經路遲延故障 시뮬레이션의 效率的인 動的 메모리 使用에 관한 硏究

김 규 철†

요 약

집적회로의 집적도가 높아지고 성능이 향상됨에 따라 회로의 지연고장에 대한 관심이 날로 높아지고 있다. 회로의 지연고장은 게이트 지연고장과 경로지연고장으로 분류할 수 있는데, 이 논문에서는 경로지연고장 시뮬레이터에 대한 두 가지 동적 메모리 사용 방법을 제안하였다. 첫 번째 방법은 고착고장에 대한 동시 고장 시뮬레이션과 유사한 방식이며, 두 번째 방법은 고장기술자의 값이 X일 때 이를 고장리스트에 삽입하지 않는 묵시적-X 방식이다. 제안된 두 방식 중 묵시적-X 방식이 동적 메모리 사용과 시뮬레이션 시간 측면에서 효율적이었다.

A Study on the Efficient Dynamic Memory Usage in the Path Delay Fault Simulation

Kyu-Chull Kim†

ABSTRACT

As the circuit density of VLSI grows and its performance improves, delay fault testing of VLSI becomes very important. Delay faults in a circuit can be categorized into two classes, gate delay faults and path delay faults. This paper proposed two methods in dynamic memory usage in the path delay fault simulation. The first method is similar to that used in concurrent fault simulation for stuck-at faults and the second method reduces dynamic memory usage by not inserting a fault descriptor into the fault list when its value is X. The second method, called Implicit-X method, showed superior performance in both dynamic memory usage and simulation time than the first method, called Concurrent-Simulation-Like method.

1. 개 요

고장 시뮬레이션(fault simulation)은 설계과정의 중요한 부분이다. 검사패턴 생성(test pattern generation)과 고장 시뮬레이션을 위한 고장 모델(fault model)은 여러 가지가 있다. 회로 내의 어떤 라인(line)이 입력값에 무관하게 고정된 값을 가지면 그 라인이 고착고장(stuck-at fault)을 가지고 있다고 한다. 회로 내에 고착고장이 하나만 있는 단일 고착고장(single stuck-at fault) 모델은 가장 많이 사용되는 고장 모델이다. 어떤 회로가 정상 속도의 시스템 클락에는 오동작(malfunction)을 하다가 시스템 클락의 속도를 낮추었을 때 정상적인 동작을 하면 그 회로는 지연고장(delay fault)을 가지고 있다고 한다. 지연고장에는 게이트지연고장과 경로지연고장 두 가지가 있다. 게이트 지연고장은 게이트 입력과 출력에서의 완상승(slow-to-rise) 및 완하강(slow-to-fall) 지연고장을 말한다[1-4].

※ 본 연구는 단국대학교 대학연구비 지원에 의하여 수행되었음
† 정 회 원 : 단국대학교 전자컴퓨터공학부 교수
논문접수 : 1998년 5월 2일, 심사완료 : 1998년 9월 2일

경로지연고장은 회로 내의 어떤 경로를 통한 신호의 전파가 정해진 시간에 이루지지 않는 지연고장을 말한다[1-4]. 게이트 지연고장은 지역적(local)인 영향을 나타내고 경로지연고장은 전역적(global)인 영향을 나타낸다[5]. 고착고장은 회로의 논리적 기능(logical function)에 영향을 미치고 지연고장은 시스템의 동작 속도에 영향을 미친다[6]. 따라서 고성능 회로에서의 지연고장 검사는 대단히 중요하다.

동기식 순차회로(synchronous sequential circuit)의 고착고장에 대한 대부분의 고장 시뮬레이터는 병렬 고장 시뮬레이션(parallel fault simulation)[7] 또는 동시 고장 시뮬레이션(concurrent fault simulation)[8]에 바탕을 둔 것이다. 예를 들면, PROOFS[9]와 HOPES[10]는 병렬 고장 시뮬레이션에 바탕을 둔 고장 시뮬레이터이며, HYSIM[11]은 동시 고장 시뮬레이션에 바탕을 둔 고장 시뮬레이터이다.

이 논문에서는 병렬 고장 시뮬레이션에 바탕을 둔 HYSIM에 사용된 여러 기법[11,12]을 도입하여 동기식 순차회로에 대한 경로지연고장 시뮬레이터를 구현하였다. 또한 경로지연고장 시뮬레이션 수행에 사용되는 메모리의 양을 분석하여 메모리 사용을 줄이는 방법을 제안하였다.

이 논문의 구성은 다음과 같다. 2절에서는 경로지연고장 및 경로지연고장 검사를 기술하고 3절에서는 경로지연고장 시뮬레이터의 구현과 관련된 사항들을 자세히 기술한다. 또한 동적 메모리 사용이 효율적인 지연고장 시뮬레이터의 자료구조가 제안된다. 4절은 실험 결과 및 고찰을 싣고 있으며 5절에는 결론이 나와 있다.

2. 경로지연고장 및 경로지연고장 검사

2.1 경로지연고장

회로의 어떤 경로를 통한 신호의 전파 시간이 한 클락 사이클을 초과하게 되면 그 경로는 경로지연고장을 가지고 있다고 한다. 즉, 경로지연고장이 존재하면, 주입력(PI : Prinmary Input) 또는 의사입력(Pseudo Input)의 신호 변화는 주출력(PO: Primary Output) 또는 의사출력(Pseudo Output)에 정해진 시간 내에 도달하지 못하게 된다. 여기서 의사입력이란 플립플롭의 출력을 의미하고 의사출력이란 플립플롭의 입력을 의미한다. 회로 내에 경로지연고장이 존재하면 정상적인 시스템 클락의 속도에 회로가 올바르게 동작하지 못하며, 시스템 클락의 속도를 늦추어 클락사이클을 충분히 길게 해주면 정상적인 동작을 한다. 경로 지연의 검사는 이런 성질을 이용하여 이루어진다.

2.2 경로지연고장의 검사

경로지연고장의 검사는 다음의 3 단계로 이루어진다[13].

(1) 회로의 초기화

동기식 순차회로의 플립플롭(flipflop)은 미지 값(unknown value)을 가질 수 있다. 때에 따라, 검사를 수행하기 위해 이 플립플롭들이 기지 값(known value)을 갖도록 할 필요가 있다. 순차회로의 플립플롭이 기지 값을 갖도록 해주는 작업을 초기화(initialization)라고 한다. 초기화에는 검사 대상이 되는 순차회로의 순차 깊이(sequential depth)에 따라 하나 또는 여러 개의 벡터가 필요하다. 초기화는 다음 상태 변수가 플립플롭에 래치(latch)되기 전에 회로 내의 모든 값이 안정되도록 충분한 사이클 시간을 가지고 있는 느린 클락(slow clock)이 사용된다.

(2) 경로지연고장의 활성화

경로지연고장의 활성화(activation)는 정상적인 시스템 클락과 같은 속도의 빠른 클락(fast clock)을 사용하여 이루어진다. 만일 주입력(또는 의사입력)의 신호 변화가 경로지연고장 때문에 주출력(또는 의사출력)에 도달하지 못하면 경로지연고장이 활성화되었다고 한다. 활성화된 경로지연고장에 의한 영향이 주출력에 나타나면 그 경로지연고장이 검출된 것이고, 의사출력에 나타난 경우에는 다음 시간대(time frame)에서 전파되기 위하여 경로지연고장의 영향이 플립플롭에 래치(latch)된다.

(3) 경로지연고장의 전파

플립플롭에 래치된 경로지연고장의 영향은 주출력에 도달하여 검출될 때까지 검사패턴이 있는 동안 느린 클락 하에서 전파된다. 느린 클락의 속도는 초기화에서처럼 충분한 길이의 사이클 시간을 가져야 한다. 경로지연고장의 영향은 전파 도중 소멸되는 수도 있다. 이는 동시 고장 시뮬레이션에서의 수렴(convergence)과 같은 현상이다.

2.3 클라킹 스킴

앞에서 설명한 경로지연고장의 검사에서 빠른 클락과 느린 클락이 사용되었는데, 빠른 클락과 느린 클락의 조합을 클라킹 스킴(clocking scheme)[14]이라 부른다. 이 논문에서 취급하는 클라킹 스킴은 하나의 빠른 클락과 여러 개의 느린 클락으로 구성된다. 첫 클락으로 항상 느린 클락을 사용하면 길이 N인 벡터는 다음과 같이 N-1 개의 클라킹 스킴이 가능하다.

클라킹 스킴 1 : SFSSS. . .S

클라킹 스킴 2 : SSFSS. . .S

. . .

클라킹 스킴 N-1 : SSSSS . . . SF

클라킹 스킴의 S는 느린 클락을 나타내고 F는 빠른 클락을 나타낸다. 첫 클락으로 느린 클락만 허용하는 이유는 첫 클락에서의 입력 (또는 의사입력)의 천이(transition)가 정의되지 않기 때문이다.

3. 경로지연고장 시뮬레이션

경로지연고장 시뮬레이션은 경로지연고장의 검사와 동일한 i) 회로의 초기화 ii) 경로지연고장의 활성화 iii) 활성화된 경로지연고장의 전파의 세 단계로 구성된다. 이 절에서는 경로지연고장 시뮬레이션의 각 단계를 자세히 설명한다. 경로지연고장 시뮬레이션에는 게이트 지연고장 시뮬레이션과는 달리 복잡한 타이밍 시뮬레이션이 필요 없다[5].

3.1 회로의 초기화

초기화는 회로가 기지 상태(known state)를 갖도록 한다. 초기화에는 하나 또는 여러 개의 느린 클락이 사용된다. 느린 클락의 사이클 시간은 다음 상태 변수가 플립플롭에 래치되기 전에 회로내의 모든 신호 변화가 완료되도록 충분한 길이를 갖는다. 그러므로 초기화 단계에서는 경로지연고장이 활성화되지 않는다. 따라서 3-값 (0, 1, X)을 사용한 정상 회로 시뮬레이션으로 회로의 기지 상태를 얻는다.

3.2 경로지연고장의 활성화

경로지연고장의 활성화 과정은 고착고장 시뮬레이션의 고착고장의 활성화와 전혀 다르다. 경로지연고장 시뮬레이션에서는 주입력(또는 의사입력)에서의 신호 천이를 주출력(또는 의사출력)으로 경성 전파(robust propagation)[5] 시킨다. 경로지연고장 검사에서 신호 천이가 출력에 경성 전파되었다 함은 어떤 경로를 통한 신호 천이 전파의 지연에 의한 영향이 회로 내의 다른 경로를 통한 신호 전파 지연의 유무와 상관없이 출력에 나타남을 의미한다. 그리고 출력에 경성 전파된 천이는 경성 천이라고 부른다. 만일 주입력(또는 의사입력)에서의 신호 천이가 주출력(또는 의사출력)으로 경성 전파되면, 경로지연고장이 활성화되었다고 한다. 경로지연고장의 활성화에는 하나의 빠른 클락이 사용된다.

3.2.1 경로지연고장의 활성화를 위한 논리값 시스템

경로지연고장의 활성화를 시뮬레이션 하는 데에는 3-값 2-패턴 시뮬레이션이 필요하다. 2-패턴은 이전 시간대(time frame)에서의 값(초기값)과 현재 시간대에서의 값(최종값)을 결합시켜 생성한다.

2-패턴 시뮬레이션에는 다음 값들이 사용된다[5,14]. S0과 S1은 무변 값(steady value)을 나타낸다. 즉, 초기값과 최종값이 같으면 무변 값이 된다. T0(T1)은 1(0)에서 0(1)으로의 천이를 나타낸다. H0과 H1은 해저드(hazard)를 나타낸다. 즉, H0(H1)은 초기값 0(1)과 최종값 0(1) 사이에 미지값 X를 가지고 있다. 이 외에도 X0, X1, 0X, 1X, XX 등과 같은 여러 가지의 초기값과 최종값의 조합이 가능하다. 이 값들의 첫 문자는 초기값 그리고 두 번째 문자는 최종값을 나타낸다. 예를 들면, X0은 미지값을 초기값으로 가지고 있고 최종값으로 0을 갖는다.

3.2.2 경성 전파(robust propagation)

게이트의 입력에 도달한 경성 천이가 게이트의 출력으로 경성 전파되기 위한 조건은 다음과 같다. AND 또는 NAND 게이트의 한 입력에 경성 천이 T0이 도달하였다면, 다른 입력은 모두 S1을 가져야 하고, 경성 천이 T1이 도달하였으면 다른 입력들은 각각 S1, T1, X1, H1 중의 하나를 가져야 한다. OR 또는 NOR 게이트의 한 입력에 경성 천이 T1이 도달하였다면 다른 입력은 모두 S0을 가져야 하고, 경성 천이 T0이 도달하였다면 다른 입력들은 각각 S0, T0, X0, H0 중의 한 값을 가져야 한다. 입력의 경성 천이가 출력에 경성 전파되어야만 출력의 천이가 경성 천이가 된다. 주입력과 의사입력의 천이는 모두 경성 천이로 간주한다. 표 1은 위에 설명된 여러 입력을 가지고 있는 게이트에서 경로 밖의 입력(off-path input)이 가져야 하는

경성 전파의 조건을 요약한 것이다. 단일 입력을 가지고 있는 게이트의 입력에 경성 천이 T0 혹은 T1이 도달하면 이는 출력에 경성 전파된다.

〈표 1〉 경성 전파의 조건
〈Table 1〉 Robust propagation condition

천이	AND, NAND	OR, NOR
T0	S1	S0, T0, X0, H0
T1	S1, T1, X1, H1	S0

3.2.3 활성화된 경로지연고장의 영향을 플립플롭으로 래치시키기

경성 천이 T0 혹은 T1이 어떤 경로를 통하여 주출력에 도달하면, 그 경로의 지연고장이 검출된 것이고, 의사출력인 플립플롭에 도달하면 다음에 계속되는 느린 클락에 의해 전파되기 위하여 적절한 값이 플립플롭에 래치되어야 한다.

회로에 N개의 플립플롭 FF_1, FF_2, . . . ,FF_N이 있다고 가정하자. 만일 T0(T1)이 경로 P_j를 통하여 플립플롭 FF_i로 경성 전파되었다면, 이 플립플롭에 래치되어야 할 정상 값은 0(1)이고 경로지연고장에 의한 값은 1(0)이 된다. 경로 P_j의 지연고장을 전파하기 위하여 다른 플립플롭에 래치되어야 할 값도 결정하여야 한다. 어떤 천이가 경로 P_k를 통하여 FF_m에 도달한 경우, 경로지연고장 P_j에 대한 플립플롭 FF_m의 값은 경로지연고장 P_k의 유무에 따라 다르다. 그러므로 경로지연고장 P_j에 대한 FF_m의 값으로 미지 값 X가 래치되어야 한다. 만일 S0 (S1)이 FF_m에 도달하였다면, 경로지연고장 P_j에 대하여도 정상 값 0(1)이 FF_m에 래치되어야 한다. 만일 H0이나 H1이 FF_m에 도달하였다면 FF_m에는 고장 P_j에 대한 값으로 X가 래치되어야 한다. 다른 모든 활성화된 경로지연고장에서도 같은 방법으로 플립플롭에 래치될 값을 결정한다. 이렇게 결정된 값은 고장기술자에 기록되어 플립플롭의 고장리스트에 삽입된다.

3.3 경로지연고장의 전파

빠른 클락에 의하여 경로지연고장이 활성화되면 앞에서 설명한 것처럼 각 플립플롭에는 고장기술자로 구성된 고장리스트가 형성된다. 플립플롭에 형성된 고장리스트는 느린 클락에 의하여 전파되므로 동시 고장 시뮬레이터에서의 고장리스트와 유사하게 전파할 수 있다. 따라서 경로지연고장기술자의 전파에는 HYSIM의 고장기술자 전파 기법 [11, 12]을 사용한다.

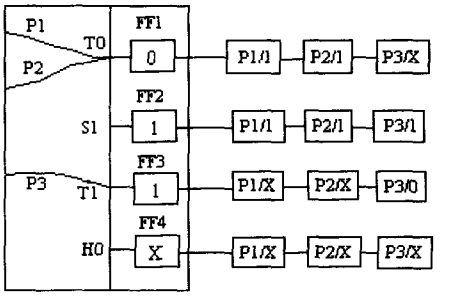
4. 경로지연고장 시뮬레이터의 메모리 사용

활성화된 경로지연고장은 다음 시간대에서 전파되기 위하여 각 플립플롭의 고장리스트에 기록되어야 한다. 고장리스트의 표현 방법에 따라 경로지연고장 시뮬레이터가 다루어야 할 고장기술자의 수가 달라지게 된다. 이 절에서는 활성화된 지연고장을 표현하는 고장리스트의 세 가지 표현 방법에 대하여 살펴본다.

4.1 직접적인 방식

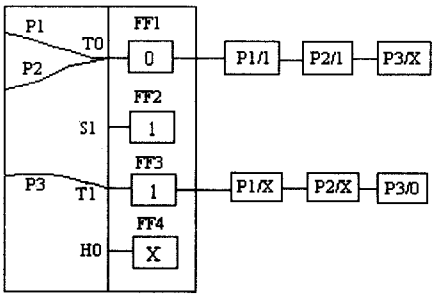
그림 1은 활성화된 각 경로지연고장에 대하여 플립플롭에 래치된 값들을 보이고 있다. 이 회로는 4개의 플립플롭 FF1,..., FF4를 가지고 있다. T0이 FF1에, S1이 FF2에, T1이 FF3에 그리고 H0이 FF4에 도달하였다고 가정한다. 활성화된 경로지연고장 P1과 P2가 FF1에 그리고 P3이 FF3에 나타났다고 하자. 각 플립플롭에는 고장리스트가 형성되었는데, 이 고장리스트는 모두 12 개의 고장기술자가 사용되었다. 이렇게 플립플롭에 형성된 고장리스트는 다음 시간대에서 느린 클락에 의하여 주출력이나 의사출력으로 전파된다. 고장리스트의 각 고장기술자는 전파 도중 소멸되지 않는다.

만일 K개의 경로지연고장이 활성화되었다면 각 플립플롭마다 K개의 고장기술자를 가지고 있는 고장리스트를 형성되어야 한다. 이 방식은 다음에 제안된 두 방식보다 더 많은 고장기술자를 가지고 있으므로 가장 많은 동적 메모리의 사용이 요구된다.



(그림 1) 직접적인 방식의 메모리 사용
(Fig. 1) Memory usage of Direct method

4.2 동시 고장 시뮬레이션 방식

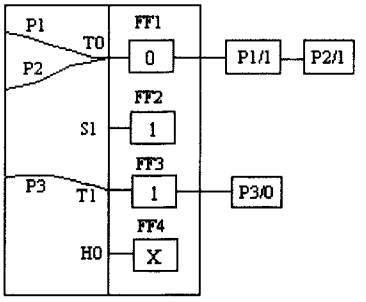


(그림 2) 동시 고장 시뮬레이션 방식의 메모리 사용
(Fig. 2) Memory usage of Concurrent-Simulation-Like method

고착고장을 취급하는 동시 고장 시뮬레이터는 활성화된 고착고장에 대한 게이트 값이 정상적인 게이트 값과 다른 경우에만 고장기술자를 생성하여 고장리스트에 삽입한다. 그러므로 고장리스트에 찾는 고착고장에 대한 고장기술자가 없으면 그 고착고장에 대한 게이트의 값은 정상 값과 같은 것으로 간주한다.

그림 3은 고착고장에 대한 동시 고장 시뮬레이션 방식으로 경로지연고장리스트를 나타낸 그림이다. 플립플롭 FF2와 FF4의 모든 활성화된 경로에 대한 고장기술자의 값이 정상 회로의 값과 같으므로 FF2와 FF4의 고장리스트는 생략되었다. 따라서 이 방식의 고장기술자의 수는 6개로 줄어든다. 이렇게 형성된 고장리스트는 다음 시간대에서 느린 클락에 의하여 주출력이나 의사출력으로 전파되거나 전파 도중 소멸된다.

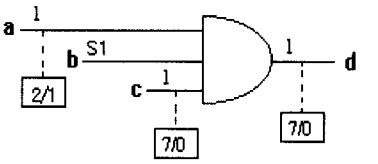
4.3 묵시적-X 방식



(그림 3) 묵시적-X 방식의 메모리 사용
(Fig. 3) Memory usage of Implicit-X method

그림 1을 살펴보면 경로지연고장 시뮬레이터의 많은 고장기술자가 미지 값 X를 나타내는 데에 사용되었음을 알 수 있다. 따라서 X 값을 저장하고 있는 고장기술자를 묵시적으로 표현한다면 많은 고장기술자를 없앨 수 있다. 즉, 고장기술자의 값이 X이면 그 고장기술자를 고장리스트에 삽입하지 않고 대신, 고장기술자의 값이 정상회로와 동일하더라도 값이 X가 아니면 그 고장기술자는 고장리스트에 삽입한다. 고장기술자를 전파할 때, 해당 고장기술자가 고장리스트에 없으면 그 경로지연고장에 대한 고장기술자의 값은 X인 것으로 간주한다. 그리고, 플립플롭의 값이 S0(S1)인 경우에는 활성화된 모든 경로지연고장에 대한 플립플롭의 값은 S0(S1)으로 한다. 느린 클락을 사용한 고장기술자의 전파에서도 정상 게이트의 값이 S0(S1)이면 전파중인 모든 경로지연고장에 대한 그 게이트의 값도 S0(S1)으로 결정한다. 따라서 정상 게이트의 값이 S0이나 S1인 경우에는 모든 경로에 대한 그 게이트에서의 고장기술자는 생략된다.

그림 3은 이 방식을 사용한 고장기술자의 표현을 보이고 있다.



(그림 4) 묵시적-X 방식의 AND 게이트를 통한 고장기술자의 전파
(Fig. 4) Fault descriptor propagation through an AND gate in Implicit-X method.

그림 4는 묵시적-X 방식 경로지연고장 시뮬레이터에서의 AND 게이트를 통한 고장기술자의 전파를 보이고 있다. 입력 a, b, c의 정상 값은 1, S1, 1이므로 출력 d의 정상 값은 1이 된다. 입력 a에 도달한 고장기술자의 경로번호는 2이다. 이 방식에서는 경로지연고장 2에 대한 입력 a, b, c의 값은 각각 1, S1, X가 된다. 따라서 경로지연고장 2에 의한 출력 d의 값은 X가 되므로 경로지연고장은 출력 d에 전파되지 않는다. 그러나 경로지연고장 7에 대하여, 입력 a, b, c의 값은 각각 1, S1, 0이므로 경로지연고장 7에 의한 출력 d의 값은 0이 되므로 출력 d로 전파된다. 즉, 고장리스트의

각 고장기술자는 시뮬레이션 과정 중 경로지연고장 7처럼 전파되거나 경로지연고장 2처럼 소멸된다.

경로지연고장의 전파에서 1과 S1이 구분되어 사용되었는데, S1은 경로지연고장의 활성화 단계에서 주입력이나 의사입력에서 발생하며 게이트의 제어 값(controlling value)으로 사용된 경우에는 게이트의 출력으로 전파된다. 0과 S0도 마찬가지로 구분되어 사용된다. 정상회로의 게이트 값이 S0이나 S1인 경우, 그 게이트의 고장리스트는 비어 있다.

5. 실험 결과

그림 2와 3의 고장기술자와 그림 1의 고장기술자를 비교하여 보면, 그림 2와 3의 고장기술자는 그림 1의 직접적인 방식에 의한 고장기술자의 부분 집합임을 알 수 있다. 따라서, 이 논문에서는 그림 2의 동시 고장 시뮬레이션 방식과 그림 3의 묵시적-X 방식을 사용한 경로지연고장 시뮬레이터만을 구현 비교하였다. 두 경로지연고장 시뮬레이터는 C 언어를 사용하여 구현되었으며, SunOS 4.1.3을 운영체제로 사용하는 SparcStation 5에서 ISCAS-89 벤치마크 회로[15]에 대하여 실행되었다. 이 실험에 사용된 검사 벡터는 순차회로의 고착고장에 대한 검사 패턴 생성기인 STG-3[16]에 의해 생성된 것을 사용하였다. 그리고 가능한 클라킹 스킴을 모두 순서에 따라 사용하였다.

표 2는 동시 고장 시뮬레이션 방식과 묵시적-X 방식의 실험 결과를 보이고 있다. STG-3에 의해 생성된 벡터의 경로지연고장에 대한 고장 검출율(fault coverage)은 고착고장에 대한 고장 검출율에 비해 매우 낮은 것으로 나타났다. 이는 STG-3의 고착고장에 대한 검사 패턴을 생성했기 때문이다. 경로지연고장 검출을 목적으로 생성된 패턴을 사용하면 경로지연고장에 대한 고장 검출율은 더 높아지겠지만 경성 전파 경로지연고장의 검출율은 근본적으로 낮다[17]. ISCAS-89 회로중 S838은 고착고장과 지연고장 모두에 대하여 낮은 고장 검출율을 나타내었다. 시뮬레이션에 사용된 클라킹 스킴이 고장 검출율을 증가시키면 그 클라킹 스킴을 유효 클라킹 스킴으로 계산하였다.

표 3은 이 실험에 대한 동시 고장 시뮬레이션 방식과 묵시적-X 방식의 ISCAS-89 회로에 대한 동적 메모리 사용량과 시뮬레이션 시간을 보이고 있다. S35932에 대한 동시 고장 시뮬레이션 방식의 데이터가 누락되었는데, 이는 동시 고장 시뮬레이션 방식의 메모리 사용량이 너무 많아 시뮬레이션을 수행할 수 없었기 때문이다.

메모리 사용량은 묵시적-X 방식이 동시 고장 시뮬레이션 방식보다 모든 회로에 대하여 우수한 결과를 보였다. 동시 고장 시뮬레이션 방식에 대한 묵시적-X 방식의 메모리 사용량 비율은 최소 0.01(S5378, S838)에서 최대 0.73(S1238)이었다. 시뮬레이션 시간에서도 묵시적-X 방식이 동시 고장 시뮬레이션 방식보다 모든 회로에 대하여 우수한 결과를 보였다. 동시 고장 시뮬레이션 방식에 대한 묵시적-X 방식의 시뮬레이션 시간의 비율은 0.41(S5378)에서 0.86(S838)이었다. 동시 고장 시뮬레이션 방식보다 묵시적-X 방식의 메모리 사용량이 적다는 것은 경로지연고장 시뮬레이션에서

〈표 2〉 ISCAS-89 회로에 대한 STG-3 벡터의 고장 검출율
〈Table 2〉 Fault Coverages of STG-3 Vectors for ISCAS-89 Circuits

회로	벡터 길이	총 경로 고장 수	검출된 경로 고장 수	고장검출율 (지연고장)	고장검출율 (고착고장)	유효 클라킹 스킴 수
S298	162	462	56	12.12	85.71	25
S349	91	730	91	12.47	95.71	40
S526	754	820	47	5.73	75.32	23
S713	107	4200	150	3.57	80.90	53
S832	377	1012	189	18.48	81.38	80
S838	137	3428	25	0.67	29.64	14
S1238	349	3022	695	22.10	94.69	169
S1494	469	1952	379	19.01	91.10	121
S5378	408	20428	1470	6.57	74.02	278
S35932	86	261674	3934	1.50	87.99	68

발생하는 많은 고장기술자의 값이 X임을 의미한다. 동시 고장 시뮬레이션 방식에서는 고장기술자의 값이 X인 경우에도 고장기술자를 고장리스트에 포함시켜야 하므로 메모리 사용량이 묵시적-X 방식보다 많게 된다.

또한 시뮬레이션 시간도 묵시적-X 방식이 동시 고장 시뮬레이션 방식보다 우수한데 이는 고장리스트의 길이가 길수록 고장리스트를 처리하는 시간이 길어지므로 메모리 사용을 줄인 영향이라고 볼 수 있다. 예외적으로 S838회로에 대해서는 묵시적-X 방식의 동적 메모리 사용량의 비율이 최소임에 반하여 시뮬레이션 시간 비율은 최대였다. S838은 고착고장은 물론 경로지연고장에 대해서도 STG-3에 의해 생성된 검사패턴의 고장 검출율이 극히 낮은 특이한 회로이다.

〈표 3〉 동시 고장 시뮬레이션 방식과 묵시적-X 방식의 메모리 사용량과 시뮬레이션 시간

〈Table 3〉 Memory usage and simulation time of Concurrent-Simulation-Like method and Implicit-X method.

회로	메모리 사용량 (KB)			시뮬레이션 시간 (초)		
	동시적	묵시적-X	비율	동시적	묵시적-X	비율
S298	1.19.2	2.4	0.13	16.02	11.68	0.73
S349	25.2	2.4	0.04	7.52	6.18	0.82
S526	16.8	2.4	0.14	649.32	383.52	0.59
S713	10.8	6.0	0.56	28.38	26.23	0.92
S832	19.2	4.8	0.25	261.20	213.10	0.82
S838	236.4	2.4	0.01	56.02	48.33	0.86
S1238	18.0	13.2	0.73	277.58	224.48	0.81
S1494	159.6	9.6	0.06	575.10	422.42	0.73
S5378	5730.0	33.6	0.01	3109.47	1259.93	0.41
S35932	N/A	172.8	-	N/A	405.88	-

6. 결 론

동적 메모리를 효율적으로 사용하기 위한 경로지연고장 시뮬레이터를 제안하고, 제안된 방식의 효율성을 입증하기 위하여 두 가지 서로 다른 방식의 경로고장 시뮬레이터를 구현하여 성능을 비교하였다. 하나는 동시 고장 시뮬레이션과 유사한 방식으로 고장리스트를 형성하여 활성화된 고장을 전파하고 다른 하나는 고장리스트의 고장기술자의 값이 X일 때는 고장기술자를 고장리스트에서 생략하는 묵시적-X 방식이다. 두 방식 모두에서 초기화에는 3-값 시뮬레이션을 사용하고 경로지연고장의 활성화에는 3-값 2-패턴 시뮬레이션을 사용하였다. 그리고 활성화된 경로지연고장의 전파에는 동시 고장 시뮬레이터의 일종인 HYSIM의 기법을 사용하였다.

구현된 두 방식 중 묵시적-X 방식이 메모리 사용과 시뮬레이션 시간 면에서 동시 시뮬레이션 방식보다 우수하였다. 이는 경로지연고장 시뮬레이션에서 활성화된 경로지연고장에 대한 회로의 많은 고장기술자가 X 값을 갖음을 의미한다. 묵시적-X 방식을 사용하여 고장기술자의 수를 줄인 결과 시뮬레이션 시간도 동시 고장 시뮬레이션 방식보다 짧아짐을 확인하였다.

참 고 문 헌

[1] T. Hayashi et al., "A Delay Test Generator for Logic LSI," *Proc. 14th Intl. Conf. Fault Tolerant Computing*, pp.146-149, June 1984.

[2] T. M. Storey and J. W. Barry, "Delay Test Simulation," *Proc. 14th Design Automation Conf.*, pp.492-494, June 1977.

[3] J. D. Lesser and J.J. Shedletsky, "An Experimental Delay Test Generator for LSI Logic," *IEEE Trans. Computers*, pp.235-248, March 1980.

[4] Z. Barzilai and B. K. Rosen, "Comparison of AC Self-Testing Procedures," *Proc. 1983 Intl. Test Conference*, Oct. 1983.

[5] G. L. Smith, "Model for Delay Faults Based upon Paths," *IEEE Intl. Test Conference*, pp.342-349, 1985.

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

[7] S. Seshu, "On an Improved Diagnosis Program," *IEEE Trans. on Elect. Comput.*, Vol.EC-12, pp.76-79, Feb. 1965.

[8] E. G. Ulrich and T. Baker, "The Concurrent Simulation of Nearly Identical Digital Networks," *Design Automation Workshop*, Vol.6, pp.145-150, April 1973.

[9] W. T. Cheng and J. H. Patel, "PROOFS: A Super Fast Fault Simulator for Sequential Circuits," *Journal of Electronic Testing: Theory and Applications*, pp.7-13, 1990.

[10] H. K. Lee and D. S. Ha, "HOPE: An Efficient

Parallel Fault Simulator for Synchronous Sequential Circuits," *IEEE Design Automation Conference*, pp.336-340, June 1992.

[11] K. Kim and K. K. Saluja, "HYSIM: Hybrid Fault Simulation for Synchronous Sequential Circuits," VLSI Design, *An International Journal of Custom-Chip Design, Simulation and Testing*, Vol.4, No.3, pp.181-197, July 1996.

[12] K. Kim and K. K. Saluja, "Reduction of Dynamic Memory Usage in Concurrent Fault Simulator for Sequential Circuits," *Asian Test Symposium*, 1992.

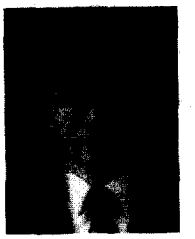
[13] Y. K. Malaiya, R. Narayanaswamy, "Modeling and Testing for Timing Faults in Synchronous Sequential Circuits," *IEEE Design & Test of Computers*, pp.62-74, Nov. 1984.

[14] I. Pomeranz, L. N. Reddy and S. M. Reddy, "SPADES: Simulator for Path Delay Faults in Sequential Circuits," *Euro-DAC*, Sept. 1992.

[15] F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," *Proc. Intl. Symp. on Circuits and Systems*, pp.1929-1934, May 1989.

[16] W. Cheng and S. Davidson, "Sequential Circuit Test Generation (STG) Benchmark Results," *Intl. Symp. on Circuits and Systems*, pp.1938-1941, May 1981.

[17] C. J. Lin and S. M. Reddy. "On Delay Fault Testing in Logic Circuits," *IEEE Trans. on Computer-Aided Design*, pp.694-703, Sept. 1987.



김 규 철

kckim@ns.dankook.ac.kr

1978년 서울대학교 자연과학대학 물리학과 졸업(학사)

1980년 서울대학교 물리학과 대학원 졸업(이학석사)

1986년 미국 위스콘신 대학 졸업 (전기컴퓨터공학 석사)

1992년 미국 위스콘신 대학 졸업(전기컴퓨터공학 박사)

1992년 미국 위스콘신 대학 연구원

1993년 삼성전자 마이크로본부 선임연구원

1993년~현재 단국대학교 전자컴퓨터공학부 조교수

관심분야 : VLSI 설계, VLSI 검사, 고장 시뮬레이션, 검사패턴 자동생성, Built-In Self Test, Design for Testability, 지연고장검사