特輯

Sequential 회로를 위한 ATPG

金 錫 源, 趙 敬 淳, 金 光 鉉
三星電子(株) ASIC 事業部

I. 서론

최근 VLSI 기술의 발달에 따라 회로의 규모가 커질뿐 아니라 그 복잡도 역시 급격히 증가하고 있다. 따라서 효과적인 검사가 이루어지지 않을 경우 출하된 chip이 불량품일 확률이 높아진다. 효과적인 검사가 이루어지기 위해서는 chip의 결함을 추출할 수 있는 양질의 test pattern이 필요하다. Test pattern의 질은 보통 fault coverage에 의해 표현되는데 fault coverage란 주어진 test pattern에 의해 detect된 fault의 비율을 나타낸다. 검사에 합격한 chip이 불량품일 확률을 reject ratio라 한다. 이 reject ratio는 chip의 질에 대한 신뢰도의 기준이

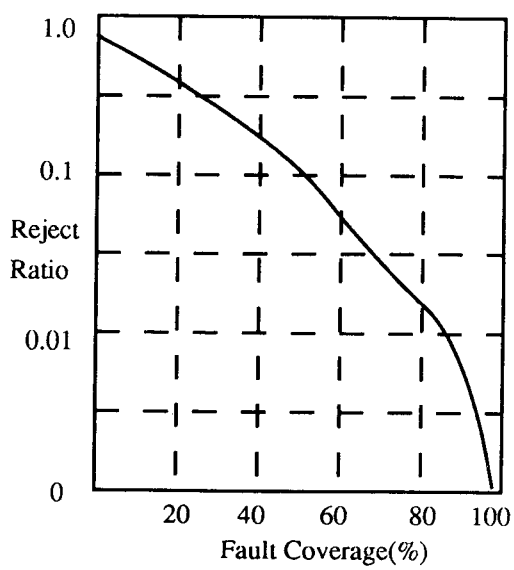


그림 1. Reject ratio versus fault coverage

되며, reject ratio는 fault coverage와 yield의 함수이다. 주어진 yield에 대한 reject ratio와 fault coverage 사이의 상관관계를 도표로 나타내면 그림 1과 같다. 이 그림에 나타나 있듯이 높은 fault coverage를 갖는 test pattern에 의해 검사된 chip은 그 만큼 reject ratio가 낮아진다. [1]

Test pattern은 여러가지 방법으로 생성할 수 있다. 첫째는 수동으로 test pattern을 생성하는 것이다. Test pattern을 생성하여 이전 test pattern이 detect하지 못한 fault를 fault simulation을 통하여 detect해 보는 것이다. 그러나 이 방법은 매우 힘들고 시간이 많이 소요된다.

두번째는 회로 스스로가 test pattern을 생성하게 하는 것이다. Test pattern을 생성하는 logic을 회로에 추가하는 것으로 소위 BIST(built-in self test)라 한다. 그러나 회로에 부가적인 circuitry를 추가하게 됨에 따라 chip의 크기가 커지고 속도가 느려지게 된다.

세번째는 자동으로 test pattern을 생성하는 것이다. 회로가 주어지면 ATPG(automatic test pattern generation) program이 test pattern을 구해준다. ATPG는 회로에 부가적인 circuitry를 추가할 필요가 없을 뿐만 아니라, 높은 fault coverage를 갖는 test pattern을 대부분의 경우 빠른 시간 내에 구해준다. 그러나 모든 회로에 대하여 항상 실용적이지는 못하다. 왜냐하면 complete한 ATPG algorithm(주어진 fault가 detect 가능한 경우 반드시 test pattern을 구해줌)이라 하더라도 주어진 fault를 detect하는 test pattern을 구하는데 지나치게 많은 CPU time을 소비하는 경우가 있기 때문이다. 특히 combinational 회로보다는

sequential 회로에 대하여 비실용적일 가능성이 높다. 따라서 sequential ATPG가 비실용적일 경우 이에 대한 보완책으로, full scan design technique을 이용하여 sequential 회로를 등가의 combinational 회로로 변환한 후 combinational ATPG를 적용한다. 즉 회로내의 모든 flip-flop을 하나 또는 몇 개의 shift register로 연결한 뒤, combinational logic에 대한 test pattern은 combinational ATPG를 이용하여 생성하며 flip-flop의 value는 이 shift register를 통하여 sacn in, scan out 한다. 이 방법은 비교적 실용적으로 industry에서 많이 사용되고 있으나, BIST와 마찬가지로 회로에 대한 overhead가 존재함으로써 chip의 크기가 커지고 속도가 느려지게 되는 단점이 있다. 최근에는 full scan design technique을 개선하여 sequential 회로를 완전히 combinational 회로로 변환하지 않고, 단순히 sequential depth만 감소시킨 뒤 sequential ATPG를 이용하는 partial scan design technique이 확산되고 있는 추세이다. 이 방법은 기존의 full scan design technique에 비하여 회로에 대한 overhead가 작을 뿐만 아니라 sequential ATPG의 난점을 보완해 줌으로, 크게 주목을 받고 있다.

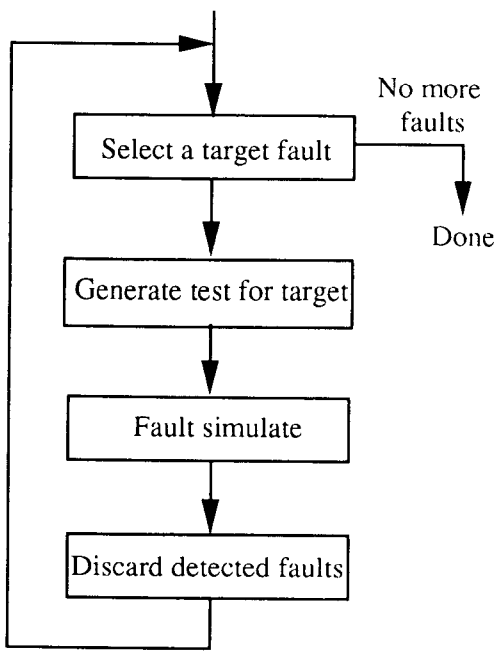


그림 2. Structure of ATPG system

ATPG는 fault simulator와 TPG(test pattern generator)로 구성되어 있다(그림 2 참고). TPG는 주어진 target fault를 detect하는 test pattern을 생성하며, fault simulator는 이 test pattern이 detect할 수 있는 다른 모든 fault를 찾아낸다. 이와 같이 하여 TPG에 주어지는 fault의 수는 현저하게 감소한다. 이 글에서는 gate level의 single stuck-at fault model 하에서 sequential 회로를 위한 ATPG에 대하여 논하려 한다. Ⅱ장에서는 Fault model과 fault simulator에 대하여 간략하게 언급하고, Ⅲ장에서는 TPG(test pattern generation) algorithm에 대하여 살펴보겠다. 그리고 Ⅳ장에서는 우리가 구현한 ATPG에 대한 구현 방법 및 실험 결과를 소개하고 마지막으로 Ⅴ장에서는 결론을 기술한다.

Ⅱ. Fault Model과 Fault Simulator

이 장에서는 fault model과 fault simulator에 대하여 기술한다. Fault simulation과 TPG을 위해서는 먼저 fault set이 정의되어야 하는 데, fault model에 따라 다양한 fault set이 존재할 수 있다. Fault simulator는 주어진 test pattern에 의해 어느 fault가 detect되었는가를 결정하여 fault coverage를 구해준다.

1. Fault Model

Fault란 회로에 존재하는 physical defect에 대한 logical model이다. Physical defect란 공정 상의 잘못으로 인해 회로 내에 존재하는 결함으로, 예를 들면 metal line의 open과 short가 있다. 회로 내에는 다양한 physical defect가 존재하며, 이 physical defect는 회로의 오동작을 야기시킨다. Physical defect는 다양하게 존재하므로 직접 analysis 하는 데는 많은 어려움이 따른다. 따라서 logical fault model이 필요한 것이다. 다양한 fault model이 존재할 수 있으며, 어느 fault model을 사용할 것인가를 결정하기 위해서는 다음의 두 가지 측면이 고려되어야 한다. 첫째는 fault model이 얼마나 physical defect를 잘 반영하는가이다. 둘째는 fault model에 의해 주어진 fault set의 크기가 적

당한가이다.

Fault level에 따른 fault model로서는 switch level fault, gate level fault, 그리고 high level fault가 있다. Switch level fault란 MOS transistor 상에서의 stuck-open, stuck-short를 말한다. Switch level fault model은 회로의 physical defect를 잘 반영한다는 장점을 갖고 있지만, fault set이 크다는 단점이 있다. Fault model은 회로의 model과 깊은 관계가 있으며, digital 회로는 보통 logic gate들의 연결상태로 기술된다. 따라서 logic gate들의 연결상태 만을 고려한 gate level의 line stuck-at fault model이 있다. 회로의 모든 line은 두 종류의 stuck-at fault(stuck-at 0, stuck-at 1)를 갖는다. Stuck-at 0(1) fault란 line이 logic 0(1)에 고정(stuck)된 것을 의미한다. Gate level의 stuck-at fault model은 회로의 physical defect도 잘 반영하고 fault set의 크기도 적당하기 때문에 가장 많이 사용된다. High level fault는 gate level의 line stuck-at 개념을 RTL(register transfer language)에서 변수의 stuck-at 개념으로 확장한 model로서, fault set의 크기가 작다는 장점이 있지만, 회로의 physical defect는 잘 반영하지 못하는 단점이 있다.

Fault 수에 따른 fault model로서는 single fault model과 multiple fault model이 있다. 일반적으로 여러 개의 fault가 회로 내에 동시에 존재한다. 따라서 n 개의 line을 갖는 회로의 경우 3^n-1 개의 line stuck-at fault 조합이 가능하다. 이러한 fault model을 multiple fault model이라 한다. Single fault model은 회로 내에 오직 하나의 fault만이 존재한다고 가정한다. 따라서 n 개의 line을 갖는 회로의 경우 2n 개의 line stuck-at fault가 있다. Multiple fault model은 비록 n이 작은 수이라 하더라도 fault set의 크기가 상당함을 알 수 있다. 예를 들어 n이 100인 경우 single fault 수는 200이지만, multiple fault 수는 10^{47}이나 된다. 따라서 multiple fault model은 실용적이지 못하다. Single fault를 detect하는 test pattern은 대부분의 multiple fault를 detect한다고 한다. [2] 따라서 single fault model이 주로 사용된다.

일반적으로 industry에서는 gate level의 single stuck-at fault model을 사용한다. 이는 이 fault model이 회로의 physical defect도 잘 반영하면서 fault set의 크기도 적당하기 때문이다. 지금부터 이 글에서 Fault라 함은 gate level의 single stuck-at fault를 의미한다.

2. Fault Simulator

Fault simulator는 주어진 test pattern을 good machine과 faulty machine에 인가한다. 여기서 good machine이란 fault-free 회로를 말하며, faulty machine이란 fault가 존재하는 회로를 말한다. 어떤 faulty machine의 output value가 good machine의 output value와 다를 때, 그 faulty machine의 fault는 detect되었다고 한다.

표 1은 fault simulator의 task를 나타낸다. 각 행은 test pattern에, 각 열은 machine에 대응된다. n 개의 test pattern과 (m+1) 개의 machine (하나의 good machine과 각기 다른 fault에 해당하는 m개의 faulty machine), 그리고 (m+1)n 개 machine의 value가 나타나 있다. Fault simulator의 task는 (m+1)n 개 machine의 output value를 찾고, 어떤 faulty machine의 output value가 good machine의 output value와 다른 지를 결정한다. 이 table을 어떤 order에 의해 채우느냐에 따라 다양한 fault simulator의 algorithm이 개발되어 왔는데, 그 대표적인 것이 concurrent fault simulation [3] 이다.

표 1. Task of fault simultor

	V_1	...	V_j	...	V_n
Good	G_1	...	G_j	...	G_n
$Fault_1$	$F_{1,1}$	...	$F_{1,j}$	...	$F_{1,n}$
.	.	...	.	...	.
.	.	...	.	...	.
$Fault_i$	$F_{i,1}$	...	$F_{i,j}$	...	$F_{i,n}$
.	.	...	.	...	.
.	.	...	.	...	.
$Fault_m$	$F_{m,1}$	...	$F_{m,j}$	...	$F_{m,n}$

Concurrent fault simulation은 good machine과 faulty machine을 동시에 Simulation하는 방법으로서, 앞에서 설명한 table을 왼쪽에서부터 오른쪽으로 채워나간다. Concurrent fault simulation은 각각의 fault에 해당하는 faulty machine을 따로 Simulation하기 때문에 불필요한 event가 발생하지

않는다. 따라서 발생하는 event의 수가 적기 때문에 기존의 다른 fault simulation algorithm보다 빠르다. 하지만 각 gate는 active faulty machine의 faulty vlue를 저장하고 있어야하기 때문에 많은 memory 사용을 요구한다.

Concurrent fault simulator처럼 event 발생은 최소화하고, memory 사용은 적게 한 것이 PROOFS [4] fault simulation algorithm이다. PROOFS는 good machine의 value와 state node(flip-flop이나 latch)의 faulty value 만을 이용하여 fault simulation을 할 수 있기 때문에 사용되는 memory의 양이 현저하게 적다. Primary output으로부터 depth-first search하여 ordering된 fault list로부터 32 개의 fault를 한 group으로 injection하여 parallel [5] 하게 simulation한다. 아래 그림 3과 같은 inactive fault를 초기에 identify하여 injection하지 않음으로써 45% 정도의 fault simulation 시간을 감소시켰다. ISCAS89 benchmark 회로에 대한 실험 결과에 의하면 PROOFS는 concurrent fault simulation보다 6 - 67 배가 빠르고 memory 사용은 평균 6 배가 적다. [4]

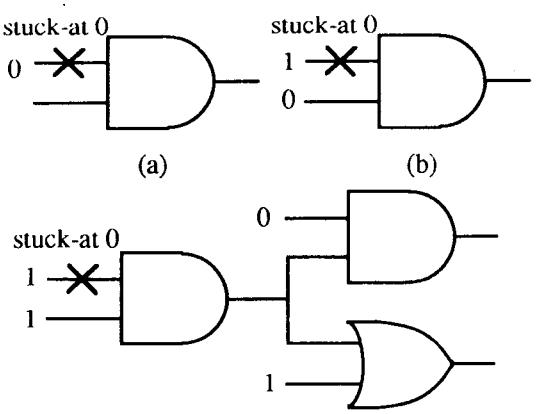


그림 3. Inactive fault classes
(a) Unexcited fault
(b) Excited fault but not propagated 1 level
(c) Excited fault but not propagated 2 levels

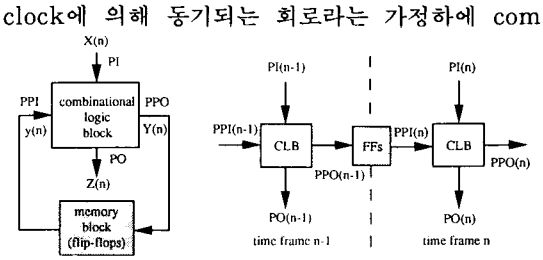
PROOFS가 synchronous sequential 회로에만 적용되는 데 반하여, general sequential 회로에 적용가능하도록 한 것이 SuperFault fault simulator이다. 삼성전자(주) ASIC사업부의 in-house fault simulator인 SuperFault는 PROOFS의 algorithm에 기초하여 개발되었으며, asynchronous sequential 회로에 적용 가능하도록 새로운 cell modeling 방법을 도입하였다. 그리고 logic simulation시 발생한 event를 analysis하여 event가 발생하지 않은 gate가 속하는 영역의 fault는 injection하지 않음으로써 fault simulation시 event 발생을 최소화하였다. 또한 한번 potential detect되었던 fault들을 함께 grouping하여 32 bit를 효과적으로 사용함으로써 event 발생을 17% 정도 감소시켰다. 또한 fault propagation을 위하여 multiple LIFO(last in first out) stack을 사용함으로써 priority queue를 사용할 때의 복잡도 O(n log n)을 O(n)으로 감소시켰다. [6] 여기서 n은 fault propagation 영역에 속하는 gate의 수를 나타낸다.

Ⅲ. TPG(test pattern generation) algorithm

이 장에서는 sequential 회로를 위한 ATPG의 기본 개념과 현재 실용적으로 사용되는 대표적인 sequential ATPG인 Gentest [7] 와 HITEC [8] 의 algorithm에 대하여 살펴보겠다.

1. Background

Seuqential TPG algorithm은 다음의 두 model을 사용하여 combintional TPG algorithm- D-algorithm [9], 또는 PODEM [10] -을 확장 적용함으로써 가능하게 되었다. 두 model은 sequential 회로 model과 iterative logic array model이다. Sequential 회로 model이란 sequential 회로를 clock에 의해 동기되는 회로라는 가정하에 com-



(a) Sequential 회로 model (b) Iterative logic array model

그림 4. Sequential 회로 model과 iterative logic array model

binational logic block과 memory(flip-flop) block으로 구성한다. (그림 4 (a) 참고) Combinational logic block에서 위쪽 input은 primary input, 아래쪽 output은 primary output, 왼쪽 input은 pseudo-primary input, 오른쪽 output은 pseudo-primary output에 해당한다.

그림 4 (b)는 sequential 회로의 time-domain 상에서의 behavior를 space-domain 상의 behavior로 표현한 것이다. 이것을 iterative logic array model이라 하는 데, 이 model을 통하여 sequential 회로를 등가의 combintional 회로로 변환한다. 이 model을 사용함으로써 combinational TPG algorithm을 sequential TPG algorithm으로 확장 적용할 수 있게 되었다.

그러나 아직도 sequential TPG의 어려움은 있다. (그림 5 참고) 첫째는 fault effect가 여러 time frame을 통해 전달된다는 것이다. 따라서 search space가 증가하고 많은 memory 사용을 요구하게 된다. 둘째는 fault effect가 매 time frame의 fault 위치에 나타난다는 것이다. 따라서 single fault model하에서도 multiple fault가 고려되어야 한다.

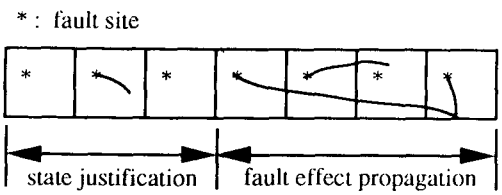


그림 5. Fault effect in iterative logic array model

Test pattern을 생성하기 위하여 fault effect propagation과 state justification이 필요하다. 그림 5는 fault effect propagation을 위해서 4개의 time frame이, state justification을 위해서 3개의 time frame이 사용된 예이다. 이 둘을 어느 방향으로 processing하는가에 따라 forward time processing, reverse time processing이라 한다. HITEC은 fault effect propagation은 forward time processing으로, state justification은 reverse time processing으로 수행한다. 그러나 Gentest는 이 둘을 reverse time processing으로 수행한다.

표 2 Five-valued and nine-valued logic model

5-valued logic model	9-valued logic model
0	0/0
1	1/1
D	1/0
D	0/1
X	X/X
D:good value 1	0/X
faulty value 0	X/0
D:good value 0	1/X
faulty value 1	X/1

Fault effect propagation을 위해서는 sensitized path 구성과 line value justification이 필요하다. Sensitized path란 faulty value에 sensitized된 (good value와 faulty value가 서로 반대인) gate들로 이루어지는 path를 말한다. Fault 위치에서 출발하여 sensitized되기 쉬운 gate들을 선택하여 primary output까지 sensitized path를 만들어 나간다. Search space를 줄이기 위하여 여러 path (multiple path sensitization) 대신 하나의 path (single path sensitization)로 fault effect를 propagation시킨다. 5-valued logic model(표 2 참고)하에서 single path sensitization이 complete하지 않음이 증명되었다. [11] HITEC은 9-valued logic model(표 2 참고)을 사용하며, Gentest는 9-valued logic model의 변형된 형태인 split value model [12] 을 사용한다. Sensitized path를 만들기 위하여 sensitized되기 쉬운 gate를 선택하여 (testability analysis를 이용하여 여러 fanout-out gate 중 하나를 선택) sensitized되지 않은 다른 input에는 non-controlling value를 할당(예를 들어 2-input AND gate의 경우 한 input으로 sensitized value가 propagation되는 경우 다른 input에 1을 할당)한다. Input에 할당된 value를 만족시키기 위하여 그 fanin gate의 input value를 또한 만족시켜야 한다. 이것을 line value justification이라 한다. Sensitized path 구성을 위한 gate 선택시 많은 choice가 있고, line value justification을 위한 input value 할당시에도 많은 choice(예를 들어 OR gate의 output value를 1로

하기 위해서는 어느 한 input이 1이면 된다)가 존재한다. 많은 choice가 존재하기 때문에 선택된 것들에 의해 충돌이 발생할 수 있으며, 이런 경우 이전 상태로 되돌아 가야 한다. 이것을 backtrack이라 한다. Backtrack시의 복원을 위하여 매 선택 단계마다 회로의 status를 저장해야 한다.

Fault effect propagation을 위한 line value justification은 primary input과 state node에 value를 할당하는 것으로 끝난다. 이후에는 state node에 할당된 value를 만족시키기 위한 state justification이 필요하다. State justification은 primary input에 value를 할당하며, state node가 주어진 조건을 만족할 때까지 계속된다. 주어진 조건이란 state node가 initial state(보통 unknown state)이거나 아니면 이미 생성된 test pattern에 의한 마지막 state이어야 한다는 것이다. Test pattern 수를 줄이고 state justification 시간을 적게하기 위하여 두번째 조건 만을 이용한다. State justification이 끝나면 모든 primary input에 value가 할당되고 이것이 주어진 fault를 detect하기 위한 test pattern이 되는 것이다.

2. GenTest

Gentest는 AT&T의 sequential ATPG system으로서, back algorithm [13]과 split value model에 기초하여 만든 sequential TPG와 PROOFS algorithm을 사용한 fault simulator로 구성되어 있다. PROOFS에 대해서는 이미 Ⅱ장에서 언급하였고, 여기에서는 back algorithm과 split value model에 대하여 설명하겠다. Back algorithm과 split value model은 fault effect propagation을 빠르면서 memory 사용이 적도록 한다.

전통적인 fault effect propagation 방법은 fault 위치에서부터 출발하여 primary output까지 sensitized path를 만들어 나가는 forward time processing 방법이다. EBT [14] algorithm은 다음 사실에 기초하여 새로운 fault effect propagation 방법을 고안하였다. Fault가 detect되기 위해서는 반드시 어느 한 primary output에 sensitized value가 나타나야 한다는 사실이다. Topology 정보를 이용하여 sensitized될 path를 미리 선택하고 이 선택된 path의 primary output으로부터 fault 위치로 sensitized path를 만든다. 그러나 topology 정보 만을 이용하여 sensitized path를 선택하기 때문에 잘못된 선택을 하게 되는 경우가 많이 발생할 수 있다. Back algorithm은 fault 위치로 부터 primary output까지 drivability(difficulty to drive fault effect)를 계산하고, drivability에 따라 primary output과 이 primary output의 sensitized value를 선택한다. 그리고 이 primary output으로부터 fault 위치로 sensitized value를 propagation시켜 나간다. 충돌이 발생하면 다른 primary output을 선택하고, 더 이상 선택할 primary output이 존재하지 않는 경우 untestable fault로 처리한다.

Line value justification을 위하여 split value model을 사용한다. Split value model은 9-valued logic model의 한 변형된 형태로서 good machine의 value와 faulty machine의 value를 따로 justify할 수 있도록 한 것이다. 9-valued logic model은 good machine의 value와 faulty machine의 value를 동시에 justify한다. 따라서 어느 한 machine에서 충돌이 발생하면 두 machine 모두 backtrack을 해야하기 때문에 충돌이 발생하지 않은 machine에 대해서는 낭비가 초래된다. Split value model은 good machine과 faulty machine을 서로 독립된 machine으로 가정하여 따로 justify한다. 따라서 good machine과 faulty machine이 서로 다른 testability를 사용할 수 있다. 그러므로 backtrack의 발생을 최소화할 수 있다. 또한 split value model은 good machine과 faulty machine 사이의 relation을 정의하고 이 relation 정보를 이용하여 search space를 줄일 수 있도록 하였다.

3. HITEC

HITEC은 효과적인 state justification 방법과 fault simulation 결과를 이용하여 test pattern을 생성하는 방법을 사용한 최신의 sequential ATPG algorithm이다. State justification을 위하여 good machine의 state 정보와 single frame reverse time processing 방법을 함께 사용한다. HITEC도 fault simulation algorithm으로는 Gentest와 마찬가지로 PROOFS를 사용한다.

State justification을 위하여 이미 생성된 test pattern에 의한 good machine의 state 정보를 유지한다. 현재 justify해야 할 state가 state 정보에 포함되어 있으면 이 state를 만든 test pattern을 적

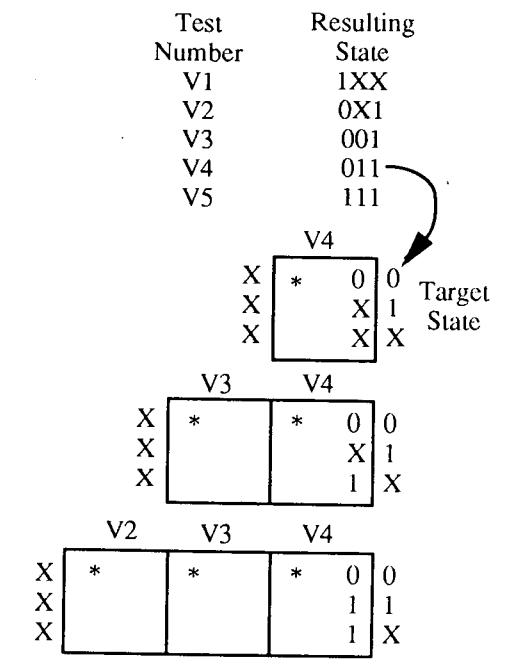
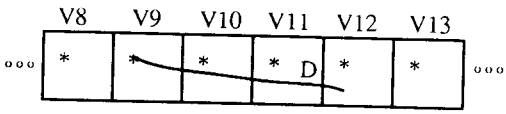
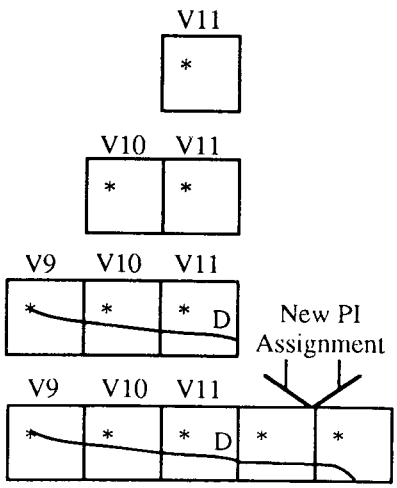


그림 6. Good machine의 state 정보 이용



(a) Fault simulation 결과



(b) Test pattern 생성

그림 7. Fault simulation 결과 이용

용한다. 충돌은 발생하지 않았고 원하는 state가 아니면, 이전 test pattern부터 다시 적용한다. 이 과정을 원하는 state가 될 때까지 계속한다. (그림 6 참고) Justify해야할 state가 state 정보에 없거나 충돌이 발생한 경우에는 한 time frame만 reverse time processing으로 새롭게 justify해야할 state를 찾는다. 그리고 다시 state 정보를 이용한 state justification 방법을 적용한다.

생성된 test pattern에 의한 fault simulation시 어느 한 fault의 faulty value가 state node에 전달된 경우(그림 7 (a) 참고), 먼저 state node에 faulty value가 나타나도록 이전 test pattern을 적용한다. State node에 faulty value가 나타나게 되면, fault 위치가 아닌 이 state node로부터 fault effect를 전달할 수 있도록 primary input에 value를 할당한다. (그림 7 (b) 참고)

Ⅳ. SuperTest

SuperTest는 삼성전자(주) ASIC사업부의 in-house ATPG system으로서 Gentest의 back algorithm과 split value model에 기초하여 개발되었다. SuperTest는 workstation 상에서 C programming language로 구현되었다. Fault simulator로는 in-house fault simulator인 SuperFault를 사용한다. SuperTest는 Gentest의 fault effect propagation 방법을 사용함으로써 fault effect propagation에서는 좋은 성능을 갖추고 있으나, fault effect propagation보다 더 시간이 걸리는 process인 state justification에서는 성능 개선의 여지가 많이 있다. 향후 HITEC의 state justification 방법, fault simulation 결과 이용 방법 등의 도입과 새로운 heuristic 적용을 통한 성능 개선이 요구된다. 하나의 fault를 detect하기 위하여 test pattern을 생성하는 데 많은 CPU time을 요구하기 때문에 multi-phase strategy를 사용한다. 첫번째 phase에서 detect되지 않았던 fault가 다른 fault를 detect하기 위하여 생성된 test pattern에 의해 detect될 수 있기 때문이다. 첫번째 phase에서 CPU time limit을 fault당 2 초로하고, 2 초 내에 주어진 fault를 detect하는 test pattern을 생성하

지 못하는 경우 이 fault에 대한 test pattern 생성을 중단하고, 다음 phase(CPU time limit이 이전 phase의 CPU time limit의 10 배)에서 다시 시도한다.

표 3에 SuperTest에 대한 실험 결과가 나타나 있다. 실험한 회로들은 가상적인 회로가 아니라 제품화된 실제 회로들로서 회로의 이름은 임의의 이름으로 표기하였다. 표 3에서 no. of gates는 회로를 Super-Test primitive로 변환하였을 때의 primitive 갯수를 나타낸다. No. of phase는 test pattern 생성시에 시도한 phase의 수를 나타낸다. Untestable은 검출 불가능으로 판명된 fault 수이고, fc와 eff.는 각각 fault coverage와 efficiency를 나타낸다. Efficiency는 모든 fault에 대하여 detect되었거나 untestable로 판명된 fault의 비를 나타낸다.

표 3. 실험결과

name	no. of gates	no. of faults	no. of phase	no. of pattern	detected	untestable	aborted	fc (%)	eff. (%)	time* (hours)
circuit 1	335	582	3	1083	530	46	6	91.1	99.0	0.40
circuit 2	674	1460	3	9909	1155	98	207	79.1	85.8	8.40
circuit 3	770	1182	3	376	933	200	49	78.9	95.9	3.37
circuit 4	984	1960	2	559	1034	112	814	52.8	58.5	3.44
circuit 5	1101	2220	2	1253	1808	29	383	81.4	82.7	2.16
circuit 6	1105	2202	2	12330	1772	82	348	80.5	84.2	3.94
circuit 7	1130	2208	1	218	1808	84	316	81.9	85.7	0.26
circuit 8	1496	3410	3	3491	3095	232	46	90.8	97.6	2.87
circuit 9	1975	2718	3	7395	1659	495	567	61.0	79.2	29.74
circuit 10	2855	4694	2	701	2168	262	2244	46.2	52.2	11.31
circuit 11	2855	4582	3	1485	4566	0	16	99.7	99.7	1.80
circuit 12	6863	15266	2	2753	3676	534	11056	24.1	27.6	47.82

* All CPU times were measured on a SUN SPARC 2 workstation with 48 MB memory.

실험결과에 나타난 것과 같이 10000 gate 이하의 회로에 대하여 SuperTest는 비교적 우수한 efficiency를 나타내고 있다. 하지만 circuit 12와 같은 큰 회로에 대해서는 efficiency가 매우 낮다. 많은 CPU time으로 인해 이 회로에 대해서는 두번째 phase 후에 test pattern 생성을 중지시켰다. 만약 세번째 phase까지 실험했다면 fault coverage 뿐만 아니라 efficiency도 증가했을 것이다.

V. 결론

Seuqential 회로에 대한 ATPG가 가능하게 된 것은 오래된 일이 아니고, commercial ATPG가 등장한 것도 최근의 일이다. 아직도 sequential 회로에 대한 ATPG에는 많은 어려움이 존재한다. Commercial ATPG도 10000 gate count(2 input NAND gate 기준) 이상의 회로에 대해서는 scan design을 요구하고 있다. Scan design methodology에는 full scan과 partial scan이 있다. Full scan은 회로의 모든 flip-flop을 scan flip-flop으로 바꾸어 scan-chain으로 연결하는 것으로 빠른 시간 내에 높은 fault coverage를 갖는 test pattern을 생성할 수 있다는 장점이 있지만, chip의 크기가 커지고 속도가 느려진다는 단점이 있다. Partial scan은 회로의 일부 flip-flop 만을 scan flip-flop으로 바꾸어 scan-chain으로 연결하는 것으로 full scan보다 fault coverage가 낮고 test pattern 생성에 많은 시간을 소비하지만, chip의 크기나 속도에 대한 overhead가 full scan보다 적다. Sequential 회로에 대한 ATPG algorithm은 지속적으로 발전할 것이며, 특히 partial scan design과 연계하여 실용적인 해답을 제공할 전망이다.

參 考 文 獻

[1] V. Agrawal, S. Seth, and P. Agrawal, "LSI product quality and fault coverage," *Proc.* 18th Design Automation Conf., pp. 196-203, June 1981.

[2] J. Hughes and E. McCluskey, "An Analysis of the multiple fault detection capabilities of single stuck-at fault test sets," *Proc. Int'l Test Conf.*, pp. 52-58, October 1984.

[3] E. Ulrich and T. Baker, "The concurrent simulation of nearly identical digital networks," *Proc.* 10th Design Automation Workshop, vol 6, pp. 145-150, June 1973.

[4] T. Niermann, W. Cheng, and J. Patel, "PROOFS: A fast, memory efficient sequential circuit fault simultor," *IEEE Trans.* on Computer Aided Design, pp. 198-207, February 1992.

[5] S. Seth, "On an improved diagnosis program," *IEEE Trans.* on Electronic Computers, vol. EC-14, pp. 76-79,

February 1965.

[6] H. Lee and D. Ha, "HOPE: An efficient parallel fault simulator for ssynchronous sequential circuits," *Proc.* 29th Design Automation Conf., pp. 336-340, June 1992.

[7] W. Cheng and T. Chakraborty, "Gentest - An automatic test generation system for sequential circuits," Computer, vol. 22, pp. 43-49, April 1989.

[8] T. Niermann and J. Patel, "HITEC: A test generation package for sequential circuits," *Proc.* European Design Automation Conf., pp. 1-5, February 1991

[9] J. Roth, W. Bouricius, and P. Schneider, "Programmed algorithms to compute tests to detect and distinguish between failures in logic circuits," *IEEE Trans.* on Electronic Computers, vol. EC-16, pp. 567-579, October 1967.

[10] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits," *IEEE Trans.* on Computers, vol. C-30, pp. 215-222, March 1981.

[11] P. Schneider, "On the necessity to examine D-chains in diagnostic test generation," IBM Journal of Research & Development, vol. 11, p. 114, January 1967.

[12] W. Cheng, "Split circuit model for test generation," *Proc.* 25th Dessign Automation Conf., pp. 96-101, June 1988.

[13] W. Cheng, "The back algorithm for sequential test generation,". *Proc. Int'l Conf.* Computer Design, pp. 66-69, October 1988.

[14] R. Marlett, "EBT, a comprehensive test generation technique for highly sequential circuits," *Proc.* 15th Design Automation Conf., pp. 332-339, June 1978.

筆 者 紹 介

金 錫 源
1964年 5月 4日生
1987年 2月 서울대학교 수학과 학사
1991年 2月 서울대학교 수학과 석사

1991年 2月 ~ 현재 삼성전자(주) ASIC 사업부

주관심 분야 : fault simulation, test generation scan design 등 design for testability

趙 敬 淳
1959年 10月 11日生
1982年 2月 서울대학교 전자공학과 학사
1984年 2月 서울대학교 전자공학과 석사
1988年 8月 Carnegie Mellon University, Electrical and Computer Engineering 박사

1988年 11月 ~ 현재 삼성전자(주) ASIC 사업부

주관심 분야 : ASIC Cell Characterization and Automation,
ASIC Design Automation, Test.

金 光 鉉
1955年 8月 22日生
1978年 2月 서강대학교 전자공학과 학사
1985年 5月 Virginia Tech 전기과 석사
1989年 5月 Virginia Tech 전기과 박사

1978年 3月 ~ 1983年 4月 국방과학연구소 연구원
1989年 6月 ~ 현재 삼성전자(주) ASIC 사업부

주관심 분야 : VLSI CAD, ASIC Design & Test.