혼합형 조합 회로용 고장 시뮬레이션 시스템의 설계 및 구현

박 영 호† · 손 진 우† · 박 은 세††

요 약

본 논문에서는 게이트 레벨 소자와 스위치 레벨 소자가 함께 사용된 혼합형 조합 회로에서의 고착 고장(stuck-at fault) 검출을 위한 고장 시뮬레이션에 대하여 기술한다. 실용적인 혼합형 회로의 고장 검출용으로 사용하기 위하여 게이트 레벨 및 정적 스위치 레벨 회로는 물론 동적 스위치 레벨의 회로들도 처리할 수 있도록 한다. 또한, wired 논리 소자에서의 다중 신호 충돌 현상을 해결하기 위하여 새로운 6치 논리값과 연산 규칙을 정의하여 신호 세기의 정보와 함께 사용한다. 고장 시뮬레이션의 기본 알고 리듬으로는 게이트 레벨 조합 회로에서 주로 사용되는 병렬 패턴 단일 고장 전달(PPSFP : parallel pattern single fault propagation) 기법을 스위치 레벨 소자에 확장 적용한다. 마지막으로 스위치 레벨 소자로 구현된 ISCAS85 벤치 마크 회로와 실제 혼합형 설계 회로에 대한 실험 결과를 통하여 본 연구에서 개발된 시스템의 효율성을 입증한다.

Design and Implementation of a Fault Simulation System for Mixed-level Combinational Logic Circuits

Young Ho Park† · Jin Woo Sohn† · Eun Sei Park††

ABSTRACT

This paper presents a fast fault simulation system for detecting stuck-at faults in mixed-level combinational logic circuits with gate level and switch-level primitives. For a practical fault simulator, the circuit types are not restricted to static switch-level and/or gate-level circuits, but include dynamic switch-level circuits. To efficiently handle the multiple signal contention problems at wired logic elements, we propose a six-valued logic system and its logic calculus which are used together with signal strength information. As a basic algorithm for the fault simulation process, a well-known gate-level parallel pattern single fault propagation(PPSFP) technique is extended to switch-level circuits in order to handle pass-transistor circuits and prechaged logic circuits as well as static CMOS circuits. Finally, we demonstrate the efficiency of our system through the experimental results for switch-level ISCAS85 benchmark combinational circuits and various industrial mixed-level circuits.

1. 서 론

최근 VLSI 회로의 설계 복잡도가 나날이 증가 됨에 따라 제작된 반도체 부품의 품질을 높이기 위한 효율적인 테스팅 기법의 개발이 매우 중요시되고 있다. 따라서 고착 고장(stuck-at fault) 검출은 물론 지연 시간 고장 검출 및 iddq 전류 테스팅 등 [illegible]

† 정 회 원 : 한국전자통신연구소 HW개발환경연구실
†† 정 회 원 : 한양대학교 공학대학 전자공학과
논문접수 : 1996년 5월 30일, 심사완료 : 1996년 11월 5일

한 고장 테스팅이 요구되고 있다[1, 2]. 일반적으로 완전 주문형 회로의 설계에서는 아직까지도 게이트 레벨 소자와 함께 스위치 레벨 소자를 사용한 혼합형 회로로 설계 표현을 하는데 이는 스위치 레벨 소자가 게이트 레벨 소자에 비하여 회로의 전기적 특성을 비교적 정확하게 표현할 수 있기 때문이다. 특히, pass transistor나 precharged 회로가 데이터 패스 설계에 많이 사용되고 있어 회로의 동작 검증이나 고장 검출 시뮬레이션에서 스위치 레벨 소자를 처리할 수 있어야 실용적인 가치가 있다.

스위치 레벨 고장 모델로는 고착 고장 외에 stuck-open 고장 및 stuck-on 고장 모델들이 지난 수년간 연구되어 왔으나 고장 검출을 위하여 과도한 설계 제약을 필요로 하기 때문에 실용적으로는 활용되지 못하고 있는 실정이다[3-5]. 스위치 레벨 회로의 고장 시뮬레이션은 주로 게이트 레벨 고장 시뮬레이션에서 사용되는 동시 고장 시뮬레이션(concurrent fault simulation) 기법을 확장 시켜 사용하고 있다[6]. 스위치 레벨 소자를 시뮬레이션 하기 위하여는 우선 논리값을 0, 1, X의 3치 논리값에 고저항 (high impedance) 상태를 표시하는 Z 값의 추가 및 신호 강도를 고려하여야 하므로 게이트 레벨 고장 시뮬레이션에 비하여 소프트웨어로 구현할 때 복잡도가 크게 증가한다. 그리고 스위치 레벨 회로의 경우에는 동일한 기능의 게이트 레벨 회로에 비하여 모델링 되는 고장의 수가 많아지므로 필요한 메모리 용량이 증가하게 되어 전체 시뮬레이션의 성능이 저하되는 단점이 있다. 이와 같은 단점을 극복하기 위하여 스위치 레벨 회로를 부울 연산식으로 표현되는 하나의 모듈로 변환시킨 후 컴파일하여 고속 시뮬레이션을 수행하는 방법도 제안되었다[7]. 그러나, 이 경우에는 부울 연산식의 입력과 출력에만 고장을 주입시킬 수 있어 모델링할 수 있는 고장의 종류와 수가 제한되는 단점이 있다.

한편, 게이트 레벨 조합 회로에서 사용되는 병렬 패턴 단일 고장 전달(PPSFP : parallel pattern single fault simulation) 기법을 스위치 레벨 소자에 확장하는 방법이 제안되었다[8]. 이 방법은 조합 회로에만 적용이 가능하지만 동시적 시뮬레이션 기법에 비하여 매우 빠른 장점이 있다. 그러나, 현재까지 발표된 병렬 패턴 단일 고장 전달 방식을 이용한 고장 시뮬레이션에서는 스위치 레벨 소자를 정적 스위치(static switch) 소자에만 국한시키기 때문에 동적 스위치(dynamic switch) 레벨 회로의 특징인 신호 세기(signal strength) 및 wired 소자에서의 다중 신호 충돌 현상에 의한 불확정(indeterminate) 논리값에 대한 처리가 미흡하여 부정확한 시뮬레이션 결과가 발생할 소지가 크다.

본 논문에서 기술하는 고착 고장 시뮬레이션은 게이트 레벨 소자 또는 단일 방향을 갖는 임의의 스위치 레벨 소자가 함께 사용된 혼합형 회로에서 사용된다. 조합 회로에서는 대부분의 스위치 소자를 신호의 전달 방향이 단일 방향인 소자로 가정할 수 있으므로 본 논문에서의 혼합형 조합 회로는 대부분의 스위치 레벨 회로 형태를 포함한다. 즉, 실용성을 높이기 위하여 스위치 레벨 회로의 종류를 정적 스위치 레벨 회로에 국한하지 않고 precharged 회로나 pass transistor로 구성된 동적 스위치 레벨의 회로들도 효율적으로 처리할 수 있도록 한다. 특별히, 동적 스위치 레벨 회로의 시뮬레이션에서는 wired 논리 소자에서의 논리값 해석이 중요한 문제이다. Wired 소자에서는 두개 이상의 서로 다른 논리값이 충돌하게 되면 불확정 논리값이 발생할 수 있으며 종래의 3치(0, 1, X) 논리값 또는 4치(0, 1, X, Z) 논리값으로는 이러한 현상을 충분히 설명하기 어렵다. 따라서, 새로운 6치 논리값 시스템을 정의하고 신호 세기 정보와 함께 사용함으로써 스위치 레벨 회로에 발생되는 문제점들을 해결하고자 한다. 또한, 고속 시뮬레이션을 위하여 PPSFP 기법을 스위치 레벨 회로에 확장 적용하고 이를 위하여 6치 논리값을 사용한 병렬 연산 법칙을 정의한다. 본 논문에서는 고착 고장에 대한 고장 시뮬레이션에 대하여 기술하고 있지만 제안된 방식은 iddq 테스팅을 위한 시뮬레이션 및 지연 시간 고장 시뮬레이션에도 쉽게 확장이 가능하다.

본 논문의 내용은 다음과 같다. 우선 2장에서는 스위치 레벨 회로 검증의 문제점들을 검토하고 그에 따른 회로 모델 및 새로운 6치 논리값 시스템을 정의하며 아울러 precharged CMOS 회로에 대한 테스팅 문제를 검토한다. 3장에서는 우선 PPSFP 알고리듬을 혼합형 회로에서도 적용할 수 있도록 6치 논리값의 비트 인코딩 방식과 병렬 논리 연산식을 기본 게이트 소자와 스위치 레벨 소자 그리고 wired 논리 소자에 대하여 정의한다. 그리고 4장에서는 제안된 고장 시뮬레이션 시스템의 전반적인 구성을 소개한다. 마지

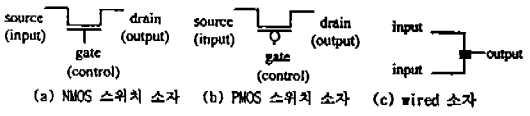
막으로 5장에서는 스위치 레벨 소자로 구현된 ISCAS-85 벤치마크 회로와 실제 혼합형 회로들에 대한 실험 결과를 통하여 본 논문에서 제시된 방법의 효율성을 입증한다.

2. 스위치 레벨 회로에서의 문제점

2.1 회로 모델

본 논문에서는 조합 회로 또는 테스트 시 조합 회로로 동작하게 하는 테스트를 고려한 스캔 기법으로 설계된 혼합형 회로에 대한 고장 시뮬레이션에 대하여 기술한다[9]. 여기서의 혼합형 회로는 게이트 레벨 소자 또는 단일 방향을 갖는 임의의 스위치 레벨 소자(NMOS, PMOS, CMOS등)가 함께 사용된 회로를 의미하는데. 조합 회로에서는 대부분의 스위치 소자를 신호의 전달 방향이 단일 방향인 소자로 가정할 수 있으므로 본 연구에서의 혼합형 조합 회로는 게이트 레벨 회로는 물론 대부분의 스위치 레벨 회로 형태를 포함한다. (그림 1)은 스위치 레벨 회로의 기본 소자를 보여 준다.

일반적으로 스위치 소자는 양방향으로 신호를 전달할 수 있으며 또한 두개 이상의 신호가 충돌하는 wired 논리 소자에서 논리값을 결정할 때 각 신호의 세기를 고려해야 하므로 회로 분석이 게이트 레벨 회로에 비해 복잡하다. 그러나 스위치 레벨에서 발생하는 문제를 모두 고려하는 것은 사실상 불가능하므로 어느 정도 실용적인 측면을 고려한 정확도와 신속성과의 타협점이 필요하다[10]. 따라서 본 연구에서는 조합 회로만을 고려함으로써 모든 스위치 소자를 단일 방향의 소자로 가정하고 각 신호의 방향은 [11]에서 개발된 방법을 사용하여 결정한다. 그러나, 본 연구에서 고려하고자 하는 스위치 레벨 회로의 형태는 정적 회로에 국한하지 않고 precharged CMOS 회로나 pass transistor로 구성된 동적 스위치 레벨의 회로들도 효율적으로 처리할 수 있도록 한다.

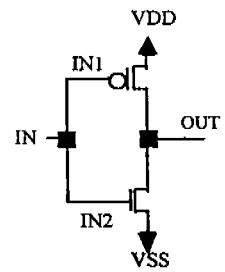


(그림 1) 스위치 레벨 회로의 기본 소자
(Fig. 1) Basic primitives in a switch-level circuit

2.2 고장 모델

본 논문에서 고려하는 고장 모델은 고착 고장 모델에 국한되어 있다. 스위치 레벨 고장 모델로는 전통적인 고착 고장 외에 stuck-open 고장 및 stuck-on 고장 모델들이 있으나, stuck-open 고장 및 stuck-on 고장을 검출하기 위하여는 과도한 설계 제약을 필요로 하기 때문에 실용적으로는 활용되지 못하고 있는 실정이다[3-5]. 따라서, 최근의 동향으로는 시간 지연 고장 테스팅으로 stuck-open 고장 테스팅을 대신하는 경향이 있으며 stuck-on 고장은 Iddq 전류 테스팅으로 대체되고 있다. 비록 본 논문에서는 기본적으로 고착 고장만을 대상으로 하고 있으나, 3장에서 기술되는 스위치 레벨 소자의 논리 연산 방식은 Iddq 전류 테스팅이나 시간 지연 고장 테스팅을 위한 고장 시뮬레이션에서도 쉽게 확장 가능하다.

따라서, 본 논문에서의 취약점은 스위치 레벨 회로에서도 고착 고장의 검출만을 목표로 하고 있기 때문에 stuck-open 고장 및 stuck-on 고장에 대한 검출이 불가능하다는 것이다. 예를 들어, (그림 2)의 CMOS에서 IN 단자의 고착 고장은 검출이 가능한 반면에 IN1 및 IN2 단자의 고착-0 고장과 고착-1 고장은 각각 stuck-open 고장 및 stuck-on 고장으로 모델링이 될 수 있으나 이들 고장은 단일 패턴의 고착 고장 테스트로는 검출이 불가능하다. 이러한 고장은 고장 시뮬레이션의 전처리 과정(preprocessing)에서 쉽게 찾아낼 수 있으므로 사전에 고장 리스트 작성시 제외시



(그림 2) CMOS 회로에서의 고착 고장 모델의 한계
(Fig. 2) Limitation of stuck-at fault model in a CMOS logic circuit

킬 수 있다. 즉, 전처리 과정에서 각 스위치 소자의 게이트 입력 단에 모델링 되는 고착 고장을 untestable 고착 고장으로 분류하여 고장 리스트에서 제거함으로써 이러한 고장을 검출하려는 시도를 사전에 방지하여 고장 시뮬레이션에 소요되는 시간을 절약할 수 있다.

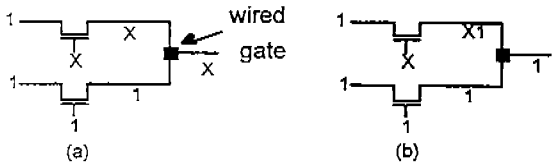
그러나 본 논문에서는 게이트 레벨 소자는 물론 스위치 소자 및 wired 논리 소자를 포함한 모든 소자의 입출력 단에서 가능한 모든 고착 고장을 대상으로 하고 있으므로 스위치 레벨 회로를 부울 연산식으로 표현되는 하나의 모듈로 변환시킨 후 부울 연산식의 입력과 출력에만 고장을 주입하는 방식(예:[7])과는 달리 직접 스위치 레벨 소자의 입출력 단에 대한 고장 시뮬레이션을 할 수 있는 장점이 있다. 예를 들어, complex CMOS 소자의 경우 [7]에서는 입력 단과 출력 단에만 고장을 주입할 수 있는 반면에 본 논문에서와 같이 complex CMOS 소자를 구성하는 각 스위치 레벨 소자의 입출력 단에도 고장 주입을 할 수 있는 장점이 있다.

고착 고장의 수는 스위치 레벨 소자의 모든 입출력 신호선에 단일 고착 고장 모델을 가정한 뒤 게이트 레벨 회로에서와 같은 fault collapsing 기법을 적용하여 산출한다. 일반적으로 스위치 레벨 회로의 고착 고장의 수는 동일한 기능의 게이트 레벨 회로에 비하여 훨씬 많은데, 그 이유는 스위치 소자에서는 fault equivalence 규칙이 적용되지 않기 때문이다.

2.3 6치 논리값 시스템

일반적으로 tri-state 소자를 포함하는 게이트 레벨 회로의 시뮬레이션에서는 0, 1, X(불확정값), Z(고저항값)의 4치 논리값을 사용하여 회로의 논리값 상태를 표현한다. 그러나 4치 논리값으로는 스위치 레벨 회로의 다양한 특성을 나타내기가 어렵다. 예를 들어 (그림 3(a))의 회로에서 4치 논리값을 사용하였을 경우 wired 소자의 출력 단의 실제 논리값이 1임에도 X로 나타나는 단점이 있다. 이러한 X 값의 발생은 회로의 시뮬레이션 결과에 큰 영향을 미치게 되어 부정확한 고장 시뮬레이션 결과를 초래할 수 있다.

본 연구에서는 스위치 레벨 회로의 논리값 상태를 보다 정확하게 나타내기 위하여 기존의 4치 논리값에 X0와 X1의 두개의 부분적으로 결정된 논리값(partially specified logic value)을 추가하여 6치 논리 시스템을 정의한다. X0는 0 또는 Z의 값을 갖는 부분적으로 결정된 논리값을 의미하며, X1는 1 또는 Z의 값을 갖는 부분적으로 결정된 논리값을 의미한다. 이 두개의 추가된 논리값들은 스위치 소자에서 게이트 신호의 논리값이 X인 경우의 출력 값을 결정하는 데 편리하며, 특히 (그림 3(b))의 경우에서처럼 wired 논리 소자의 출력 값을 확정 값인 1로 결정하게 되어 논리 연산 중에 발생될 수 있는 불확정 값을 최소화하는 장점이 있다. 추가된 논리값에 의하여 논리 연산이 다소 복잡해 질 수 있지만 그의 영향은 전체 고장 시뮬레이션에 소요되는 시간에서 볼 때 극히 미미한 정도이다. (그림 4)는 6치 논리값 시스템의 Hasse diagram이다.

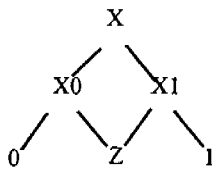


(그림 3) 논리값 X1의 사용 예
(Fig. 3) An example of the logic value X1

본 논문에서 제안된 새로운 논리값인 X0, X1과 비슷한 논리값 시스템으로 MOS 회로의 논리 시뮬레이션에서 사용된 Z0, Z1 논리값이 있다[12]. Z0와 Z1은 각기 고저항 상태에서 0과 1의 충전된 값을 가짐을 의미한다. 그러나, 본 연구에서는 조합 회로를 대상으로 하기 때문에 모든 테스트 패턴은 하나의 패턴으로 결정되며 또한 연속적인 테스트 패턴이 인가되는 주기가 충전된 논리값이 완전히 방전될 정도로 길다고 가정하기 때문에 이러한 논리값은 불필요하다. 그리고 X0와 X1의 논리값은 이전에 충전된 논리값과는 상관없이 스위치 소자의 게이트 단자가 불확정값(논리값 X)을 가질 때 출력 값을 부분적으로 확정된 값으로 표현하는데 사용된다.

6치 논리값 시스템에 의한 논리 소자의 논리 연산은 다음과 같다. 게이트 레벨 소자들의 경우에는 Z, X0 및 X1 논리값을 불확정 값인 X와 동일하게 처리함으로써 일반적인 3치 논리값(0, 1, X)에 의한 연산과 동일하게 된다. 예를 들어, AND 소자의 논리 연

산은 〈표 1〉과 같다. 또한, NMOS 스위치 및 wired 논리 소자의 논리 연산은 각각 〈표 2〉, 〈표 3〉과 같다. 그 밖의 다른 소자의 논리 연산도 〈표 1〉, 〈표 2〉 및 〈표 3〉으로 부터 쉽게 도출할 수 있다.



(그림 4) 6치 논리값 시스템의 Hasse Diagram
(Fig. 4) A hasse diagram for 6-valued logic system

〈표 1〉 AND 소자의 6치 논리 연산표
〈Table 1〉 6-valued logic calculus for an AND gate

	0	1	X	Z	X0	X1
0	0	0	0	0	0	0
1	0	1	X	Z	X0	X1
X	0	X	X	X	X	X
Z	0	X	X	X	X	X
X0	0	X	X	X	X	X
X1	0	X	X	X	X	X

〈표 2〉 NMOS 스위치 소자의 6치 논리 연산표
〈Table 2〉 6-valued logic calculus for an NMOS switch
(G : gate signal, I : input signal)

G/I	0	1	X	Z	X0	X1
0	Z	Z	Z	Z	Z	Z
1	0	1	X	Z	X0	X1
X	X0	X1	X	Z	X0	X1
Z	X0	X1	X	Z	X0	X1
X0	X0	X1	X	Z	X0	X1
X1	X0	X1	X	Z	X0	X1

〈표 3〉 Wired 논리 소자의 6치 논리 연산표
〈Table 3〉 6-valued logic calculus for a wired logic gate

	0	1	X	Z	X0	X1
0	0	X	X	0	0	X
1	X	1	X	1	X	1
X	X	X	X	X	X	X
Z	0	1	X	Z	X0	X1
X0	0	X	X	X0	X0	X
X1	X	1	X	X1	X	X1

2.4 신호 세기 정보의 처리

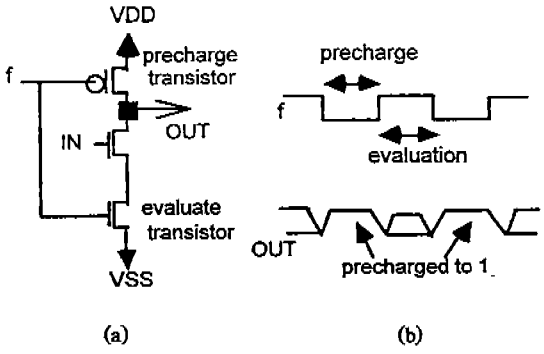
완벽한 스위치 레벨 회로의 분석을 위해서는 무한 개의 등급을 갖는 신호 세기를 신호의 논리값 상태와 함께 고려해야 하지만[13], SPICE와 같은 회로 수준(circuit-level) 분석이 아니고는 서로 다른 세기를 갖는 신호들의 충돌 현상을 완벽하게 분석하기는 어렵다. 더구나 실용적인 측면에서 볼 때, 미세한 신호 세기의 차이를 이용하여 회로 동작을 시키는 경우는 거의 없으며 실제 있다고 하더라고 논리 시뮬레이션을 통하여 검증을 마친 것으로 가정할 때 고장 시뮬레이션에서는 비교적 간단한 신호 세기의 체계를 사용할 수 있다. 본 논문에서는 신호의 세기를 STRONG, WEAK, CHARGE의 3단계로 표현한다. 모든 신호는 묵시적으로 STRONG하다고 가정하며 신호 세기의 크기는 STRONG, WEAK, CHARGE 순서로 가정한다.

Precharged 스위치 레벨 회로의 경우 precharge된 신호 값이 CHARGE 신호세기를 가지게 되며 시뮬레이션 중에는 고저항 논리값 Z가 precharge된 논리값으로 대체된다. 또한 pull-up 또는 pull-down 저항을 거친 신호의 세기는 WEAK로 표현된다. 예를 들어, WEAK 세기의 논리값 1이 STRONG 세기의 논리값 0와 wired 논리 소자에서 만나게 되면 출력 값이 STRONG 세기의 논리값 1이 된다. 이러한 WEAK 신호 세기는 NMOS 회로 및 pseudo-NMOS 회로 분석에도 유용하게 사용될 수 있다. 이러한 3단계의 신호 세기의 구분은 종래의 많은 스위치 레벨 시뮬레이션에서 사용된 것과 유사하다고 할 수 있지만 정적 스위치 소자만으로 구성된 회로만을 대상으로 하는 고장

시뮬레이션에서는 무시되어 왔다. 그러나 일반적으로 동적 스위치 레벨 회로에서 신호 세기를 고려하지 않으면 고장 검출율의 감소 현상을 초래할 수 있다. 따라서, 본 연구에서는 회로의 각 노드에 대한 신호값을 (논리값, 신호세기)의 두개의 값으로 표현한다.

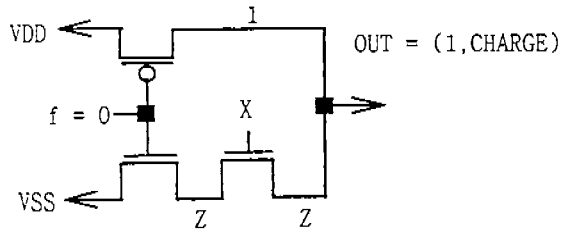
2.5 Precharged 동적 스위치 레벨 회로의 처리

(그림 5)의 precharged CMOS 회로에서 f는 precharge clock이며 f의 논리값이 0인 동안 출력 값이 1로 precharge되고 f가 1일 때 precharge transistor가 off 상태가 되어 입력 IN의 값에 따라 출력 OUT의 논리값이 결정된다. 즉, (그림 5(b))에서 f=1이고 IN=0일 때 출력 값은 Z이지만 이미 precharge된 1에 의해 대체되며 f=1이고 IN=1일 때는 VSS에서 OUT 단자로 나오는 STRONG 세기의 논리값 0이 precharge된 논리값 1보다 강하므로 출력 값은 0이 된다.



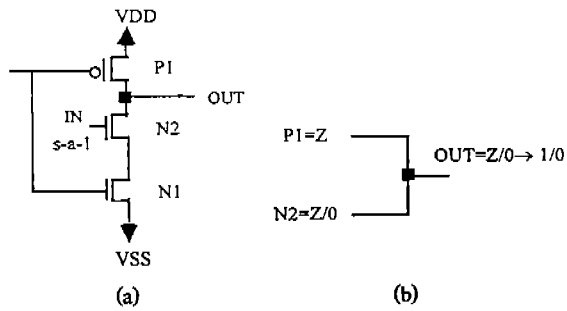
(그림 5) Precharged 스위치 레벨 inverter 회로의 분석
(Fig. 5) Circuit analysis for a precharged switch-level inverter

Precharged 회로를 고장 시뮬레이션에서 처리하기 위하여는 precharge되는 노드와 그 논리값을 사전에 알고 있어야 한다. 이를 위하여, 본 연구에서는 회로의 precharge clock ((그림 5(a))의 f)신호가 미리 입력의 형태로 주어진다고 가정하고 고장 시뮬레이션의 전처리 과정에서 시뮬레이션을 통하여 precharge되는 노드와 그 논리값이 CHARGE 신호 세기로 결정된다. 예를 들어, (그림 6)에서 precharge clock f가 주어지면 모든 주입력 신호를 X 값으로 한 뒤 f에 논리값 0을 가하여 시뮬레이션을 하게 되면 wired 소자의 논리값이 1로 결정되며 신호세기는 CHARGE로 표시하여 wired 소자의 출력에 저장한다. 이와 같은 방식으로 결정된 precharge되는 논리값은 고장 시뮬레이션의 논리값 연산에서 wired 소자의 출력 값이 Z이면 Z의 값이 precharge된 논리값으로 대체된다.



(그림 6) Precharged 노드와 논리값 결정
(Fig. 6) Analysis of a precharged node and its logic value

고장이 존재하는 경우의 동작은 다음과 같다. 예를 들어, (그림 7(a))에서와 같이 IN의 고착-1 고장을 검출하기 위하여 IN=0와 f=1 논리값을 인가하면 (그림 7(b))에서 알 수 있듯이 NMOS 스위치 N2의 출력값이 고장이 없을 경우에는 Z로 나타나고 고장이 있을 시에는 0으로 나타나게 된다. 이때, 출력 OUT에 precharge된 값이 없으면 OUT=Z/0 (good Z/faulty 0)가 되지만 출력 단이 논리값 1로 precharge 되어 있으므로 OUT=1/0이 되어 고장이 전파된다. 따라서 고장에 의해 Z 값을 갖는 노드가 존재하더라도 만약 그 노드가 precharge되어 있거나 또는 Z 논리값 상태가 precharge되어 있는 노드까지 전달된다면 Z 값이



(그림 7) Precharged inverter 회로의 테스팅
(Fig. 7) Testing for a precharged inverter circuit

미리 precharge된 0 또는 1에 의해 대체되어 고장을 검출할 수 있다. 한편, precharge된 회로에서의 stuck-on 및 stuck-open 고장이 고착 고장을 검출하는 테스트로 모두 검출 가능함이 증명되어 있다[14].

3. 병렬 논리 연산

3.1 논리값 비트 인코딩

PPSFP 알고리듬에서 사용하는 병렬 패턴 논리 시뮬레이션은 컴퓨터 워드에 각 테스트 패턴을 비트 단위로 표시하고 컴퓨터 언어의 논리 연산자를 사용하여 시뮬레이션을 수행하게 된다. 본 연구에서는 6치 논리값을 사용하므로 각 논리값을 표시하려면 3개의 비트가 필요하다($6<2^3$이므로). 32개의 테스트 패턴을 나타내려면 3개의 32비트 컴퓨터 워드가 필요하며 따라서 회로 내의 모든 소자의 입출력 신호는 3개의 32비트 워드를 병렬로 사용하여 표시된다. 각 논리값의 비트 인코딩은 〈표 4〉와 같이 정의하였으며 게이트 및 스위치 레벨 소자의 병렬 논리 연산 규칙을 결정하는데 사용된다.

〈표 4〉 6치 논리값의 비트 인코딩
〈Table 4〉 Bit encoding of the 6-valued logic system

워드/논리값	0	1	X	Z	X0	X1
워드0	0	1	0	1	0	1
워드1	1	0	0	1	1	0
워드2	0	0	0	1	1	1

3.2 게이트 레벨 논리 소자의 연산

게이트 레벨 논리 소자들에 대한 병렬 비트 연산은 논리 연산자(&: AND, |: OR, !: NOT, ^: exclusive-OR)를 사용하여 정의할 수 있다. 우선, Z, X0, X1의 논리값을 갖는 신호들은 게이트 레벨 회로에서는 논리값이 결정되지 않은 상태이므로 그 신호 값을 X로 만들어 주고 난 후에 연산을 수행한다. 다음은 2개의 입력을 갖는 AND 소자에 대한 논리 연산을 정의하였으며 소자의 출력 단 OUT과 입력 단 IN1, IN2의 논리값은 모두 3개의 32 비트 워드로 나타낸다. OR 또는 NOT 소자 등의 경우도 동일한 방법으로 논리 연산을 할 수 있다.

〈2-input AND 소자의 병렬 논리 연산〉

Step 1: IN1 또는 IN2의 논리값이 Z, X0, X1이면 X로 변환(논리값 Z, X0, X1는 워드2의 비트가 1이므로 워드 1, 2, 3의 해당 비트를 0으로 변환한다)

아래의 식에서 IN1[i]와 IN2[i]는 i-번째 32 비트 워드를 의미하며 mask는 32 비트 워드로 X0 및 X1을 갖는 비트의 위치를 알아내기 위하여 사용된다.

```
mask = IN1[2]
IN1[0] = IN1[0] & (!mask)
IN1[1] = IN1[1] & (!mask)
IN1[2] = IN1[2] & (!mask)
```

Step 2: 출력값 결정

```
OUT[0] = IN1[0] & IN2[0]
OUT[1] = IN1[1] | IN2[1]
OUT[2] = IN1[2] & IN2[2]
```

3.3 스위치 레벨 논리 소자의 연산

NMOS 스위치 소자의 논리 연산은 게이트 신호의 논리값에 따라 〈표 2〉에 의해 비트 연산 규칙을 정의하며 PMOS와 CMOS의 논리 연산도 비슷한 방법으로 정의할 수 있다. 논리 연산 방법을 설명하기 위하여 입력(source) 신호를 IN, 출력(drain) 신호를 OUT, 그리고 게이트 신호를 GATE로 표기하며 이들 각 신호도 3개의 32 비트 워드로 나타낸다.

〈NMOS 스위치 소자의 병렬 논리 연산〉

Step 1: 게이트 신호의 논리값이 Z, X0, X1이면 X로 변환

```
mask1 = GATE[2]
GATE[0] = GATE[0] & (!mask1)
GATE[1] = GATE[1] & (!mask1)
GATE[2] = GATE[2] & (!mask1)
```

Step 2: 게이트 신호의 논리값이 0, X인 비트를 각각 mask2, mask3으로 구한다.

```
mask2 = (!GATE[0]) & GATE[1]
mask3 = !(GATE[0] | GATE[1])
```

Step 3: 출력 단의 논리값 결정(게이트 신호가 0이면 출력 값은 Z가 되고 1이면 입력 값이 출력

으로 그대로 통과하게 된다. 최종적으로 게이트 신호가 X인 비트에 대하여 mask3으로 출력 단의 논리값을 보정한다)

OUT[0]=IN[0] | mask2
OUT[1]=IN[1] | mask2
OUT[2]=IN[2] | mask2
OUT[2]=OUT[2] | mask3

〈NMOS 스위치 소자의 병렬 논리 연산의 예〉

설명의 편이를 위하여 워드의 크기를 1 비트라고 가정하고 IN=0, GATE=X인 경우와 IN=1, GATE=X0인 경우에 대하여 병렬 논리 연산 과정을 설명하기로 한다. 우선, 비트 인코딩에 의하여 IN=(워드0, 워드1, 워드2)=(0, 1, 0), GATE=(0, 0, 0)으로 표기하고 연산 과정은 앞에서 정의한 논리 규칙에 따라 다음과 같이 계산한다.

Step 1:게이트 신호의 논리값이 X이므로 설명을 생략한다.

Step 2:게이트 신호의 논리값이 0, X인 비트를 각각 mask2, mask3으로 구한다.

mask2=(!GATE[0]) & GATE[1]=(!0) & 0=0
mask3=!(GATE[0] | GATE[1])=!(0 | 0)=1

계산 결과에서 볼 때, 게이트 신호가 0이 아니고 X 임을 알 수 있다.

Step 3:출력 단의 논리값 결정

OUT[0]=IN[0] | mask2=0 | 0=0
OUT[1]=IN[1] | mask2=1 | 0=1
OUT[2]=IN[2] | mask2=0 | 0=0

이때, 게이트 신호가 X이므로 mask3으로 출력 단의 논리값을 보정한다. 즉,

OUT[2]=OUT[2] | mask3=0 | 1=1

따라서, 출력단 OUT의 비트 인코딩은 (워드0, 워드1, 워드2)=(0, 1, 1)이 되어 논리값 X0 임을 알 수 있다.

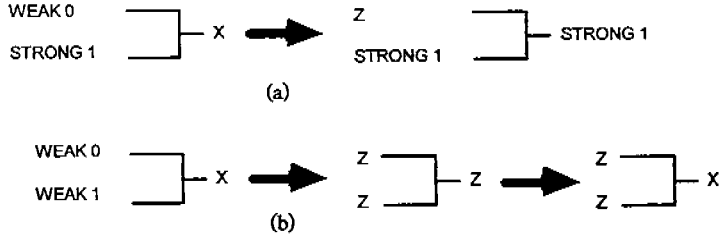
3.4 Wired 소자의 연산 및 신호세기를 고려한 연산

기본적으로 2-input wired 논리 소자에 대한 비트 논리 연산은 아래와 같다.

OUT[0]=IN1[0] & IN2[0]
OUT[1]=IN1[1] & IN2[1]
OUT[2]=IN1[2] & IN2[2]

Wired 논리 소자에서는 출력 값이 X 또는 Z인 경우에 WEAK 또는 CHARGE의 신호 세기에 의한 논리값 보정이 필요한 경우가 발생할 수 있다. Wired 논리 소자의 출력 값이 Z이고 wired 논리 소자의 출력이 precharge되어 있을 때는 precharge된 논리값이 CHARGE 신호 세기를 가지고 있으므로 출력 값인 Z를 precharge된 논리값으로 바꾼다.

WEAK 신호 세기에 의한 논리값 보정 방법은 다음과 같다. Wired 논리 소자의 출력 값이 X이고 입력 신호 중에서 WEAK 신호 세기가 있을 경우에 논리값 보정을 시도한다. 우선 wired 논리 소자의 입력 값 중에서 신호의 세기가 WEAK인 신호를 논리값 Z로 바꾼 후 논리 연산을 다시 한다. 이때 출력 값이 Z로 나오면 wired 소자의 모든 입력들이 WEAK 세기이며 동시에 논리값 0과 1을 가지고 있는 상태이므로 그 값을 X로 원상 복귀시킨다. (그림 8)은 wired 소자 연산 과정을 보여 준다.

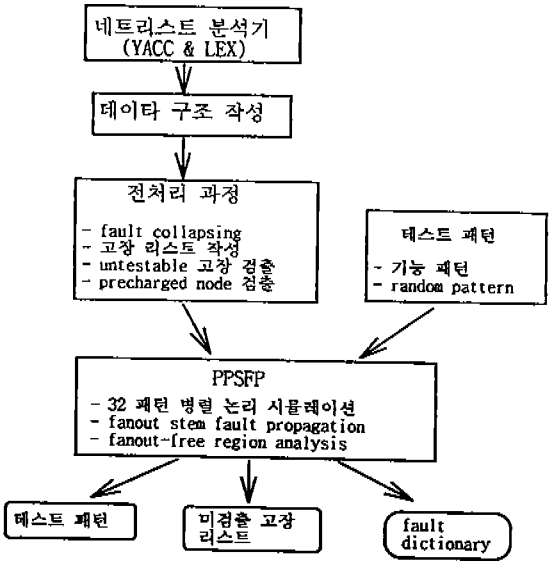


(그림 8) 신호 세기를 고려한 wire 소자의 논리 연산
(Fig. 8) The logical operation of a wire gate with signal strength

4. 고장 시뮬레이션 시스템 구성

본 연구에서 개발된 고장 시뮬레이션 시스템의 구성도는 (그림 9)와 같다. 우선, EDIF로 표현된 회로를 YACC 및 LEX로 구현된 네트리스트 분석기를 통하여 메모리 내에 데이터 구조를 작성한다. 그리고 전처리 과정에서는 먼저 fault collapsing을 수행하여 고장 리스트를 작성하고 그 다음 스위치 레벨 untestable 고장을 검출한다. 스위치 레벨 untestable 고장은 power 신호에 가정된 고착-1 고장 및 ground 신호의 고착-0 고장 등과 NMOS 및 PMOS 스위치 소자의 게이트 단자에 가정된 고착-0 및 고착-1 고장들을 의미하는데, 이러한 고장들은 CMOS 회로의 고착 고장 모델 하에서는 검출이 불가능하기 때문에 전처리 과정에서 미리 검출하여 고장 리스트에서 제거함으로써 고장 시뮬레이션 시스템의 효율을 높일 수 있다. 그리고 precharge 회로의 경우에는 2.2절에서와 같이 precharge clock 신호로부터 precharge되는 노드와 그 논리값을 사전에 알아내 고장 시뮬레이션 수행 중에 이용하게 된다.

PPSFP 알고리듬은 32 패턴을 병렬로 논리 시뮬레이션을 하고 난 뒤 한번에 하나의 fanout stem 고장을 주출력 단으로 전달시켜 봄으로써 고장 검출 가능 여부를 판정한다. 이때, 무분기선 영역(fanout-free region)에서의 고장의 영향을 분석하는 기법과 함께 사용하여 고속 고장 시뮬레이션을 가능하게 한다[15, 16]. 본 연구에서 사용된 고장 시뮬레이션의 알고리듬은 기존의 PPSFP 알고리듬과 동일하기 때문에 기본적인 PPSFP 기법에 대한 설명은 생략하기로 한다. 또한, 고장 시뮬레이션에 사용되는 테스트 패턴은 사용자가 정의한 기능 패턴 또는 임의의 패턴 중에 선택할 수 있게 되어 있다. 고장 시뮬레이션의 결과로는 테스트 패턴, 미검출 고장 리스트, 고장 검출 통계 자료 및 고장 사전 등이 생성된다.



(그림 9) 고장 시뮬레이션 시스템 구성도
(Fig. 9) Overall structure of fault simulation system

5. 실험 결과

본 논문에서 제안한 고장 시뮬레이션 시스템은 C 언어로 구현되었으며 여러 가지 실험이 ISCAS85 게이트 레벨 회로[17]와 스위치 레벨로 구현된 회로, 그리고 실제 혼합형 조합 회로에 대하여 HP710(주기억장치 32Mb) 워크스테이션에서 시행되었다. ISCAS85 회로에서 게이트 레벨과 스위치 레벨 회로간의 비교는 본 연구에서 구현된 고장 시뮬레이션의 성능을 비교하고 동시에 고착 고장 모델의 한계점을 보이기 위한 것이다. 한편, 실제 혼합형 회로에 대한 실험은 본 논문에서 제안된 스위치 레벨 소자의 논리 연산 방식의 효율성을 보여 주며 기존의 제한된 형태의 혼합형 회로 또는 스위치 레벨 회로의 고장 시뮬레이션 방법[8, 11]과의 차별성을 보여 준다.

5.1 게이트 레벨 ISCAS85 회로의 고장 시뮬레이션 결과

〈표 5〉는 게이트 레벨 ISCAS85 벤치마크 회로에 대한 실험 결과이다. 실험에서 사용한 테스트 패턴은 임의로 생성된 패턴(random pattern)이며 모두 4096개의 패턴이 한번에 32개씩 병렬로 시뮬레이션 되었다. 사용된 컴퓨터 기종 및 패턴 수의 차이가 있기 때문에 〈표 5〉의 고장 시뮬레이션의 결과를 기존에 발표된 PPSFP 방식의 고장 시뮬레이션들의 결과[14, 15]와 객관적인 비교는 할 수 없었으나 대체로 비슷한 수준의 성능을 갖는 것으로 사료된다. 〈표 5〉에서 고장 검출율은(검출된 고장 수/전체 고장 수)×100%

〈표 5〉 게이트 레벨 ISCAS85회로의 고장 시뮬레이션 결과
〈Table 5〉 Fault simulation results for gate-level ISCAS85 circuits

ckt. name	# of gates	# of PIs	# of POs	# of faults	# random patterns	fault cov.(%)	CPU seconds
C432	211	36	17	524	4096	99.2	0.4
C880	469	60	26	942	4096	100	1.0
C1355	627	41	32	1574	4096	99.5	1.8
C1908	990	33	25	1879	4096	99.3	3.0
C2670	1575	233	140	2747	4096	84.4	6.5
C3540	1775	50	22	3428	4096	95.8	16.2
C5315	2631	178	123	5350	4096	98.6	7.3
C6288	2480	32	32	7744	4096	99.6	26.5
C7552	3883	207	108	7550	4096	93.6	16.2

로 정의하였다.

5.2 스위치 레벨 ISCAS85 회로의 고장 시뮬레이션 결과

〈표 6〉은 스위치 레벨로 변환된 ISCAS85 회로에 대한 고장 시뮬레이션의 결과이다. 스위치 레벨 회로로의 변환을 위하여 각 게이트 레벨 소자를 동일한 기능을 갖는 fully-complemented static CMOS의 스위치 레벨 소자로 대체하였다. 〈표 6〉에서 고착 고장의 수는 스위치 레벨 소자의 모든 입출력 신호선에 단일 고착 고장 모델을 가정한 뒤 게이트 레벨 회로에서와 같은 fault collapsing 기법을 적용하여 산출한 것이다. 〈표 6〉에서 알 수 있듯이 스위치 레벨 회로의 고착 고장의 수는 동일한 기능의 게이트 레벨 회로에 비하여 훨씬 많은데, 그 이유는 스위치 소자에서는 fault equivalence 규칙이 적용되지 않기 때문이다.

〈표 6〉의 결과를 〈표 5〉의 게이트 레벨 회로의 결과와 비교할 때 고장 검출율(fault coverage)의 현저한 저하와 CPU 시간의 증가를 알 수 있다. 고장 검출율의 저하는 스위치 레벨 회로에서는 단일 고착 고장 모델로는 검출이 불가능한 untestable 고장이 많기 때문이다. 이러한 untestable 고장은 대부분 power 신호에 가정된 고착-1 고장 및 ground 신호의 고착-0 고장 등과 NMOS 및 PMOS 스위치 소자의 게이트 단자에 가정된 고착-0 및 고착-1 고장들이다. 고착 고장 모델 하에서는 untestable 고장에 대한 검출이 불가능하기 때문에 그러한 고장을 고장 시뮬레이션의 전처리 과정에서 미리 검출하여 고장 리스트에서 제거시킬 수 있다. 〈표 6〉에서는 전처리 과정에서 검출된 untestable 고장의 수를 보여 주고 있으며 고장 검출율과 구별하여 테스트 효율을 [(검출된 고장수 +untestable 고장수)/전체 고장수]×100%로 정의하였다.

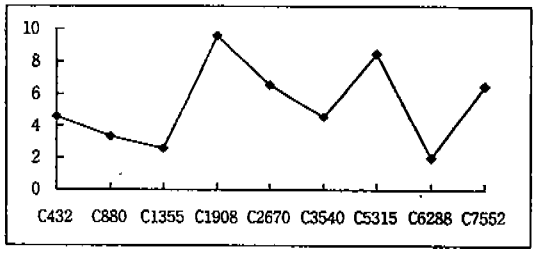
CPU 시간의 증가를 설명하기 위하여 스위치 레벨 회로와 게이트 레벨 회로에서 하나의 고장 당 소요되는 평균 CPU 시간의 비율로 (그림 10)에 나타내었다. 이 비율은 〈표 6〉의 전체 고장 중에서 untestable 고장을 제외한 고장만을 고려한 것이므로 전체 고장을 고려하면 스위치 레벨 회로의 성능이 훨씬 우수하게 나타날 수 있지만 여기서는 고장 시뮬레이션 자체의 성능만을 비교하기 위하여 untestable 고장을 제외하였다. 이와 같이 untestable 고장이 많은 것은 고착 고장에 의한 모델링이 CMOS 회로에서 모든 고장을 효과적으로 모델링할 수 없음을 보여주는데 이에 대한 해결책으로는 2장에서 기술한 바와 같이 Iddq 테스팅 및 시간 지연 고장 테스팅의 적용이라 할 수 있다. 그

〈표 6〉 스위치 레벨 ISCAS85회로의 고장 시뮬레이션 결과
〈Table 6〉 Fault simulation results for switch-level ISCAS85 circuits

ckt. name	# of switches	# of PIs	# of POs	# of faults	untest-able faults	# target faults	# random patterns	fault cov. (%)	test effect-iveness (%)	CPU seconds
C432	896	36	17	5428	3356	2072	4096	36.4	96.4	7.3
C880	1802	60	26	11354	6812	4542	4096	37.6	100.0	16.1
C1355	2308	41	32	14170	8652	5518	4096	35.4	95.4	16.5
C1908	3446	33	25	21720	12844	8876	4096	35.9	95.9	136.2
C2670	5668	233	140	36614	21247	15367	4096	31.6	91.6	240.7
C3540	7504	50	22	47552	28302	19250	4096	36.5	96.5	421.3
C5315	11262	178	123	70878	42345	28553	4096	37.8	97.8	332.4
C6288	10112	32	32	61316	38066	23250	4096	37.7	97.7	163.2
C7552	15400	207	108	97722	57916	39806	4096	35.7	95.7	568.1

러나, 검출된 고장의 수는 게이트 레벨 ISCAS85 회로에서 검출된 고장을 모두 포함하고 있다. 또한, [11]의 실험 결과에서도 ISCAS85 스위치 레벨 회로에 대하여 고착 고장은 실제로 게이트 레벨 고착 고장만을 삽입하였다.

(그림 10)에서 알 수 있듯이 평균적으로 스위치 레벨 회로의 고장이 2.0~9.6배 정도의 시간이 더 소요되었지만, 이는 스위치 레벨 회로가 동일 기능의 게이트 레벨 회로에 비하여 9~14배 더 많은 소자를 포함하고 있고 일반적으로 고장 시뮬레이션의 속도가 전체 소자의 개수의 제곱에 비례함을 감안할 때, 본 연구에서 제안된 고장 시뮬레이션의 속도가 고속임을 입증한다고 할 수 있다. 이러한 고속 고장 시뮬레이션이 가능한 이유는 첫째, 기존 PPSFP 알고리듬의 우수성을 들 수 있고, 둘째, 본 연구에서 도입된 비트 인코딩 및 병렬 논리 연산 방법이 PPSFP 알고리듬에 효율적으로 적용될 수 있었기 때문이라 판단된다.



(그림 10) 스위치 및 게이트 레벨 ISCAS85 회로에서 하나의 고장에 소요되는 고장 시뮬레이션 CPU 시간 비율

(Fig. 10) CPU time ratio per fault between switch-level and gate-level ISCAS85 circuits

〈표 6〉의 결과를 보다 정확하게 평가하려면 기존에 보고된 결과와 비교하여야 하지만 불행히도 직접적인 비교가 되는 연구 결과가 발표된 적이 없었다. 가장 근접한 결과는 Hwang[8]등에 의한 결과이나, 그들의 실험 결과는 전체 스위치 레벨 ISCAS85 회로에 대한 결과가 아니라 비교적 작은 크기의 회로 만에 대한 결과이며 더구나 가능한 모든 고장에 대한 고장 시뮬레이션이 아닌 일부 테스트 가능한 고장만을 대상으로 하였기 때문에 직접 비교가 어렵다. 그러나, CPU 성능을 감안한 간접적인 성능 비교를 하더라도 ([8]의 결과는 SUN3/110 사용), 본 연구의 결과가 평균적으로 3배 이상 빠름을 알 수 있었다.

5.3 혼합형 회로의 고장 시뮬레이션 결과

마지막으로 실제 혼합형 조합 회로에 대하여 고장 시뮬레이션을 수행하였다. 〈표 7〉에서 회로 A는 순수 스위치 레벨 회로로 주출력 단이 precharged bus에 연결되어 있는 데이터 패스 회로로 논리값이 1로 precharge되어 있어 주출력 단까지 0/Z 또는 Z/0가 전달되면 고장 검출이 가능하다. 회로 B와 C는 혼합형 회로인데, 특히 회로 B는 일부 스위치가 WEAK 신호 세기를 가지고 있으며 ALU 기능을 한다. 또한, 회로 C는 스위치 레벨 회로가 pass transistor로 구성되어 있는 barrel shifter 회로이다. 그리고 회로 D는 순수 게이트 레벨 회로이며 PLA 구조이다. 이 실험에서는 임의의 패턴을 한번에 32개씩 고장 시뮬레이션할 때, 다섯번(32×5=160 패턴) 연속해서 새로운 고장을 검출하지 못할 때는 고장 시뮬레이션을 멈추게 하였다. 특별히 언급할 것은 precharge된 회로인 A에서 untestable인 고장이 없었으며 고장 시뮬레이션 후에 남은 고장들도 모두 테스트가 가능하거나 redundant한 고장이었다. 그리고 회로 D는 PLA 구조이기 때문에 임의의 패턴에 테스트가 잘 안되어 다른 회로에 비하여 패턴 수와 CPU 시간이 많이 들었다.

이 실험의 결과로 알 수 있는 것은 본 연구에서 제안된 고장 시뮬레이션이 특히 혼합형 조합 회로에서 우수한 성능을 보여 주고 있는데, 이는 신호 세기 및 6치 논리값 시스템을 PPSFP 알고리듬에 무리 없이 접목되어 신속 정확한 논리 연산이 가능하기 때문이라고 판단된다.

6. 결 론

본 논문에서는 혼합형 조합 회로에서의 고착 고장을 검출하기 위한 고장 시뮬레이션에 대하여 기술하였다. 게이트 레벨 고장 검출 시뮬레이션에서 널리 사용되고 있는 PPSFP(parallel pattern single fault propagation) 기법을 스위치 레벨 회로에 확장 적용하였으며, 또한 스위치 레벨 회로의 종류를 정적 스위치 레벨 회로에 국한하지 않고 precharged 회로나 pass transistor로 구성된 동적 스위치 레벨 회로들도 처리

〈표 7〉 실용 혼합형 회로의 고장 시뮬레이션 결과
〈Table 7〉 Fault simulation results for practical circuit examples

ckt. name	# of logic elements	# of swit-ches	# of PIs	# of POs	# of faults	untesta-ble faults	# of target faults	# of random patts.	fault cov. (%)	test eff. (%)	CPU (sec)
A	1256	1256	36	17	5428	0	5428	736	87.5	87.5	13.7
B	5878	338	535	368	11724	265	11459	4096	83.9	90.0	49.6
C	1381	508	182	113	4205	976	3229	1120	78.8	98.8	28.9
D	924	0	23	29	9359	0	9359	11968	67.6	67.6	366.7

할 수 있도록 하여 실용적인 완전 주문형 반도체 회로에도 적용이 가능하도록 하였다. 아울러, 다수의 신호가 충돌하는 wired 논리 소자에서의 불확정 논리값에 의한 부정확한 시뮬레이션을 방지하기 위하여 새로운 6치 논리값을 정의하여 신호 세기의 정보와 함께 사용하였다.

ISCAS85 벤치마크 회로와 실제 설계 회로에 대한 실험을 통하여 새로 도입된 6치 논리값 시스템과 병렬 연산에 의한 고장 시뮬레이션이 매우 우수한 결과를 얻을 수 있음을 알 수 있었다. 특히, 종전의 고장 시뮬레이션에서 처리하기 어려웠던 precharged CMOS 회로 및 pass-transistor로 이루어진 동적 스위치 레벨 회로에서 더욱 효과적이었다. 그러나, ISCAS85 스위치 레벨 회로의 각 고장 당 소요되는 평균 시뮬레이션 시간이 동일 기능의 게이트 레벨 고장에 비하여 2~9.6배 증가함을 알 수 있었는데, 이에 대한 해결 방법은 계층적 데이터 구조를 갖는 회로에 대한 고장 시뮬레이션 알고리듬의 개발을 들 수 있다. 즉, 고장이 존재하는 소자만을 스위치 레벨로 표현하고 나머지 소자들은 게이트 레벨로 표현하여 전체 소자의 수를 감소시킴으로써 고장 시뮬레이션의 속도를 크게 향상시킬 수 있을 것으로 기대된다. 또한, 고착 고장 모델로 해결할 수 없는 스위치 소자의 게이트 신호의 고장에 대한 해결책으로 iddq 전류 테스팅과 지연 시간 고장 테스팅 기능의 추가 및 테스트 패턴의 수를 줄이기 위한 compaction 기법 개발 등을 꼽을 수 있다.

참 고 문 헌

[1] T. W. Williams and N. C. Brown, "Defect Level as a Function of Fault Coverage," IEEE Trans. on Comput., vol.C-30, no.12, pp.987-988, 1981.

[2] F. J. Ferguson and J. P. Shen, "Extraction and Simulation of Realistic CMOS Faults Using Inductive Fault Analysis," IEEE Proc. Int. Test Conference., pp.475-484, 1988.

[3] R. L. Wadsack, "Fault Modeling and Logic Simulation of CMOS and MOS Integrated Circuits," Bell System Technical Journal, Vol.57, pp. 1449-1488, 1978.

[4] T. Storey, W. Maly, J. Andrews, and M. Miske, "Stuck Fault and Current Test Comparison Using CMOS Chip Test," IEEE Proc. Int. Test Conference, pp.311-318, 1991.

[5] P. C. Maxwell, R. Aitken, V. johansen, and I. Chiang, "The Effectiveness of IDDQ, Functional and Scan Tests: How Much Fault Coverages Do We Need?", IEEE Proc. Int. Test Conference, pp. 168-177, 1992.

[6] R. E. Bryant, "Performance Evaluation of FMOSSIM, a Concurrent Switch-Level Fault Simulator," ACM/IEEE Design Automation Conference, pp.715-719, 1985.

[7] R. E. Bryant, D. Beatty, K. Brace, k. cho, and T. Sheffler, "COSMOS: A Compiled Simulator for MOS Circuits," ACM/IEEE Design Automation Conference, pp.9-16, 1987.

[8] T. Hwang, C. Lee, W. Shen, and C, Wu, "A Parallel Pattern Mixed-Level Fault Simulator," ACM/IEEE Design Automation Conference, pp.

716-719, 1990.

[9] E. B. Eichelberger and T. W. Williams, "A Logic Design Structure for LSI Testability," ACM/IEEE Design Automation Conference, pp.462-468, 1977.

[10] E. S. Park and M. R. Mercer, "Switch-Level ATPG Using Constraint-Guided Line Justification," IEEE Proc. Int. Test Conference, pp. 616-625, 1993.

[11] K. J. Lee, C. Njinda, and M. Breuer, "SWiTEST :A Switch Level Test Generation for CMOS Combinational Circuits," ACM/IEEE Proc. Design Automation Conference, pp.26-29, 1992.

[12] A. Miczo, Digital Logic Testing and Simulation, Happer & Row, pp.154-155, 1986.

[13] P. Agrawal, "Automatic Modeling of Switch-Level Network Using Partial Orders," IEEE Trans. on Computer-Aided Design, vol.9, no.7, pp.696-707, 1990.

[14] N. K. Jha and S. Kundu, Testing and Reliable Design of CMOS Circuits, Chap.3, Kluwer Academic Pub., Boston, pp.57-63, 1990.

[15] J.A. Waicukauski, E. Eichelberger, D. Forlenza, E. Lindbloom, "Fault Simulation for Structured VLSI," VLSI Systems Design, pp.20-32, December 1985.

[16] F. Maamari and J. Rajski, "A Fast Simulation Based on Stem Regions," IEEE Proc. Int. Conference Computer-Aided Design, pp.170-173, 1988.

[17] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran," IEEE Proc. Int. Symp. Circuits System, 1985.

박 영 호

1986년 대전산업대학교 전자계산학과(학사)

1983년~현재 한국전자통신연구소 H/W개발환경연구실 선임기술원

관심분야:CAD, MCM설계, 컴퓨터 네트워크

손 진 우

1973년 영남대학교 전자공학과(학사)

1994년 아주대학교 전자공학과(박사)

1973년~1977년 공군정보장교

1978년~1983년 삼성반도체통신 설계기술과장

1984년~현재 한국전자통신연구소 H/W개발환경연구실장

관심분야:CAD/E, 데이타통신

박 은 세

1980년 서울대학교 전기공학과(학사)

1982년 한국과학기술원 전기 및 전자공학과(석사)

1989년 미국 텍사스주립대(어스틴) 전기 및 컴퓨터공학과(박사)

1982년~1995년 한국전자통신연구소 책임연구원

1989년~1990년 미국 Mentor Graphics사의 연구원

1990년~1991년 미국 Motorola사 선임 연구원

관심분야:VLSI CAD 및 테스트 설계