회로에 대한 테스트 패턴 생성 알고리즘을 적용할 수 있게 함에 따라, 완벽한 고장 효율을 보장하는 테스트 패턴을 빠른 시간 안에 생성할 수 있게 하였다. 그리고 생성된 테스트 패턴을 순차회로에 인가하기 위하여 기존의 방법들보다 적은 면적의 하드웨어만을 추가하여 면적 오버헤드를 최소화하였고 테스트 패턴의 재배열과정을 거쳐 조합회로 모델에 대해 생성된 테스트 패턴을 순차회로에 짧은 시간으로 인가할 수 있도록 하였다. 본 논문에서 제안된 기법은 완전 주사 기법 및 기존의 기법들에 비해 적은 오버헤드를 갖고 완벽한 고장 효율을 보장하는 테스트 패턴 생성을 보장한다.

참 고 문 헌

[1] 홍성제, 박은세, 강성호, 최호용, 장 훈, 테스팅 및 테스팅을 고려한 설계, 홍릉과학 출판사, 1998

[2] S. Bhattacharya and S. Dey, "H-SCAN: A High Level Alternative to Full-Scan Testing With Reduced Area and Test Application Overheads," VLSI Test Symposium, pp. 74-80, 1996

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

[4] S. Ohtake, H. Wada, T. Masuzawa and H. Fujiwara, "A non-scan DFT method at register-transfer level to achieve complete fault efficiency," ASP-Design Automation Conference, 2000.

[5] M. Abramovici, M. A. Breuer and A. D Friedman, Digital system testing and testable design, Computer Science Press, 1990.

[6] P. C. Maxwell, R. C. Aitken, V. Johansen and I. Chiang, "The effect of different test sets on quality level prediction : when is 80% better than 90% ?," International Test Conference, pp. 358, 1991.

[7] Benoit Nadeau-Dostie, Design For At-speed Test, Diagnosis and Measurement. Kluwer Academic Publishers, 2000.

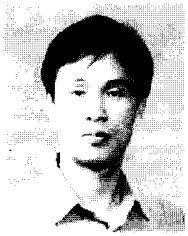
[8] V. Chickermane, E. Rudnick, P. Banerjee and J. H. Patel, "Non-scan design-for-testability techniques for sequential circuits," ACM/IEEE Design Automation Conference, pp. 236-241, 1993.

[9] S. Ohtake, T. Masuzawa and H. Fujiwara, "A non-scan DFT method for controllers to achieve complete fault efficiency," Asian Test Symposium, pp. 204-211, 1998.

[10] (V)HDL Compiler Reference Manual, Version 3.4a, Synopsys Inc. 1996.

[11] Syntest User's Guide, Syntest Inc.

저 자 소 개



沈 載 憲(正會員)

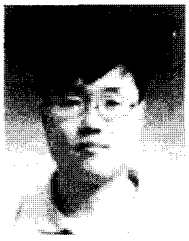
1987년 : 서울대학교 물리교육과 졸업(학사). 1989년 : 서울대학교 물리교육과 졸업(석사). 2000년~현재 : 숭실대학교 대학원 컴퓨터학과 박사과정. <주관심분야 : VLSI 설계 및 테스트, 컴퓨터구조, VLSI CAD>

金 文 俊(正會員)

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>

朴 在 興(正會員)

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>



梁 善 雄(正會員)

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>

張 勳(正會員)

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>