論文96-33A-6-25

MOS 로직 및 타이밍 시뮬레이션을 위한 데이타 구조 및 알고리즘

(A data structure and algorithm for MOS logic-with-timing simulation)

孔 鎭 興 *

(Jin-Hyeung Kong)

요 약

본 논문에서는 MOS 로직 및 타이밍 시뮬레이션의 계산 속도 및 정확도면의 성능 향상을 위한 새로운 네트워크 모델링의 데이타 구조와 검증 방법을 기술한다. 드라이버-로드 네트워크의 로직 및 타이밍을 효율적으로 시뮬레이션 하기 위한 (1) 스위치와 노드의 상호 연결 토포로지가 유지되는 트리 데이타 구조와 (2) 데이타 구조를 효과적으로 처리하기 위한 대수적 모델링, (3) 새로운 모델링에 대한 선형 저항 및 용량 특성의 검증 알고리즘을 개발하였다. 제안된 상위 모델링은 선형 스위치 레벨 모델에 대하여 구조적(structural) 및 기능적(functional) 호환성을 유지하여, 디지탈 MOS 회로의 로직 및 타이밍을 혼합 레벨에서 시뮬레이션할 수 있도록 한다. 본 연구에서는 제안된 데이타 구조 및 대수적 검증 방법을 기존의 시뮬레이터 RSIM[1]에 결합시켜 새로운 혼합 레벨 로직 및 타이밍 시뮬레이터 MIXIM을 구현하였다. 실험 결과를 통하여 MIXIM이 RSIM과 비교해서 계산 속도와 타이밍 정확성을 개선한 것과 특히 드라이버-로드 네트워크에 대한 타이밍 시뮬레이션은 아날로그 시뮬레이터 SPICE[2]에 대하여 17%의 오차 범위를 갖는 것을 확인하였다.

Abstract

This paper describes a data structure and evaluation algorithm to improve the performances MOS logic-with-timing simulation in computation and accuracy. In order to efficiently simulate the logic and timing of driver-load networks, (1) a tree data structure to represent the mutual interconnection topology of switches and nodes in the driver-load network, and (2) an algebraic modeling to efficiently deal with the new representation, (3) an evaluation algorithm to compute the linear resistive and capacitive behavior with the new modeling of driver-load networks are developed. The higher modeling presented here supports the structural and functional compatibility with the linear switch-level, to simulate the logic-with-timing of digital MOS circuits at a mixed-level. This research attempts to integrate the new approach into the existing simulator RSIM[1], which yields a mixed-level logic-with-timing simulator MIXIM. The experimental results show that (1) MIXIM is a far superior to RSIM in computation speed and timing accuracy; and notably (2) the timing simulation for driver-load networks produces the accuracy ranged within 17% with respect to the analog simulator SPICE[2].

* 正會員, 光雲大學校 컴퓨터工學科 新技術硏究所

(Kwangwoon Univ. Computer Eng. Dept. Institute of New Technology)

※ 본 연구는 한국과학재단의 '92 일반 목적 기초 연구 과제(921-1100-017-2)로 지원되었음.

接受日字: 1995年1月12日, 수정완료일: 1996年4月29日

I. 서 론

디지탈 MOS 집적회로 설계를 검증하기 위해서는 양방향 신호 전달과 동적 전하 저장 및 저항 비율 등 회로 고유의 동작을 분석하는 것이 필요하다. 이러한 MOS 회로의 동작을 게이트나 상위 레벨에서 분석하는 것은 모델링의 한계 때문에 정확성이 제한되어, 실

제 회로 구조와 유사한 스위치와 회로 레벨의 모델링과 분석 방법을 요구한다. 이때 구체적인 레벨 및 방법은 설계 검증이 요구하는 정확성과 수반되는 경비/오버헤드간의 상호 trade-off를 고려해서 결정하게 된다. 설계된 회로를 분석할때 회로 레벨의 전기적 모델링을 이용하면 결과의 정확성은 좋지만 오버헤드가 매우 크고, 스위치 레벨의 논리적 모델링은 결과의 정확성은 떨어지며 오버헤드가 작게 된다. 집적회로의 크기 및 복잡성이 증가함에 따라 정확성과 오버헤드가 절충된 설계 검증 방법이 요구되는데, 스위치 레벨 로직 및 타이밍(logic-with-timing) 시뮬레이션이 이러한 역할을 수행하고 있다[3]. 이 방법은 스위치 레벨 모델의 스위치와 노드에 대하여 선형 저항과 용량 특성을 부여하고 디지탈 MOS 회로를 RC 네트워크로 모델링 및 시뮬레이션하여 네트워크의 정상 상태(steady-state)와 타이밍 동작(timing behavior)을 계산한다. 스위치 레벨 로직 및 타이밍 시뮬레이션은 기존 스위치 레벨 방법의 한계인 타이밍 정보를 해결하면서 회로 레벨 방법에 비하여 검증 속도를 대폭 개선함으로써, 최근의 VLSI 및 ULSI의 레이아웃 설계를 검증할 수 있는 방법으로 이용된다.

RSIM[11]은 대표적인 스위치 레벨 로직 및 타이밍 시뮬레이터이다. 본 연구에서는 로직 및 타이밍 시뮬레이션을 위한 선형 스위치 레벨 정확성을 산출하는 상위 레벨 모델링을 스위치 레벨 네트워크에 부분적으로 적용하여 혼합 레벨 시뮬레이션을 통한 계산 속도의 향상을 이루고자 한다. 스위치 레벨 네트워크에 대한 상위 레벨 모델링은 시뮬레이션의 전처리 단계에서 네트워크 표현을 검증에 적합한 형태로 정적 변환 과정을 가리키는데, 기존에는 패스 검색을 이용한 (불리안) 심볼 분석[4,5,6]과 기능적 추상화[7,8]등의 스위치 레벨 로직 시뮬레이션에서 이용되었다. 그러나 이들 연구는 이산 스위치 레벨 모델링의 네트워크에 대한 게이트 및 기능 단위의 추출을 통해 정상 상태를 시뮬레이션하지만 타이밍에 관한 동작을 예측할 수 없는 제한을 갖는다. 이러한 문제를 해결하고자 본 연구는 선형 스위치 레벨 네트워크에서 드라이버-로드(driver-load) 네트워크의 토포로지를 상위 레벨에서 기술하는 데이타 구조를 제안하고, 스위치 레벨 로직 및 타이밍 시뮬레이션의 정확성을 산출하기 위한 대수적 모델링과 검증 방법을 개발하였다. 새로운 데이타 구조는 MOTIS[9]의 검증 트리 표현[10]이나 블리안 함수에 대응하는 DAG(directed acyclic graph) 표현[5]등의 이산 처리와 달리 선형 스위치 레벨 모델링으로부터 저항 및 용량 특성을 연산하기 위하여 스위치 레벨의 순회 과정과 동일한 처리가 가능하도록 데이타 구조에 스위치 및 노드의 상호 연결(mutual interconnection) 토포로지 정보를 저장하였다. 이를 위하여 네트워크의 직렬 토포로지에서 파워 노드로부터 출력 노드까지 물리적 연결의 레벨(level) 정보를 AND 트리의 상속자들의 등록 순서로써 이용한 "AND 순서 토포로지 트리" 구조를 설계하였다. 이 트리 데이타 구조는 스위치로 구성된 단(leaf) 노드와 네트워크 노드에서의 직렬 및 병렬 연결 정보를 표현하는 내부(internal) 노드로 구성된다. 이 트리 구조는 순차 연산식으로 저장할 수 있는데, 연산자는 네트워크의 직렬/병렬 토포로지를 표현하며 피연산자는 스위치나 노드에 대응된다. 이 순차 연산식을 통하여 네트워크를 구성하는 스위치 및 노드들의 선형 저항/용량 특성을 처리하기 위해서 토포로지 연산의 계산 규칙을 부울 대수를 확장한 대수 구조[11]를 이용하여 정의하였다. 정의된 대수 구조에 따른 순차 연산식 처리는 드라이버와 로드 네트워크의 실질적인 저항 및 용량 특성을 산출해서, 출력 노드에서의 Thevenin 등가회로 구성[11]과 정상 상태 및 지연 시간의 연산을 가능토록 한다. 이와 같은 드라이버와 로드 네트워크에 대한 선형 저항 및 용량 특성의 대수적 검증은 스위치 레벨 로직 및 타이밍 시뮬레이션의 정확성을 상위 레벨 시뮬레이션을 통해 얻을 수 있도록 한다.

본 논문에서 제안된 데이타 구조와 모델링 및 검증 알고리즘을 기존의 스위치 레벨 시뮬레이터 RSIM에 결합시켜 새로운 혼합 레벨 시뮬레이터 MIXIM을 구현하였다. MIXIM은 디지탈 MOS 회로에 대한 정상 상태와 타이밍 예측을 위해 RSIM을 개선한 로직 및 타이밍 시뮬레이터이다. 성능 개선을 확인하기 위하여 2nd Booth's pipelined multiplier 회로 설계에 대하여 RSIM과 MIXIM의 계산 속도를 비교하였고, 타이밍의 정확성은 MOS 로직 게이트에 대해서 아날로그 회로 시뮬레이터 SPICE[12] 결과를 기준으로 MIXIM과 RSIM의 시뮬레이션 편차를 각각 실험하였다.

다음 장에서는 본 논문의 연구 배경을 소개하고 Ⅲ장은 새로운 데이타 구조와 대수적 모델링에 대하여 기술하며, Ⅳ장은 드라이버와 로드 네트워크의 검증 알고리즘을 설명한다. Ⅴ장에서는 제안된 방법의 실험 결

과를 보이고, 마지막으로 결론을 VI장에서 맺는다.

II. 연구 배경

스위치 및 회로 레벨에서의 설계 검증 방법과 특성은 다음과 같이 요약된다 :

(1) 아날로그 회로 시뮬레이션(circuit simulation) [2,12]

DC/AC/Transient 분석, 비선형회로 모델, 정적 및 동적 분석, 미분방정식

(2) 아날로그 타이밍 시뮬레이션(timing simulation) [9,13,14]

Transient 분석, 비선형회로 모델, 동적 분석, Relaxation

(3) 스위치 레벨 타이밍 검증(timing verification) [15,16]

Worst-case 지연 시간 분석, 선형 스위치 모델, 정적 분석, 패스 추적

(4) 스위치 레벨 로직 및 타이밍 시뮬레이션(logic with timing simulation) [13,17-19]

정상 상태와 지연 시간 분석, 선형 스위치 모델, 동적 분석, 등가 회로 유도

(5) 스위치 레벨 로직 시뮬레이션(logic simulation) [5-8,20-22]

정상 상태 분석, 이산 스위치 모델, 동적 분석, 그래프 및 반복적 방법 등

이 설계 검증 방법들의 시뮬레이션 속도는 (1)에서 (5) 순으로 증가하며, 정확성은 같은 순서에 따라 감소하게 된다. (5)의 스위치 레벨 로직 시뮬레이션은 MOS 회로 고유의 동작을 정상 상태로서 기능 분석할 수 있으나, 타이밍에 관련된 회로의 동적 행위를 예측할 수 없는 한계를 갖는다. 반면에 (1) 아날로그 회로 시뮬레이션은 매우 정확한 정적 및 동적 결과의 계산이 가능하나, 분석을 위한 미분 방정식의 구성 및 해결의 오버헤드가 과중한 문제를 갖고 있다. 이러한 두 방법의 사이에서 설계자가 요구하는 스위치 레벨 로직 시뮬레이션의 검증 속도와 아날로그 시뮬레이션의 정확성을 적절히 만족시키기 위한 검증 방법이 타이밍 시뮬레이션이다[13]. 이를 위하여 (2)와 (4)의 두 접근 방법이 제안되었는데, (2) 아날로그 타이밍 시뮬레이션은 아날로그 시뮬레이션의 적용 범위를 제한시키면서 계산 속도를 개선한 방법으로서 연산 결과인 아날로그 파형을 디지탈 상태로 변환시킨다. (4)의 스위치 레벨 로직 및 타이밍 시뮬레이션은 디지탈 시뮬레이션에 지연 시간 모델을 결합시킨 방법으로서 디지탈 정상 상태 및 타이밍에 관한 결과를 예측한다. 집적회로의 크기 및 복잡성이 급격히 증대함에 따라 스위치 레벨 로직 및 타이밍 시뮬레이션은 아날로그 타이밍 시뮬레이션에 비하여 보다 큰 관심을 받게 되었는데, 이는 CPU 및 메모리면에서 오버헤드를 크게 절약하면서 설계 검증에 필요한 정상 상태 및 타이밍 행위를 적절하게 예측할 수 있기 때문이다. 이외에도 회로의 타이밍 특성이 설계 명세(specification)에 적합한지를 확인하기 위해 critical path의 정적 분석을 수행하는 (3)의 스위치 레벨 타이밍 검증 방법 등이 디지탈 MOS 회로의 설계 검증에 이용될 수 있다.

스위치 레벨 로직 및 타이밍 시뮬레이터로서 대표적인 RSIM[11]은 사건-구동(event-driven) 방식의 시뮬레이션을 수행한다. 디지탈 MOS 회로는 저항성 스위치들로 연결된 용량성 노드들의 단위로 표현되며, 노드의 신호는 레벨(0/1/X)과 강도(선형 저항 및 용량 임피던스)로 나타낸다. 노드는 접지된 캐패시터(C)로써 MOS 회로의 연결선을 모델링하고, MOS 트랜지스터는 게이트 노드의 상태, 사용 방법(context), 크기 및 타입 등에 따라 결정되는 저항(R) 특성과 양방향 신호 전달 능력을 갖는 스위치로 기술한다. 사건의 발생에 의하여 스위치들의 저항값이 결정되고 네트워크는 스위치-게이트와 OFF 스위치를 경계로 스위치와 노드로 구성된 "stages"[13,16] 혹은 "cluster"[18]들로 분할된다. 이 부네트워크(subnetwork)는 분석의 기본 단위로서 ON 스위치를 통해 전기적으로 연결된 모든 노드를 포함한다. 검증 과정에서 분석의 단위인 stage의 각 노드에 대하여 Thevenin 등가 회로가 유도되어 노드의 정상 상태와 저항 및 용량 특성, RC 지연 시간 등이 결정된다.

집적회로 기술의 지속적인 발전은 VLSI 및 ULSI의 개발을 가져왔으며, 이는 스위치 레벨 로직 및 타이밍 시뮬레이션에 대한 성능 향상의 요구를 계속하여 RSIM의 성능을 개선한 연구들이 발표되었다. IRSIM[18]은 타이밍 및 전하 공유의 모델링을 개선하고 스위치 레벨에서 증분식 시뮬레이션(incremental simulation)을 시도해서 검증 속도를 향상시켰다. MOS 회로에 국한되는 RSIM의 검증 범위를 해결하

고자 Mom[19]은 일반적인 구간적 선형(piecewise linear) 모델을 개발하여 시뮬레이션 적용 대상을 CMOS 전체, BiCMOS 및 ECL까지 확장시켰다. 본 연구는 MOS 로직 게이트의 선형 스위치 레벨 네트워크에 대한 상위 레벨 모델링 및 검증 알고리즘을 개발하여, 설계 검증에서 혼합 레벨 시뮬레이션의 장점인 CPU 시간과 메모리의 오버헤드 감소를 얻고자 한다. 또한 이 상위 레벨 모델링을 통해 로직 네트워크 부분에 대한 RSIM의 타이밍 예측을 개선하여 선형 스위치 레벨의 정확성을 산출하고자 한다.

Ⅲ. 데이타 구조와 모델링

디지탈 MOS 회로를 스위치 레벨 네트워크로 기술하면 구조 특성면에서 수직(vertical) 및 수평(horizontal) 네트워크로 분할된다. 신호 전달 기능을 수행하는 수평 네트워크로는 전도(transmission) 게이트, 다중화기(multiplexor) 및 캐리 체인(carry chain) 등이 있다. 수직 네트워크에서는 VDD 및 GND 파워 노드로부터 출력 노드까지를 구성하는 로드(load) 및 드라이버(driver) 네트워크의 전도 상태가 스위치 게이트들의 로직 함수에 따라 결정되어 출력 노드의 신호를 정한다.

1. 트리 데이타 구조

로직 기능을 수행하는 수직 네트워크의 드라이버와 로드 네트워크에서 직렬 및 병렬 연결 토포로지가 AND OR 불리안 함수와 대응되는 것은 잘 알려져 있다. 대응 관계를 바탕으로 네트워크의 로직 기능을 검증하기 위한 트리 형태의 데이타 구조를 이용한 연구[10]도 MOTIS에서 비롯되어 많이 활용되고 있다. 그러나 이들 방법은 트리 데이타 구조를 통해서 네트워크의 기능 및 동작을 이산 논리적으로 처리하기 때문에 타이밍 정보를 위해 필요한 선형 저항 및 용량 특성을 검증하지 못하는 문제를 갖는다. 이와 같은 한계는 드라이버 및 로드 네트워크에 대한 기존의 트리 데이타 구조가 네트워크의 스위치 및 노드들의 상호 연결 정보를 유지하지 못해서 데이타 구조로부터 네트워크를 복원할 때 원래의 네트워크와 기능은 같지만 각 스위치와 노드의 주변 네트워크 구성이 변경되는 토포로지의 동질성(isomorphism) 상실을 가져오기 때문이다. 따라서 기존 트리 데이타 구조를 이용한 처리가 스위치 레벨 네트워크에 대한 직접적인 순회 과정과 대응될 수 없다는 것을 뜻하며, 네트워크에 대한 선형 스위치 레벨의 저항 및 용량 특성을 분석하는 것을 제한하게 된다.

본 연구는 드라이버와 로드 네트워크의 토포로지에 대응하는 데이타 구조를 다음과 같은 트리 형태로 제안한다. 트리 데이타 구조에서 스위치들은 항상 단(leaf) 노드에 위치하며, 네트워크 노드에서 토포로지는 AND와 OR 연결 함수로 표현되어 트리의 내부(interior) 및 뿌리(root) 노드를 구성한다. 트리의 OR 노드는 상속자(successor)들이 순서와 관계없이 병렬 연결된 상속자들의 토포로지를 표현하며, AND 노드는 왼쪽부터 오른쪽으로 순서가 부여되어 파워 노드쪽에서 출력 노드 방향으로 순서대로 직렬 연결된 토포로지를 나타낸다. 트리 구조는 AND 노드에 대해서 순서 정보를 갖고 토포로지 정보를 표현하기 때문에 "AND-순서 토포로지 트리(AND ordered topology tree)"라 부른다.

2. 데이타 구조 vs. 네트워크

스위치 레벨 네트워크는 전처리 과정에서 수직 및 수평 네트워크의 연결로 분할된다. 이 과정에서 수직 네트워크의 스위치들에 대해서 파워 노드를 소스(source)로 하고 출력 노드를 싱크(sink)로 하는 신호 전달 방향을 가정해서 정렬(alignment)이 이루어진다. 이때 n/p형 스위치의 데이타 노드는 파워 노드쪽을 source 터미날로 하고 출력 노드쪽을 drain 터미날로 정렬하게 된다. 실제로 이러한 방법은 타이밍 검증 방법[15,16]에서 MOS회로의 모든 트랜지스터 방향성에 대한 flow tag을 지정한 것과 유사한데, 본 연구에서는 드라이버 및 로드 네트워크에 대하여 적용하였다. 그림 1은 분할된 수직 네트워크의 스위치들이 정렬된 예를 보인다. 전처리된 방향성(directed) 수직 네트워크에 대하여 트리 데이타 구조는 그림 2(a)의 과정을 통해 유도된다. 파워 노드로부터 시작되는 네트워크 순회(network traversal)는 기본적으로 Breadth First Search(BFS) 알고리즘에 따라 스위치들을 소스에서 드레인 터미날 방향으로 순차적으로 횡단한다. 각 노드에 대한 횡단은 노드로 향한 가장 긴 패스의 순회가 끝난 후에 진행된다. 이와 같은 "Nested BFS (NBFS)" 네트워크 순회 과정에서 트리 구조는 직렬 및 병렬 토포로지 노드에 상속자를 가입시키는 AND

및 OR 가입(affiliation) 기본 작업을 조합해서 유도된다. 이 과정에 따라서 드라이버와 로드 네트워크로부터 AND 순서 토포로지 트리 데이타 구조가 각각 유도된다. 단노드에 위치한 스위치들에 대한 NBFS의 횡단 순서가 명기되어 그림 1의 수직 네트워크를 표현하는 두 토포로지 트리 구조가 그림 3에 보인다.

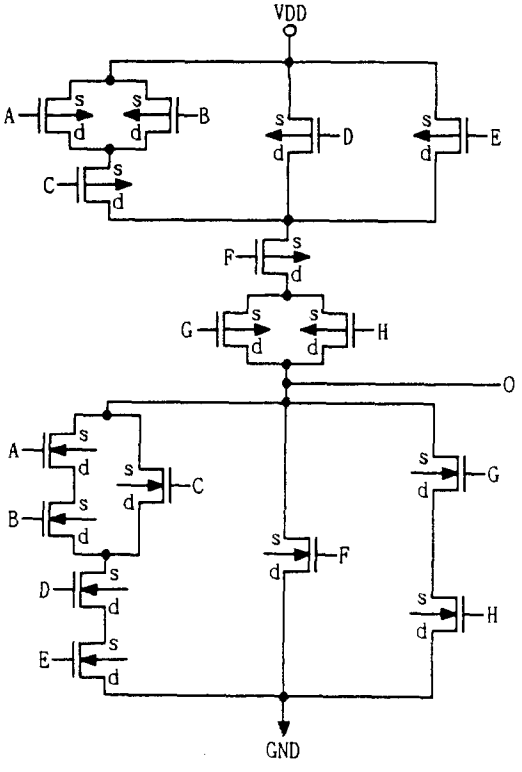


그림 1. 전처리된 수직 네트워크의 예
Fig. 1. A typical preprocessed vertical network.

AND-순서 트리 구조는 수직 네트워크의 토포로지 정보를 저장하고 있기 때문에 트리 구조로부터 네트워크 토포로지의 동질성(isomorphism)이 원상 복원되어야 한다. 이를 위해서는 스위치의 순회 순서와 노드에서의 토포로지 등의 정보가 트리 구조로부터 검색될 수 있어야 한다. 네트워크의 토포로지 정보를 트리 구조로부터 검색하는 과정은 그림 2(b)와 같다. 순회 번호의 지정 및 토포로지 정보의 검색은 내부노드 각각에 대하여 pre-order 순회에 따르면서 하향적(top-down)으로 진행된다. 이와 같은 토포로지 정보의 검색 과정을 통해 그림 3의 트리 구조에서 (1) 단노드의 스위치들에 복원된 순회 번호가 네트워크에 대한 스위치의 NBFS 횡단 순서와 일치한다는 것과 (2) 네트워크 노드의 연결 토포로지 정보가 검색됨을 알 수 있다. 이것은 제안된 AND-순서 토포로지 트리 데이타 구조가 스위치 레벨의 드라이버와 로드 네트워크 토포로지 정보를 저장 및 복원할 수 있다는 것을 확인한다.

```
class TR {
public:
    TR    &operator+=(TR &GivenTR);        // this += GivenTR
    TR    &operator*=(TR &GivenTR);        // this *= GivenTR
    // Build up a new tree whose left subtree is 'this' and right subtree is 'GivenTR'. Its root
    // would be 'OR' for '+=' and 'AND' for '*='. If its root is the same as the root of the
    // right subtree, affiliate this into the leftmost child of right one. Connect the drain
    // of 'this' to the drain of 'GivenTR' to delete 'GivenTR'.
};

// Power node must be given as 'GivenNode'.
void NBFS(Node *GivenNode) {
        TR      &GivenTR = *(GivenNode->AnySourceConnectedTR());
        TR      *TempTR;

        if(GivenTR.HasSourceCommonTR()) {   // @ Split node
                GivenTR.DisconnectFromSourceNode();
                NBFS(GivenNode);
                GivenTR.ConnectSourceTo(GivenNode);
        }
        while((TempTR = GivenTR.AnyParallelTR()) != NULL)
                GivenTR += *TempTR;         // OR 가입

        if(GivenTR.HasDrainCommonTR())      // @ Merge node
                return;

        if(GivenTR.HasSerialSubnet()) {
                NBFS(*GivenTR.DrainNode);
                GivenTR *= *(GivenTR.SerialTR());   // AND 가입
        }

        if((GivenTR.DrainNode != BottomNode || GivenTR.HasParallelSubnet())
                NBFS(GivenNode);  // When output node = merge node
        return;
}
```

(a)

```
Traverse(TreeNode *Node, int Distance) {
        Node->Distance = Distance;
        if(Node->Operation == OR) {
                for(ChildNode in (Node->AllChildren)) {
                        Traverse(ChildNode, Distance);
                }
        }
        if(Node->Operation == AND) {
                for(ChildNode in (Node->AllChildren)) {
                        Traverse(ChildNode, Distance);
                        Distance++;
                }
        }
}
```

(b)

그림 2. AND-순서 토포로지 트리 유도 및 복원과정
(a) 토포로지 트리 데이터 구조 유도 과정
(b) 스위치의 횡단 순서 및 토포로지 정보 검색 과정

Fig. 2. Derivation and retrieval procedures of AND-ordered topology tree structures.
(a) Deviration procedure for the topology tree representation (b) Retrieval procedure for the traversed order and the topology information of switches.

3. 대수적 토포로지

수직 네트워크의 토포로지를 기술하는 트리 데이타 구조는 순차식(sequential equation)의 형태로 기술될 수 있다. 순차식은 트리 데이타 구조를 in-order 순회

하여 유도된다. 그림 3의 로드와 드라이버 트리 구조로부터 (((A .OR. B) .AND. C) .OR. D .OR. E) .AND. F .AND. (G .OR. H)와 (E .AND. D .AND. ((B .AND. A) .OR. C) .OR. F .OR. (H .AND. G)가 각각 유도된다. AND-순서 토포로지 트리 구조에 대한 순차식 표현은 스위치 레벨의 수직 네트워크를 효율적으로 저장하고 처리할 수 있도록 한다.

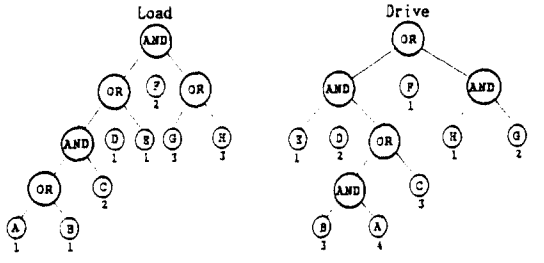
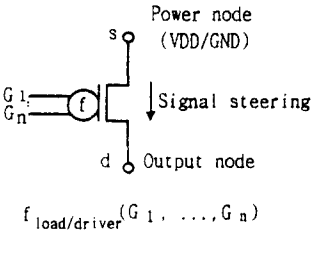


그림 3. 그림 1의 수직 네트워크에 대한 트리 구조 표현
Fig. 3. The tree representations for the vertical network of Fig. 1

드라이버-로드 형태의 수직 네트워크를 순차식으로 표현하면 상위 레벨(higher level)의 모델링이 가능하다. 본 연구에서는 수직 네트워크를 표현하기 위해 그림 4(a)의 "네트워크 스위치" 모델을 제안한다. 네트워크 스위치는 구조적으로 "소스"와 "드레인"이라는 데이타(data) 터미날과 "게이트"라 불리는 다중(multiple) 전도-제어(conduction-control) 터미날을 갖는 다중 터미날 모델이다. 소스와 드레인 데이타 터미날은 구조면에서 상이한 점이 없어 기존 스위치 모델과 같이 양방향 신호 전달 특성을 갖는다. 두 데이타 터미날간의 전도 상태는 다중 전도 제어 터미날 $G_1 \sim G_n$의 steering 함수 $f(G_1, \cdots, G_n)$에 의해서 결정된다. 드라이버나 로드 네트워크를 구성하는 스위치들의 게이트 터미날을 네트워크 스위치의 전도-제어 터미날에 대응시키면, steering 함수는 드라이버나 로드 네트워크의 토포로지를 나타내는 대수 순차식으로 표현된다. 스위치 레벨 네트워크에서 부울 함수 기능을 수행하는 모든 드라이버-로드 형태의 수직 네트워크를 네트워크 스위치로 표현하면 그림 4(b)와 같은 혼합 레벨 네트워크가 얻어진다. 혼합 레벨 네트워크는 수평 네트워크를 기술하는 기존 스위치 레벨 모델과 수직 네트워크의 드라이버 및 로드 네트워크를 표현하는 네트워크 스위치 모델의 연결 토포로지로 구성된다. 네트워크 스위치가 수평 네트워크를 구성하는 스위치 레벨 모델과 구조적 호환 관계를 유지하기 때문에, 혼합 레벨 네트워크의 정상 상태(steady-state)를 계산하기 위해 기존 스위치 레벨 시뮬레이터[1,18,19]의 네트워크 횡단 알고리즘과 RC 트리 검증 모델(evaluation model)[1,16]을 활용하는 것이 가능하다.



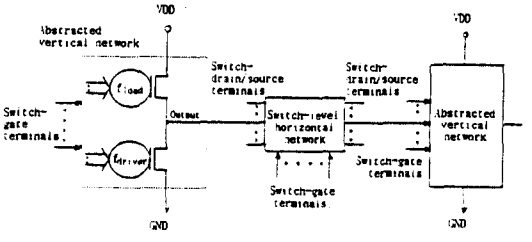


그림 4. 혼합레벨 네트워크의 모델링
(a) 네트워크 스위치 (b) 혼합레벨 네트워크의 토포로지
Fig. 4. Mixed level network modeling.
(a) Network switch (b) Topology of a mixed-level network

네트워크 스위치는 디지탈 MOS 회로의 드라이버나 로드 네트워크 동작(behavior)을 기술할 수 있어야 한다. 드라이버나 로드 네트워크의 동작은 파워 노드와 출력 노드간의 정적(static) 저항성 신호강도와 동적(dynamic) 용량성 신호강도에 의해 표현될 수 있다. 드라이버나 로드 네트워크의 신호강도를 네트워크 스위치로부터 연산하기 위해 기존의 부울 대수를 확장시킨 대수 구조[11]를 이용하였다. 본 연구에서는 이 대수 구조가 저항성 신호강도의 연산만이 가능한 한계를 해결하기 위해, 드라이버와 로드 네트워크의 내부 용량성 신호강도를 다룰 수 있도록 대수 구조의 연산자를 확장 정의하였다. 이 과정에서 기존 스위치 레벨의 네트워크 모델링이 노드의 용량 소자를 접지시키고 있는 것을 고려하여 노드의 직렬 연결은 종속(cascaded) 구

조로 병렬 연결은 분기(forked) 구조로 각각 표현한다.

대수 구조는 AND(**θ**)와 OR(**Ω**) steering 연산자를 신호강도 세트 **S** = [0,∞)∪{∞, x}에 대하여 정의하고 있다. 신호강도 세트 **S**는 저항 및 용량 신호강도를 표현하는 스위치의 콘덕턴스와 노드의 캐패시턴스 값으로 구성되는데, 신호강도는 실수 r ∈ (0,∞) < 미지(unknown)나 무효(invalid)의 x < 무한대 ∞의 순서를 갖는다. 표 1은 steering 연산자의 저항성 신호강도 산출을 위한 연산 법칙을 정의하고 있다. AND 연산자는 직렬 연결된 스위치들의 저항 임피던스를 계산하여, OR 연산자는 병렬 연결된 토포로지를 처리한다. 용량성 신호강도를 위한 steering 연산자의 연산 법칙은 표 2에 보인다. AND 및 OR연산자는 노드들의 직렬 종속(serial-cascaded) 및 병렬 분기(parallel forked) 연결 토포로지에 대한 용량성 신호강도를 각각 계산한다. 이 대수 구조 < **S**, **θ**, **Ω**, 0, ∞ > 는 다음 정리들을 만족한다.

표 1. 저항성 steering 연산자의 연산 법칙
Table 1. Computation rules of the resistive steering operators.

(a) a .AND. b 저항 steering 기능
(a) Resistive steering of a .AND. b

a \ b	0	u	v	x	∞
0	0	u	v	x	∞
u	u	2u	u+v	x	∞
v	v	u+v	2v	x	∞
x	x	x	x	x	∞
∞	∞	∞	∞	∞	∞

(b) a. OR. b 저항 steering 기능
(b) Resistive steering of a .OR. b

a \ b	0	u	v	x	∞
0	0	0	0	0	0
u	0	u/2	$\frac{uv}{u+v}$	u	u
v	0	$\frac{uv}{u+v}$	v/2	v	v
x	0	u	v	x	x
∞	0	u	v	x	∞

(1) < **S**, **θ**, 0 >는 monoid이다.

여기서 monoid는 1) 연산 결과가 세트 **S**에 속하고, 2) 연산자가 결합(associative)법칙을 만족하며, 3) 요소(element)가 연산자에 대해 identifier가 됨을 가리킨다.

(2) < **S**, **Ω**, ∞>는 monoid이다.

(3) 연산자 **θ**와 **Ω**는 상호 교환(commutative) 법칙을 만족한다.

(4) 요소 ∞와 0은 **θ**와 **Ω** 연산자 각각에 대한 annihilator이다.

표 2. 용량성 steering 연산자의 연산 법칙
Table 2. Computation rules of the capacitive steering operators.

(a) a .AND. b 용량 steering 기능
(a) Capacitive steering of a .AND. b

a \ b	0	u	v	x	∞
0	0	u	v	x	∞
u	u	2u	u+v	x	∞
v	v	u+v	2v	x	∞
x	x	x	x	x	∞
∞	∞	∞	∞	∞	∞

(b) a .OR. b 용량 steering 기능
(b) Capacitive steering of a .OR. b

a \ b	0	u	v	x	∞
0	0	0	0	0	0
u	0	u	min (u,v)	u	u
v	0	min (u,v)	v	v	v
x	0	u	v	x	x
∞	0	u	v	x	∞

Ⅳ. 드라이버-로드 네트워크 검증 알고리즘

수직 네트워크에서 출력 노드의 정상 상태(steady state)를 구하기 위해서 네트워크 스위치들의 용량성 및 저항성 특성을 분석한다. 또한 수직 네트워크의 전파-지연시간(propagation-delay time)을 Thevenin 등가 회로의 RC 모델을 이용하여 계산한다.

1. 저항 특성 분석

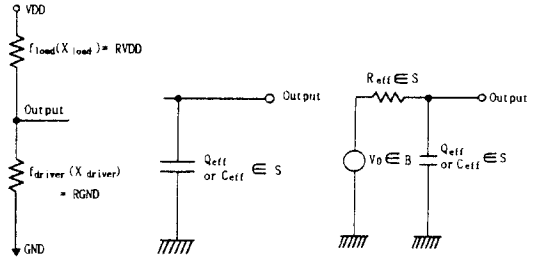
네트워크 스위치의 저항성 동작을 분석하는 과정은 다음의 (1) **B**(불리안 세트: 0/1/X)에서 **S**(신호강도 세

트)로의 사상(mapping)과 (2) steering 함수의 순차적 연산 처리로 이루어진다:

(1) 입력 부호화(**B** -> **S**) : 각 스위치에서 게이트 터미날의 신호 레벨(0/1/X)은 스위치의 전도 상태(On, Off, Unknown/Invalid)를 결정한다. 스위치가 ON되면 신호강도는 스위치의 ON-채널 저항값인 실수 r∈(0, ∞)가 되고, OFF 경우에는 무한대(∞)의 신호강도 값이 지정된다. Unknown 이나 Invalid한 스위치의 전도 상태는 ON-채널 저항 r에서 무한대 ∞까지의 범위 값(r, ∞)을 갖는 x 신호강도 값을 갖는다.

(2) Steering 함수의 연산 처리 : 표 1의 계산 법칙에 따른 steering함수에 대한 순차 연산 처리를 stack 구조를 이용하여 실행한다. Steering 함수의 왼괄호 "("는 따르는 계산 결과를 stack에 푸쉬(push)하고 오른괄호 ")"는 stack으로부터 값을 팝(pop)하는 처리를 행한다. 함수로부터 얻어지는 계산 결과는 네트워크 스위치의 데이타 터미날간 저항성 신호강도를 나타낸다.

로드와 드라이버 네트워크 스위치에 대한 저항성 신호강도 연산은 수직 네트워크에서 출력 노드의 정상 상태를 한쌍의 신호강도 RVDD&RGND로 표현한다(그림 5(a)).



(a) (b) (c)

그림 5. 수직 네트워크의 등가 회로
(a) 저항성 등가회로 (b) 용량성 등가회로 (c) 레브닌 등가회로

Fig. 5. Equivalent circuits for a vertical network.
(a) A resistive equivalent circuit (b) A capacitive equivalent circuit (c) A Thevenin's equivalent circuit

여기서 RVDD와 RGND는 로드 및 드라이버 네트워크의 저항성 신호강도 값을 나타낸다. 출력 노드의 신호강도 RVDD&RGND로부터 수직 네트워크의 출력 임피던스 R_{eff}와 출력 신호 레벨 V_0(0/1/X)의 결정은 표 3에 따른다. 이 표는 **S**-도메인에서 **B**-도메인으로의 역사상(inverse-mapping)을 보이고 있다.

표 3. 출력 노드의 신호강도 RVDD&RGND로 부터 신호레벨 V_0와 출력 임피던스 R_{eff}의 산출

Table 3. Derivation of signal level V_0 and Output impedance R_{eff} from the signal strength RVDD&RGND at the output node.

RVDD & RGND (S)	Signal level V_0 (**B**)	Output impedance R_{eff} (S)
∞ & r	0	r
r & ∞	1	r
r_1 & r_2	0 if $\frac{r_2}{r_1+r_2} < V_{low}$	r_2
	1 if $\frac{r_2}{r_1+r_2} > V_{high}$	r_1
	X ; O.W.	x
x & x	X	x
x & ∞	X	x
∞ & x	X	x
∞ & ∞	Z	∞

Boolean domain B = { 0, 1, X } real
r, r_1, r_2 ∈ (0, ∞)
Strength domain S = (0, ∞) ∪ { x, ∞ }
Z: High-impedance state

2. 용량 특성 분석

수직 네트워크의 용량 특성은 출력 노드에서의 부하(load)나 전하(charge) 신호강도로 구해진다. 부하 신호강도는 출력 노드와 연결된 전도 패스(conduction path)를 구성하는 용량성 노드들의 파워 노드에 대한 부하 효과를 가리키며, 전하 신호강도는 VDD와 GND 파워 노드로부터 격리되어 있는 출력 노드에 전기적으로 연결되어 있는 수직 네트워크의 내부 노드들에 저장되어 있는 전하량을 의미한다. 이러한 용량성 신호강도는 드라이버와 로드 네트워크의 전도 패스 토포로지를 동적으로 분석하여 산출한다. 토포로지의 동적 분석은 드라이버와 로드 네트워크를 표현하는 네트워크 스위치의 steering 함수에 대하여 다음과 같이 행하여진다.

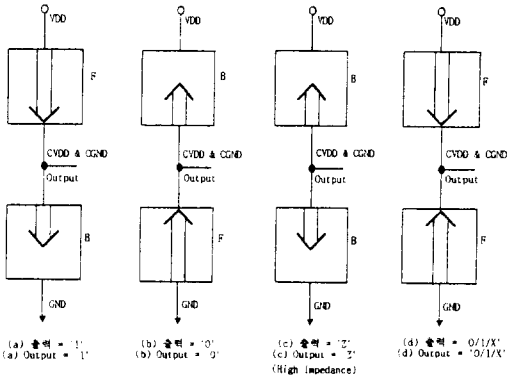
(1) 입력 부호화(**B** -> **S**):

네트워크를 구성하는 각 스위치에서 게이트 터미날의 신호 레벨(**B**)은 데이타 터미날(소스나 드레인) 노

드의 용량성 신호강도(S)로 사상된다. ON 스위치에서 소스나 드레인 터미날의 노드 용량 $r \in (0,\infty)$이 용량 신호강도가 되고, OFF 스위치의 경우에는 무한대 용량 신호강도가 주어진다. Unknown/Invalid 스위치는 최악의 경우(worst-case)로 시뮬레이션 하기 위해 ON 스위치로 처리한다. 각 스위치에서 데이타 터미날 간의 용량 효과는 순방향(forward)및 역방향(backward)의 양방향으로 생각할 수 있다. 순방향 효과는 드레인 터미날의 소스에 대한 용량 효과를 가리키고 역방향 효과는 반대 방향의 용량 부하 효과를 의미한다.

(2) Steering 함수의 연산 처리:

표 2의 계산 규칙에 따라 네트워크 스위치의 steering 함수를 순차 연산한다. 스위치의 양방향 용량 효과를 고려하면 steering 함수에 대한 연산은 순방향 및 역방향으로 처리될 수 있다. Steering함수를 왼쪽에서 오른쪽으로 계산하는 순방향 연산은 네트워크 스위치의 전도패스에 위치하는 내부 노드를 파워 노드로부터 출력 노드까지 순서대로 다루게 되며, Steering 함수를 반대로 처리하는 역방향 연산은 출력 노드에서 파워 노드 방향으로 출력 노드와 전기적으로 연결된 내부 노드를 다루게 된다. 연산 방향에 따라 각 스위치에 대하여 용량 효과를 갖는 데이타 터미날 노드를 결정하는데, 순방향과 역방향 연산에서 ON 스위치들에 대한 드레인과 소스 터미날의 용량 값이 각각 입력 부호화 과정에서 이용된다.



(a) (b) (c) (d)

그림 6. 순방향과 역방향 연산의 4가지 조합 경우
(a)출력='1' (b)출력='0' (c) 출력='Z' (d) 출력='0/1/X'

Fig. 6. Four combinations of forward(F) and backward(B) evaluation.
(a)Ouptut='1' (b)Ouptut='0' (c)Ouptut='Z' (d)Ouptut='0/1/X'

드라이버-로드 형태 수직 네트워크의 출력 노드에서 용량성 신호강도를 시뮬레이션하기 위해 순방향 및 역방향 연산은 그림 6과 같이 조합된다. 로드와 드이버 네트워크 스위치의 steering 함수로부터 얻어지는 한 쌍의 신호강도 CVDD&CGND는 출력 노드의 정상 상태(steady state) 용량 특성(capacitive effect)을 나타낸다.

a) 출력 노드가 VDD파워에 의해 구동될 경우(그림 6(a))

C_{eff} = CVDD + CGND;

CVDD = $\sum C_i$ | 순방향 연산
로드 네트워크 스위치의 0이나 X 상태를 갖는 내부 노드

CGND = $\sum C_j$ | 역방향 연산
드라이버-네트워크 스위치의 0이나 X 상태를 갖는 내부 노드

b) 출력 노드가 GND파워에 의해 구동될 경우(그림 6(b))

C_{eff} = CVDD + CGND;

CVDD = $\sum C_i$ | 역방향 연산
로드-네트워크 스위치의 1이나 X 상태를 갖는 내부 노드

CGND = $\sum C_j$ | 순방향 연산
드라이버 네트워크 스위치의 1이나 X 상태를 갖는 내부 노드

c) 출력 노드가 파워 노드로부터 격리될 경우(그림 6(c))

$Q_{eff} = (Q_{min}, Q_{max})$
$= (CVDD_{min}+CGND_{min}, CVDD_{max}+CGND_{max})$;

$CVDD_{min}$ = $\sum C_i$ | 역방향 연산
로드-네트워크 스위치의 1의 상태를 갖는 내부 노드

$CVDD_{max}$ = $\sum C_i$ | 역방향 연산
로드 네트워크 스위치의 1이나 X의 상태를 갖는 내부 노드

$CGND_{min}$ = $\sum C_j$ | 역방향 연산
드라이버 네트워크 스위치의 1의 상태를 갖는 내부 노드

$CGND_{max}$ = $\sum C_j$ | 역방향 연산
드라이버 네트워크 스위치의 1이나 X의 상태를 갖는 내부 노드

d) 출력 노드가 VDD와 GND 파워 노드에 의해 구동될 경우(그림 6(d))

C_{eff} = CVDD output = 1 일때
혹은 CGND output = 0 일때;

CVDD = $\sum C_i$ | 순방향 연산
로드 네트워크 스위치의 0이나 X의 상태를 갖는 내부 노드

CGND = $\sum C_j$ | 순방향 연산
드라이버 네트워크 스위치의 1이나 X의 상태를 갖는 내부 노드

출력 노드에서의 용량성 신호강도 CVDD와 CGND는 부하나 전하 효과로서 그림 5(b)의 등가 회로로 나타낼 수 있다.

3. 타이밍 분석

수직 네트워크에 대한 저항성 및 용량성 신호강도를 통해 그림 5(c)과 같은 Thevenin 등가 회로를 얻을 수 있다. 출력 임피던스 R_{eff}와 C_{eff}에는 수직 네트워크의 토포로지와 입력 레벨 및 내부 노드 상태가 반영되기 때문에, 출력 노드에 대한 모든 영향을 고려된 등가 회로가 구성된다. 기존의 RC 타이밍 지연 모델을 이용하면, 수직 네트워크의 입력과 출력간의 전파 지연시간(propagation-delay time)을 다음과 같이 계산할 수 있다.

$$T = R_{eff} \times C_{eff} \; ; \; V_0 = 0 \text{ 혹은 } 1$$
$$= \Delta_{min} \; ; \; V_0 = X \text{ 혹은 } Z$$

여기서 Δ_{min} 은 최소 해상(resolution) 시간 단위를 가리킨다.

실제로 이와 같은 수직 네트워크의 입력과 출력간의 전파-지연시간은 다중 입력 단자에 대한 차이[23]와 MOS 회로 고유의 축차(sequential) 동작에 따른 차이를 고려하고 있다. 본 논문에서 제안된 지연시간 모델은 저항성 및 용량성 신호강도 연산 과정이 출력 노드에 영향을 미치는 모든 효과(즉, 토포로지와 스위치 입력상태 및 내부 노드 상태)를 고려했기 때문에 선형 스위치 레벨의 타이밍 계산이 가능했다.

V. 실험 결과 및 고찰

본 연구에서 수직 네트워크 기술을 위해 제안된 네트워크 스위치 모델링은 스위치 레벨 모델과 구조 및 기능적 호환성(compatibility)을 유지하기 때문에 네트워크 모델링 및 검증 알고리즘을 기존 스위치 레벨 로직 및 타이밍 시뮬레이터에 결합시키는 것이 가능했다. 제안된 방법을 기존의 스위치 레벨 시뮬레이터 RSIM[11]에 구현해서 실험적인 혼합 레벨 시뮬레이터 MIXIM을 개발하였는데, 본 장에서는 혼합 레벨 시뮬레이션의 성능을 평가하기 위한 실험 결과를 보인다. 실험은 시뮬레이션 성능의 1) 계산 시간과 2) 타이밍 정확도 두 가지 측면에서 행하여 졌다.

1. 시뮬레이션 오버 헤드

혼합 레벨 시뮬레이션의 계산 오버헤드를 알아보기 위해서 MIXIM의 시뮬레이션 시간을 RSIM과 비교한다. 모든 계산 시간은 Sun Sparc II 호환 기종 워크스테이션에서 100개의 입력 사건에 대하여 측정된 결과이다. 검증될 디지탈 MOS 회로는 2차 Booth 승산 알고리즘[24]에 따른 10×15 비트 파이프 라인 승산기(pipelined multiplier)를 사용하였는데, 회로의 크기에 따른 오버 헤드의 변화를 알아보기 위해 크기 순서로 분할된 (1) 시스템 클럭 발생기(TC1), (2) 시스템 순차 클럭 발생기(TC2), (3) 시스템 제어기(TC3), (4) 파이프라인 승산기(TC4)의 회로들을 시뮬레이션하였다. 표 4는 MIXIM과 RSIM의 시뮬레이션 결과를 요약한다. 실험 결과로부터 다음과 같은 고찰이 가능했다.

1) 회로의 크기가 커짐에 따라 RSIM의 시뮬레이션 오버 헤드는 급격히 증가하나 MIXIM의 증가는 완만하다. MIXIM이 RSIM에 비하여 시뮬레이션 성능이 우수하나 전처리 과정에서 보다 많은 오버 헤드가 요구된다.
2) 시뮬레이션의 오버 헤드는 주로 스케쥴링과 네트워크 처리 및 대수 처리 부분에서 요구된다. 스케쥴링 부분은 시뮬레이션의 사건(event) 처리를 담당하고, 네트워크 처리 부분은 스위치 레벨의 RC 네트워크 분석을 행하며, 대수 처리 부분은 본 논문에서 제안한 네트워크 스위치의 시뮬레이션 기능을 수행한다. 세가지 계산 오버 헤드로 구분된 전체 시뮬레이션 시간은 회로 크기에 따라 그림 7과 같다. 스위치 레벨 시뮬레이터 RSIM에서는 네트워크 처리 부분에 의한 오버헤드 때문에 회로가 커짐에 따라 성능이 급격히 떨어지는 현상이 나타나지만, 혼합 레벨 시뮬레이터 MIXIM에서는 대수 처리 부분의 완만한 계산 오버 헤드가 네트워크 부분의 급격한 오버 헤드를 경감시켜서 전체 시뮬레이션 오버헤드를 개선함을 보이고 있다.
3) 파이프라인 승산기(TC4)에 대하여 MIXIM은 RSIM에 비해서 7배(217.3scc/ 34.5sec) 가량의 시뮬레이션 속도를 보인다. 그러나 MIXIM은 전처리 과정에서 RSIM보다 3배(35.1sec/12.7sec) 가량의 오버헤드를 요구하고 있다. 시뮬레이션 시간에 대한 전처리 과정의 오버 헤드는 수행될 시뮬레이션이 여러 입력 값에 대해 진행되기 때문에 상대적으로 오버헤드가 작아져서 허용의 여지가 커진다고 볼 수 있다.

표 4. RSIM과 MIXIM의 계산 성능
Table 4. Computational performances of RSIM and MIXIM.

Circuit	Simulation results			RSIM			MIXIM		
TC 1 Figure of merit = 0.57	Primitives	Node		130	48		75	25	
		TR			82			50	
	Preprocessing(sec)			0.05			0.05		
	Simulation (sec)	Scheduling		0.91	0.09		0.69	0.01	
		Network Manipulation	Network trace		0.81	0.58		0.31	0.29
			Thevenin			0.23			0.02
		Algebraic Manipulation			0.0			0.36	
TC 2 Figure of merit = 0.504	Primitives	Node		1131	438		571	188	
		TR			693			383	
	Preprocessing(sec)			0.4			0.55		
	Simulation (sec)	Scheduling		5.59	0.42		3.14	0.06	
		Network Manipulation	Network trace		5.17	3.01		1.35	1.32
			Thevenin			2.16			0.03
		Algebraic Manipulation			0.0			1.73	
TC 3 Figure of merit = 0.295	Primitives	Node		2938	1247		869	287	
		TR			1691			581	
	Preprocessing(sec)			1.01			1.40		
	Simulation (sec)	Scheduling		21.57	1.32		5.61	0.06	
		Network Manipulation	Network trace		20.24	2.49		1.49	1.48
			Thevenin			17.75			0.01
		Algebraic Manipulation			0.0			4.05	
TC 4 Figure of merit = 0.588	Primitives	Node		17491	6627		9535	2980	
		TR			10864			6555	
	Preprocessing(sec)			6.99			35.09		
	Simulation (sec)	Scheduling		217.30	6.48		34.52	0.74	
		Network Manipulation	Network trace		210.82	42.46		12.68	12.66
			Thevenin			168.36			0.02
		Algebraic Manipulation			0.0			21.09	

4) 혼합 레벨 모델링은 스위치 레벨보다 네트워크 원소(primitive)의 갯수를 현저하게 감소시키는 장점이 있다. 본 연구에서는 Figure of Merit(F.M.)을 시뮬레이션 할 네트워크에서 (MIXIM이 요구하는 모델링 원소의 갯수)/(RSIM이 필요로 하는 모델링 원소의 갯수)의 비율로 정의한다. 어떤 회로에 대해 혼합 레벨 표현이 얼마나 효과적인가를 나타내는 F.M.에 대한 시뮬레이션 성능을 살펴보면 주요한 사실이 발견된다. 즉, 스위치 레벨에서 TC3의 회로가 TC2에 비해서 3배 정도 많은 모델링 원소를 요구함에도 불구하고 혼합 레벨 시뮬레이션에서 스케쥴링과 네트워크 처리가 요구하는 오버 헤드는 거의 같다. 이러한 처리 부분들은 네트워크 원소의 갯수에 매우 민감한 데, TC3의 F.M.이 0.3이고 TC2의 F.M.이 0.5인 점을 생각하면 F.M.이 작은 회로일수록 MIXIM이 계산 오버 헤드를 줄이는데 효과적이라는 점을 추정할 수 있다.

2. 시뮬레이션 정확도

본 논문에서는 스위치 레벨의 수직 네트워크를 효과적으로 시뮬레이션하기 위한 네트워크 스위치 모델링을 제안하고 있는데, 새로운 모델로부터 선형 스위치 레벨에 대응되는 정확도를 산출할 수 있는지에 대한 확인이 요구된다. 따라서 수직 네트워크의 정상 상태

및 전파-지연(propagation delay) 시간을 네트워크 스위치로 모델링한 드라이버-로드 네트워크에 대하여 시뮬레이션하는 실험을 실행하였다.

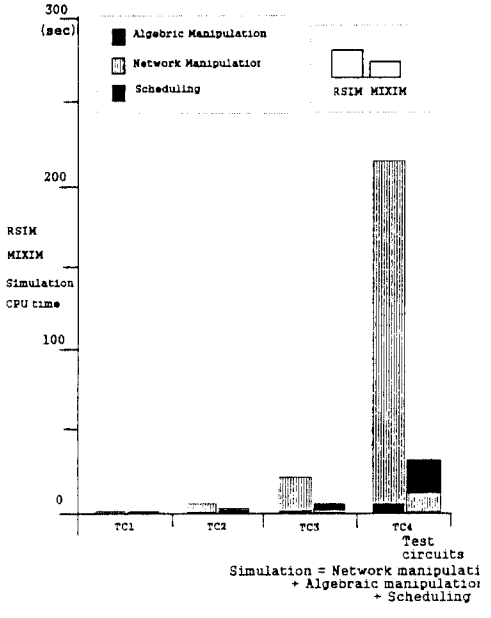


그림 7. RSIM과 MIXIM의 시뮬레이션 CPU 시간
Fig. 7. Simulation CPU times of RSIM and MIXIM.

실험결과의 정확성을 확인하기 위하여 MIXIM의 전파 지연 시간 예측 능력을 RSIM[11]과 SPICE[12]와 비교했다. 2/4/8/16입력의 CMOS 로직 게이트의 지연 시간을 랜덤 입력 벡터에 대하여 시뮬레이션 하여 그림 8과 같은 실험 결과를 얻었다. SPICE는 기준 데이타(reference data)를 보이고, RSIM은 스위치 레벨에서의 전형적인 RC 지연시간을 산출한다. 실험 결과에서 MIXIM은 SPICE에 근접한 지연시간을 예측하고 있으나, RSIM은 네트워크에 분산된 용량 특성과 병렬 패스의 저항 특성을 적절하게 처리하지 못해서 과도(over)하거나 불충분한(under) 지연 시간을 산출하고 있다. SPICE를 기준으로 하여 RSIM과 MIXIM의 시뮬레이션 능력을 상관계수(covariance) 및 상대오차(relative error) 항목을 통해서 비교한 것이 표 5에 정리되어 있다. MIXIM은 SPICE에 대하여 17%의 오차 범위와 0.986 상관 관계의 타이밍 정확성을 산출함이 보인다. 이같은 실험 결과를 통해서 MIXIM이 RSIM보다 수직 네트워크에 대한 전파 지연 시간 예측에서 우수하다는 점을 정량적으로 확인할 수 있었다.

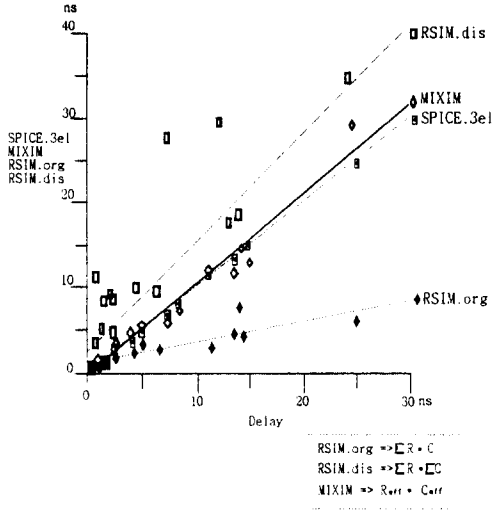


그림 8. 2/4/8/16 입력 CMOS 로직 게이트에 대한 SPICE, RSIM 및 MIXIM의 전파지연 시간 시뮬레이션 결과
Fig. 8. Simulation results from SPICE, RSIM and MIXIM for 2/4/8/16 inputs CMOS combinational logic circuits.

표 5. SPICE에 대한 RSIM과 MIXIM의 상관계수 및 상대오차
Table 5. Covariance and Relative error of RSIM and MIXIM with respect to SPICE.

	2 input gate	4 input gate	8 input gate	16 input gate	Avg.
Covariance (RSIM, SPICE)	0.911	0.243	0.766	0.755	0.842
Covariance (MIXIM, SPICE)	0.983	0.962	0.994	0.981	**0.986**
$\frac{\lvert RSIM - SPICE \rvert}{SPICE}$	17.95%	41.70%	31.43%	73.61%	45.96%
$\frac{\lvert MIXIM - SPICE \rvert}{SPICE}$	10.26%	13.61%	**16.81%**	15.21%	14.17%

VI. 결 론

본 논문에서는 스위치 레벨 로직 및 타이밍 시뮬레이션의 성능 개선을 위한 데이타 구조와 상위 레벨 모델링 및 검증 알고리즘을 제안한다. 디지탈 MOS 회로에서 (1) 드라이버-로드 형태의 로직 게이트를 네트워크 토포로지와 병행한 트리 데이타 구조로 표현하고, (2) 데이타 구조를 대수적으로 연산 처리하여, (3) 전체적으로 선형 스위치 레벨의 정확성을 유지하면서 혼

합 레벨 시뮬레이션에 따른 오버헤드의 경감을 얻었다. 제안된 방법을 RSIM에 결합시킨 스위치 레벨 로직 및 타이밍 시뮬레이터 MIXIM이 SPICE에 대하여 17%의 지연시간 오차를 갖고 RSIM에 대하여 7배 이상의 속도로 시뮬레이션을 행한다는 것을 실험 결과에서 확인하였다.

참 고 문 헌

[1] C. J. Terman, "RSIM-A Logic-Level Timing Simulator," *Proceedings of 1983 ICCD Conference*, pp. 437-440, 1983.

[2] W. A. Christoppher, *Spice3el manual*, UC-Berkeley, 1985.

[3] V. B. Rao, D. V. Overhauser, T. N. Trick and I. N. Hajj, *Switch-level Timing Simulation of MOS VLSI Circuits*, Kluwer Academic Publishers, 1989.

[4] I. N. Hajj and D. Saab, "Symbolic logic Simulation of MOS circuits," *Proc. Int. Symp. Circuits Syst.*, pp.246-249, 1983.

[5] R. E. Bryant, "Algorithmic Aspect of Symbolic Switch Network Analysis," *IEEE Trans. on CAD*, CAD-6, No. 4, pp.618-633, July 1987.

[6] R. E. Bryant, "Boolean Analysis of MOS Circuits," *IEEE Trans. on CAD*, CAD-6, No. 4, pp.634-649, July 1987.

[7] Z. Barzilai, et al., "SLS- A Fast Switch-level Simulator," *IEEE Trans. on CAD*, Vol. 7, No. 8, pp.838-849, 1988.

[8] D. T. Blaauw, et al., "SNEL: A Switch-level Simulator using Multiple levels of Functional Abstraction," *Proceedings of 1990 ICCAD Conference*, pp. 66-69, 1990.

[9] B. R. Chalwa, H. K. Gummel, and P. Kozak, "MOTIS- an MOS timing simulator," *IEEE Trans. on Circuits and Systems*, pp.901-909, Dec. 1975.

[10] C. Y. Lo, et al., "A Data Structure for MOS Circuits," *Proceedings of 20th Design Automation Conference*, pp. 619-624, 1983.

[11] 공진흥, "MOS 게이트 검증을 위한 대수적 접근 방법," *전자공학회 논문지*, 제 30권 A편 제 4호, 1993년 4월

[12] W. T. Weeks, et al., "Algorithm for ASTAP - a network analysis program," *IEEE Trans.*, CT-20, pp.628-643, 1973.

[13] R. A. Saleh, J. E. Kleckner, and A. R. Newton, "Iterated timing analysis in Splice1," *IEEE Proceedings of ICCAD*, pp.139-140, Nov. 1983.

[14] E. Lelarasmee and A. Sangiovanni-Vincentelli, "RELAX: a new circuit Simulator for large scale MOS integrated circuits," *Proceedings of the 19th DAC*, pp.682-690, June 1982.

[15] N. Jouppi, "Timing Analysis for NMOS VLSI," proceedings of 20'th DAC, 1983.

[16] J. K. Ousterhout, "A Switch-level Timing Verifier for Digital MOS VLSI," *IEEE Trans. on CAD*, Vol. CAD-4, No. 3, pp.336-349, July 1985.

[17] T. J. Schaefer, "A Transistor-level Logic-with-Timing Simulator for MOS Circuits," *Proceedings of the 22nd DAC*, pp.762-765, 1985.

[18] A. Salz and M. Horowitz, "IRSIM: An Incremental MOS Switch-level Simulator," *Proc. of the 26th DAC*, pp.173-178, 1989.

[19] R. Kao and M. Horowitz, "Piecewise Linear Models for RSIM," *Proceedings of 1993 ICCD Conference*, pp. 753-758, July 1993.

[20] R. E. Bryant, "A Switch-level Model and Simulator for MOS Digital System," *IEEE Transaction on Computer*, Vol. c-33, No. 2, pp. 160-177, Feb. 1984.

[21] R. E. Bryant, "COSMOS: A Compiled Simulator for MOS Circuits," *Proceedings of 24th Design Automation Conference*, pp. 9-16, 1987.

[22] J. P. Hayes, "Pseudo-Boolean Logic Circuits," *IEEE Trans. on Computers*, Vol. C-35, No. 7, pp.602-612, July, 1986.

[23] H. N. Nham and A. K. Bose, "A multiple delay simulator for LSI circuits," *Proceedings of 17th Design Automation Conference*, pp. 610-617, 1980.

[24] M. M. Mano, *Computer System Architectures*, Prentice-Hall, Reading, 1993.

저 자 소 개

孔 鎭 興(正會員)

1957年 9月 20日생. 1980年 서울대학교 전자공학과(공학사). 1982年 한국과학기술원 전기 및 전자공학과(공학석사). 1989年 미국 텍사스 주립대학교(Austin) 전기 및 컴퓨터 공학과(공학박사). 1982年~1986年 삼성전자 반도체 연구소 선임 연구원. 1989年~현재 광운대학교 컴퓨터공학과 부교수. 주관심 분야는 VLSI 설계 자동화 및 ASIC 설계 등임