특 집

IC설계를 위한 논리와 회로 시뮬레이션 (Logic and Circuit Simulation for intergated circuits)

李 起 煊

忠南大學校 電子工學科

I. 서 론

최근에, 반도체 제작 기술의 발전과 더불어, 다기능, 고집적, 저전력, 고신뢰도, 경량화, 기술보호 등을 위하여 전자 시스템의 IC화에 대한 중요도가 점점 더 증가하고 있다. 그러나, IC의 설계와 제작 과정은 복잡한 여러 단계를 거치게 되므로, 비용, 시간, 설계 및 제작 정보 등의 관점에서 일반적인 시스템 설계자가 IC설계에 쉽게 접근하기 어려운 현실이다. 이에 따라, 여러 가지 종류의 IC설계 방식이 개발되어 시스템 설계자와 반도체 제작자 사이의 교량 역활을 담당하고 있다. 특히, standard-cell 방식 또는 gate-array 방식으로 설계되는 semi-custom ASIC(application specific intergra-ted circuit)이나 FPGA는 기존의 board-level의 설계에 익숙한 시스템 설계자들로 하여금 IC 설계분야에 뛰어들 수 있는 좋은 IC 설계 방식 중의 하나이다.

IC의 설계 및 제작과정은 계층적인 top-down 설계방식에 의하여 다음과 같이 나눌 수 있다. (1) System-level Design, (2) Circuit-level Design, (3) Physical Layout, (4) IC Fabrication, (5) Test. 이상의 IC 설계 및 제작과정은 각각의 단계 별로 가능한 독립성을 유지하는 것이 바람직하다. 이를 위하여는, 각각의 설계 단계별로 설계 정보가 표준화되어야 하며, 설계 단계들 사이의 정보교환 창구가 있어서 설계 오류에 대한 검증 및 재설계 작업을 원활히 할 수 있어야 한다. IC설계시 고려하여야 할 설계 사양 들은 기능, 동작 속도, 칩 면적, 전압, 전력, 신호 레벨, 잡음 강도 등이 있다. 또한, IC 설계가 하위 레벨로 내려갈수록 반도체 제작 공정과 관련된 TR의 특성, interconnection, parsitic parameter, layout에서의 design rule, test 전략 등에 대한 고려가 있어야 한다.

일반적으로, IC 설계의 복잡도를 효과적으로 처리하기 위하여서는 컴퓨터를 이용한 설계(CAD)가 필요하다. CAD의 종류는 (1) synthesis, (2) simulation, (3) layout, (4) testing, (5) DRC, ERC, parameter 추출, circuit 추출 등이 있다. IC

설계를 위한 CAD는 독립적인 기능을 수행하는 point CAD tool과 통일된 설계 과정을 위하여 여러 가지 point tool들을 통합한 CAD system들이 있으며, 대부분 상용화되어 실제 설계환경에서 사용하고 있다. 그러나, 대부분의 CAD tool들은 고가이며, 다양한 기능을 제공하는 CAD tool의 완벽한 사용기술의 습득이 어렵고, 기술선진국(주로 미국)의 기술 종속적인 성향을 가지고 있다. 또한, 아무리 well-defined CAD tool이라 할지라도 완벽하지 않으며, 자체적으로는 많은 문제점을 가지고 있다. IC 설계자는 사용하는 CAD tool의 장점과 단점을 정확히 파악하여 설계환경을 정확히 CAD tool에 입력하여야 하며, CAD tool의 출력에 대한 충분한 검증이 있어야 한다. 즉, IC 설계는 설계자의 지식과 경험에 의하여 효과적인 CAD tool의 사용이 필요하다. 바꾸어 말하면, CAD tool은 IC 설계의 보조장비에 불과하며, 최종적인 설계 검증은 IC 설계자의 판단에 의하여 이루어진다. 결국, IC 설계는 설계 인력의 배양과 더불어 설계 환경에 적합한 CAD의 개발 및 사용이 중요하다.

이 글은 VLSI-CAD 중 시뮬레이션 분야에 대하여 설명한다. 시뮬레이션이란 주어진 전자 시스템 또는 IC에서 입력 정보가 주어졌을 때, 회로의 전기적 동작 특성을 컴퓨터에 의하여 해석하는 것을 의미한다. IC를 설계하는 과정에서 시뮬레이션은 수없이 반복 사용되며, 최적의 설계 결과를 얻기 위하여서는 많은 컴퓨터 해석시간이 요구된다. 실제로, 여러 가지 CAD 분야 중 시뮬레이션 분야는 가장 먼저 개발이 되었으며, 가장 많이 사용되고 있는 분야이다. 시뮬레이션에서의 해의 정확도와 컴퓨터 해석시간은 서로 상반되는 관계를 나타낸다. 해의 정확도를 높이기 위하여서는 많은 해석시간이 소요되며, 빠른 해석을 위하여서는 해의 정확도가 희생된다. 이 글은 여러 가지 시뮬레이션 분야 중에서 회로레벨 설계에서 사용되고 있는 논리시뮬레이션과 회로시뮬레이션에 대하여 언급하겠다. 제2장에서는 여러 종류의 시뮬레이터에 대한 간단한 소개가 있으며, 제3장에서는 논리시뮬레이션에 대한 설명이, 제4장에서는 회로시뮬레이션에 대한 설명이 주어진다.

II. 시뮬레이션의 종류

현재 널리 사용되고 있는 시뮬레이션의 종류들로는 IC 설계의 여러 단계에 맞추어, (1) 시스템 설계에서 사용할 수 있는 기능 시뮬레이션(behavioral/functional/register-transfer level simulation), (2) 논리 회로의 해석을 목적으로 하는 논리 시뮬레이션(gate-level/switch-level simulation), (3) 능동회로의 해석을 목적으로 하는 회로 시뮬레이션(timing/circuit simulation), (4) 반도체 제작 공정과 연관된 소자 시뮬레이션(device simulation)과 공정 시뮬레이션(process simulation) 등이 있다. 또한 testing을 위한 고장 시뮬레이션(fault simulation)과 지연시간 및 동작 속도의 검증을 위한 timing verification 등도 있다. 최근에는, 앞에서 언급한 여러 종류의 시뮬레이션을 묶어서 하나의 통합된 시뮬레이션 시스템으로 사용할 수 있는 혼합 모드 시뮬레이션(mixed-mode/multi-level simulation)과 시스템의 정상상태와 전력 소모를 해석하는 power simulation 및 초고속 MMIC를 위한 microwave simulation 등의 여러 가지 다양한 설계 목적에 부응하는 시뮬레이터들도 연구 개발되어 사용되고 있다.

기능 시뮬레이션은 여러 개의 부시스템들로서 구성된 전체 논리 시스템의 기능 및 timing 해석을 그 목적으로 한다. 전체 시스템은 주로 기능을 고려한 계층적 시스템 분할에 의하여 여러 개의 부시스템들로서 구성되며, 부시스템은 자체적으로 독립적인 기능을 수행하는 게이트 레벨 이상의 논리 블록들이 일반적이다. 전체 시스템 또는 부시스템들은 구조묘사(structural description) 또는 기능묘사(behaviour description)로서 설명을 한다. 기능 시뮬레이션에서는 시스템의 구조(architecture), 동작 원칙(behaviour algorithm, protocol) 등의 해석이 주목적이며, 주 clock cycle과 관련된 system timing에 대한 분석도 이루어진다. 기능 시뮬레이션의 입력은 주로 HDL(hardware description language)에 의하여 주어지며, 해석 방식은 논리 시뮬레이션에서 처럼 사건구동방식

(event-driven method)이 주로 사용된다. 기능 시뮬레이션을 위한 HDL로는 VHDL 또는 Verilog HDL 등이 있다. 현재 널리 사용되고 있는 VHDL 시뮬레이터는 기능 시뮬레이션의 한 예이다.

논리 시뮬레이션은 해석하고자 하는 회로의 기술 방식에 따라 gate-level과 switch-level로 나누어 진다. Gate-level은 NAND, NOR 등의 기본 논리 소자들로 구성된 회로를 의미하며, switch-level은 MOS 논리 회로의 논리 해석을 위하여 기본 소자가 MOSFET가 된다. 논리 시뮬레이션에서의 신호는 1, 0, X 등의 논리 신호가 되며, 회로 구성에 따른 boolean equation을 해석한다. 해석 방식으로는 사건구동방식이 대표적이다. Switch-level 시뮬레이션에서는 MOSFET를 gate-controlled switch로 간주하여 논리 시뮬레이션을 수행하며, signal strength 개념을 도입하여 high-impedance state 및 charge sharing 효과, 양방향성 소자해석 등을 처리한다. 논리 시뮬레이션은 bit 단위의 논리 해석이 이루어지며, 기본적인 시간해석 정보(first order timing information : harzard, glitch, spike, race condition)를 얻을 수 있다. 결국, 논리 시뮬레이션은 하위 개념의 기능 시뮬레이션으로서, 그 기능이 미리 정의가 된 논리 소자들로서 구성된 논리 회로를 해석한다. 각각 논리소자에 주어진 기능과 지연시간 정보를 기초로 하여 회로 연결구조에 따른 각각의 노드에서의 시간 파형을 해석하게 된다. Gate-level 논리 시뮬레이션에서는 사용자에 의하여 소자 지연시간이 주어지게 된다. Switch-level 논리 시뮬레이션에서는 지연시간을 자체 계산하고 있으나 그 정확도는 매우 떨어진다.

기본적으로, 기능 시뮬레이션과 논리 시뮬레이션은 논리 시스템의 해석을 그 목적으로 하고 있는 관점에서 동일하다. 가장 큰 차이점으로는 기능 시뮬레이션에서는 word 단위의 신호 정의가 가능하고 구성소자들의 기능을 사용자에 의하여 임의로 묘사할 수 있는데 반하여, 논리 시뮬레이션은 bit 단위의 신호 정의와 그 기능을 미리 정의한 내장된 논리 블록들만을 사용하게끔 되어 있다. 즉, 논리 시뮬레이션은 구조 묘사만이 허용되고 있는데 반하여, 기능 시뮬레이션은 구조 묘사와 기능 묘사가 동시에 가능하다. 최근에는, 논리 시뮬레이션과 기능 시뮬레이션의 구분이 거의 없이 혼합하여 사용하고 있다.

회로 시뮬레이션은 저항, 커패시터, 인덕터, 트란지스터 등의 기본 회로소자들로서 구성된 회로의 해석을 그 목적으로 하고 있다. 회로 시뮬레이션의 대표적인 예로 SPICE가 있다. 회로 시뮬레이션에서는 DC 해석, 과도시간 해석, 주파수 해석 등의 여러 가지 다양한 해석이 가능하다. 회로 시뮬레이션에서의 신호는 노드 전압 또는 소자 전류가 되며, KVL, KCL, 소자 특성식들을 고려한 비선형상미분방정식 형태의 회로 방정식들을 수치해석 기법에 의하여 컴퓨터 해석된다. 회로 시뮬레이션을 위하여서는 회로소자들의 연결 구조와 소자 모델 변수값들을 포함하는 회로정보와 해석기능의 선택, 출력 파형의 선정 등의 해석 정보가 입력 파일 형태로 제공되어야 한다. 이 중에서, 해석 정보가 주어지면 그에 맞는 내장된 해석 방식이 자동적으로 선택이 되어 주어진 회로 정보들을 기초로 한 해석식의 형성과 계산 등의 해석 과정을 수행하게 된다. 결과해의 정확도의 관점에서 보면, 정확한 회로 정보가 주어져야 하고, 해석 방식들의 알고리즘 오차가 최소화 되어야 한다. 회로 정보들 중에서는 특히 반도체 소자들의 반도체 제작과정과 관련된 정확한 모델 변수값이 필요하며(model parameter extraction), 경우에 따라서는 layout도면에서부터 회로 추출(circuit extraction)을 하여 interconnection, parasitic parameter, TR size 등의 (parasitic)회로 정보가 정확히 제공되어야 한다. 해석 방식의 오차를 최소화하기 위하여서는 SPICE의 예를 보면 option value(절대오차, 상대오차, 수치적분 방법, 반복 회수 등) 를 조절하여 원하는 해를 얻을 수 있으나, 근본적인 해석방식의 변경은 프로그램 자체를 변경하여야 되므로, 회로 시뮬레이션의 사용자 입장에서는 불가능하다. 회로 시뮬레이션은 입력정보가 정확히 주어지게 되면 정확한 결과해(1% 이내의 오차)를 얻을 수 있지만, 계산 시간이 많이 소모되는 단점이 있다.

보다 빠른 해석을 위하여 제공되는 시뮬레이션의 종류로는 시간 시뮬레이션(timing simulation)이 있다. 시간 시뮬레이션은 여러 가지 종류의 회로 분할을 기초로 한 이산해석방식(waveform/nonlinear/linear relaxation), ITA(iterative timing analysis), variable precision방식 또는 voltape step에 의한 시간격 조절법 등을 해석방식으로 채택함으로서 빠른 해석시간의 개선이 이루어진다. 그러나, 그 사용 범위가 MOS 논리 회로의 해석에 국한되어 있고, 결과해의 정확도도 떨어진다(10% 이상의 오차). 시간 시뮬레이션은 대규모 MOS 논리 회로의 prototype evaluation에 유용하게 사용되고 있다.

소자 시뮬레이션은 반도체 소자의 내부 물리적인 현상을 해석하는 프로그램으로서, 편미분방정식 형태의 continuity equation과 poisson equation의 해석이 되며, 결과는 반도체 소자의 구조에 따른 field 분포, 전압.전류·전하 분포 등을 얻을 수 있다. 공정 시뮬레이션은 반도체 제작 공정의 여러 단계를 단계별로 해석하는 프로그램으로서, 공정 설계환경에 따른 불순물 분포 및 확산, junction depth, 온도에 따른 물리 현상 등을 해석할 수 있다.

이제, 회로 및 시스템의 해석을 중심으로 이상의 시뮬레이션 종류들을 종합하여 보면, 해석시간의 관점에서는 기능 시뮬레이션이 가장 빠르고, 하위 레벨로 내려갈 수록 해석 시간이 느려진다. 예로서, 기능 시뮬레이션의 해석 속도를 1이라 하면, SPICE로 대표되는 회로 시뮬레이션은 10,000배 이상의 해석 시간이 요구된다. 해석할 수 있는 회로의 크기도 해석 시간과 비슷한 비율로서 생각할 수 있다. 즉, 기능 시뮬레이션은 수십만개 이상의 기본 소자들로 구성된 회로를 해석할 수 있는데 반하여, 회로 시뮬레이션은 1000개 정도의 반도체 소자들로 구성된 회로를 해석할 수 있다. 해의 정확도 관점에서는, 회로 시뮬레이션이 가장 정확한 해의 결과를 얻을 수 있다. 특히, IC 설계에서의 timing analysis 관점에서 보면, 기능 시뮬레이션 또는 논리 시뮬레이션은 주어진 소자 지연시간을 기초한 system timing 해석만이 가능한데 반하여, 회로 시뮬레이션에서는 주어진 회로 정보에 따른 신호 레벨값, 지연시간 등의 timing analysis 결과를 자동적으로 얻을 수 있다. 또한, 기능 및 논리 시뮬레이션에서는 주로 논리 회로 또는 논리 시스템의 해석만이 가능한데 반하여, 회로 시뮬레이션에서는 회로의 형태에 관계없이 능동회로와 논리 회로를 동시에 해석할 수 있다.

III. 논리 시뮬레이션

앞에서 언급한 바와 같이, 논리 시뮬레이션은 논리 회로의 논리 해석을 그 목적으로 한다. 논리 시뮬레이션은 수십만개 이상의 논리 소자들로 구성된 논리 회로를 SPICE에 비하여 100배 이상의 빠른 속도로 해석할 수 있다. 논리 시뮬레이션을 위하여 사용자가 정의하여야 하는 회로 정보는 논리 소자의 기능 정보, 연결구조 정보, 지연시간 정보 등이 있다. 논리 시뮬레이션에서 사용하는 모든 신호값들은 정수 개념으로 정의가 된다. 예로서, 시간의 표현도 기본적인 MRT(minimum resolvable time)의 정수배로 표시가 되며, 논리 신호도 discrete logic state로 정의가 된다. 이 장에서는, 논리 시뮬레이션에서 채택하고 있는 논리 신호의 정의, 여러 종류의 지연시간 모델 및 해석 방식에 대하여 설명하고자 한다.

1. 논리 신호

일반적으로, 논리 신호는 논리 상태(logic state)와 논리 세기(logic strength)의 조합으로 정의가 된다. 논리 상태는 신호의 전압 레벨을 추상화한 것으로, 1(true)과 0(false)의 binary value가 기본이다. 일반적으로는, 1과 0의 기본 논리 상태 이외에 X상태(unknown state)가 첨가된다. X상태는 (1) 능동 전압 신호의 관점에서 1과 0의 중간적인 상태값(intermediate state), (2) 1 또는 0으로 정의할 수 없는 상태(undetermined state), (3) 0에서 1로, 또는 1에서 0으로의 상태 천이 현상(rising/falling transition state) 등을 포

괄적으로 포함하고 있다. 경우에 따라서는, X상태를 여러 가지 논리 상태로 세분화하여 사용하기도 한다. 논리 상태가 정의되면, 소자들의 기능은 정의된 논리 상태를 변수로 하여 입·출력 관계가 truth table 또는 state transition table로서 묘사된다. 1, 0, X의 ternary 논리 상태를 사용한 경우의 m-input 논리소자의 기능을 나타내기 위하여서는 3^m의 입력 경우에 대한 소자 진리표가 필요하다. 초기 논리 상태의 결정은 t=0에서의 입력값을 가지고 지연시간을 고려하지 않는 회로의 논리 해석을 수행하여 얻을 수 있다. 그러나, ring oscillator 또는 flip-flop의 경우에는 t=0에서의 값을 결정할 수 없는 경우도 있다. 이를 위하여서는 initial unknown state(Xi)를 정의하여 노드의 초기값으로 준다. Xi는 시뮬레이션 도중에 발생하는 X상태와는 구분이 되며, 어떤 노드의 값이 Xi라는 의미는 그 노드가 아직 해석되지 못하였다는 것을 의미한다.

논리 신호를 논리상태로만 표시하는 경우에는 tristate logic, MOS 논리 회로의 pass transistor logic, dynamic logic의 해석에 있어서 부정확한 결과를 얻을 수 있다. 이상의 회로에서는 회로내의 한 노드 또는 일부분이 어떤 특정 시간동안 외부와는 open되는 현상과 wired-logic이 나타나게 되므로 이를 위하여서는 논리 상태 이외에 논리 세기에 대한 정의가 필요하다. 논리 세기란 해당되는 노드와 외부 회로 사이의 신호 전달도(즉, impedance condition)를 의미한다. 만약, impedance condition이 적으면(즉, 저항 성분이 거의 0) 그 노드의 신호 상태가 외부로 쉽게 전달될 수 있으나, 반대로 impedance condition이 크면 그 노드의 신호 상태는 외부로 전달되기가 어렵다. 논리 세기는 supply, driving, resistive, high-impedance 등으로 구분된다. 논리의 세기가 supply인 경우는 회로의 외부 입력 노드, 전압 전원, 접지 등에서의 신호 세기를 의미하며, 신호의 전달 세기가 제일 강하다. Driving 세기는 회로내의 일반적인 논리 소자들의 출력 노드에서의 신호 세기를 의미한다. Resistive 세기는 해당되는 노드가 midium resistance path로서 외부와 연결되어 있을 경우로서 NMOS logic회로에서 출력이 0인 경우가 한 예가 된다. High-impedance 세기는 노드가 외부와 차단된 경우로서 pass-transistor가 open된 경우에 발생한다. High-impedance state는 신호의 세기가 제일 약한 경우로서, 신호의 전압값이 floating grounded capacitor에 저장되어 있다. High-impedance 세기를 가지는 신호는 외부에 영향을 줄 수는 있지만, 일정 시간이 지나면 원래의 신호 상태를 잃어버릴 수 있는 temporary memory state를 의미한다. 논리 세기는 wired logic에서처럼 상이한 논리 상태가 한 노드에서 상충되는 경우에 논리의 세기를 비교함으로서 그 노드의 논리 상태를 결정하는 경우에도 사용이 된다.

이상의 설명에서, 논리 시뮬레이션에서의 논리 신호는 논리 상태(전압 레벨의 추상화)와 논리 세기(저항의 추상화)를 조합하여 표시한다. 예로서, 어떤 노드의 신호 상태가 1이고, 논리 세기가 high-impedance이면, Z1(high-impedance-1)으로 논리 신호가 정의된다. 일반적으로, 논리 소자의 출력 신호값은 입력 신호의 논리 상태로서 출력 신호의 논리 상태가 결정이 되며, 출력 신호의 세기는 논리 소자의 내부 성질에 의하여 결정이 된다. 이 절에서 논의한 논리신호의 상태와 세기에 관하여 그림 1에서 정리하였다. 그림 1은 3종류의 논리 상태와 4종류의 논리세기를 조합하여 12종류의 논리신호들을 보여준다. 논리 상태와 세기의 종류는 (switch-level을 포함하여) 논리 시뮬레이터마다 시뮬레이션의 목적에 따라 다르게 정의할 수 있다. 회로 설계자는 사용하고 있는 논리 시뮬레이터에서 정의한 논리 신호들의 의미를 정확히 파악하고 사용하여야 한다.

strength / state	SUPPLY	DRIVING	RESISTIVE	HIGH-Z
1	S1	D1	R1	Z1
X	SX	DX	RX	ZX
0	S0	D0	R0	Z0

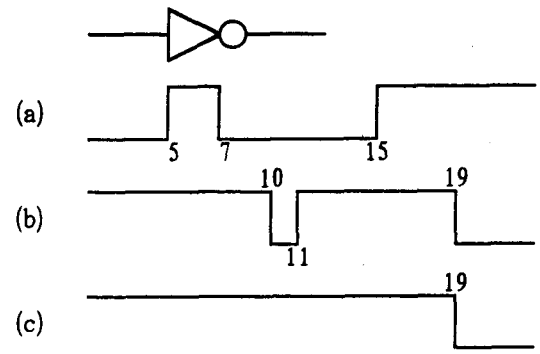
〈Fig.1〉 Logic state and strength.

2. 지연시간

모든 논리 소자는 입력신호에 대하여 일정 시간이 경과한 후에 출력 신호가 반응된다. 이상의 일정 경과 시간을 지연시간으로 표시하는데, 논리 시뮬레이션에서는 여러 가지 모델을 사용할 수 있다. 논리 소자의 지연시간 모델로는 (1) zero delay, (2) unit delay, (3) assignable delay, (4) min-max delay 등이 있다. Zero delay는 논리 소자의 지연시간을 고려하지 않는 경우이며, unit delay는 논리 소자의 지연시간이 최소의 시간 단위인 one-MRT로서 정의된 경우이다. Assignable delay는 논리 소자의 지연시간을 사용자가 임의로 정의할 수 있는 경우이며, min-max delay는 worst-case analysis로서 출력의 반응이 min-delay까지는 나타나지 않고, max-delay 이후에는 출력의 반응이 나타난다고 지연시간을 정의한다. 이 경우, min-delay와 max-delay 사이의 시간에는 출력 신호가 X상태로 표시가 된다. 논리 소자의 지연시간의 정의는 입력 파형이 50% 변화한 시간과 그에 반응된 출력 파형이 50% 변화된 시간의 차로서 정의가 된다. 능동 입·출력 파형을 고려한 경우에는 출력 파형이 10%에서 90%까지 변화하는 시간을 천이시간(transition time)으로 추가하여 정의하기도 한다. 그러나, 대부분의 논리 시뮬레이션에서는 천이 시간을 고려하지 않는다.

출력 신호의 천이 상태에 따른 지연시간의 차이를 표시하고자 할 때에는 rise/fall delay로서 정의한다. Rise delay는 출력의 신호가 0→1로 변화할 경우의 지연시간이며, fall delay는 출력 신호가 1→0으로 변화될 경우의 지연시간이다.

입력 신호의 pulse 폭에 따른 지연시간의 모델로는 전달 지연(transport delay)과 관성 지연(inertial delay)가 있다. 전달 지연이란 입력의 신호에 따른 출력의 신호의 반응이 무조건 정의된 지연시간 이후에 나타나는 경우를 말하며, 관성 지연에서는 입력 신호의 pulse 폭이 관성 지연시간 이상의 pulse 폭을 가져야만 출력 신호가 반응된다는 것을 의미한다. 예로서, 그림 2는 assignable delay model로서 rise delay가 4이고 fall delay가 5로 주어진 한 inverting gate의 입·출력 파형을 보여



〈Fig.2〉 Inverter gate(TR=4 and TF=5). (a) Symbol and input waveform. (b) Output waveform : transport delay. (c) Output waveform : inertial delay.

준다. Inverting gate의 입력 신호가 시간 5에서 0에서 1로 변화하였고, 시간 7에서 1에서 0으로 변화한 경우를 생각하여 보자. 이때, 관성 지연이 없는 순수 전달 지연에서는 출력 신호가 시간 10에서 1에서 0으로 변화하고, 시간 11에서 0에서 1로 변화한다. 그러나, 관성 지연시간을 3으로 주면, 시간 5와 시간 7사이의 pulse 폭이 관성 지연시간보다 적으므로 입력 신호에 대하여 반응을 하지 않게 되어 출력 신호는 1을 그대로 유지하고 있게 된다. 관성 지연시간과 전달 지연을 같은 값으로 준 경우를 전파 지연(propagation delay)라고 한다. 전파 지연은 기본 논리 소자의 지연 시간 모델로서 사용을 하며, 관성 지연과 전달 지연을 달리한 경우는 flip-flop 등과 같이 여러 개의 논리 소자가 조합된 논리 블록의 지연시간 모델로서 사용한다.

이상의 지연시간 모델 이외에, 논리 시뮬레이션에서는 high-impedance 세기에서의 신호 상태의 유지시간을 표시하기 위하여, reflesh time 또는 decay delay로서 신호 유지시간을 정의하는 경우도 있다. 이 경우, high-impedance 노드는 high-impedance 세기가 되기 직전의 신호 상태를 high-impedance 세기로 reflesh time 동안만 유지하고, reflesh time 이후에는 high-impedance 세기의 미지 상태로 변화된다. 또한, 연속적인 짧은 펄스들에

대한 반응을 표시하기 위하여, early/late/compensate 등의 spike 발생 표시법 등도 사용하고 있다.

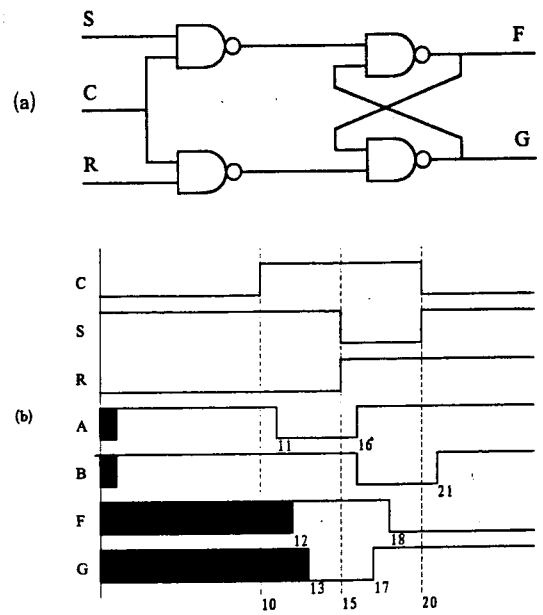
한 논리 소자의 지연시간은 (1) 논리 소자의 내부 구조, (2) fanout, (3) 입력 파형, (4) 출력 신호의 변화에 영향을 미치는 입력 노드의 위치 등에 의하여 결정이 된다. 논리 소자의 내부 구조에 따른 지연시간은 논리 소자의 설계 양식에 따라 좌우되며, fanout에 의한 지연시간의 변화는 fanout의 수에 비례한 선형적인 수식으로 표시가 된다. 입력 파형에 대한 지연시간의 변화는 입력 파형의 천이 시간 또는 입력 파형의 기울기에 의하여 수식으로 표시된다. 여러 개의 입력 노드가 있는 경우에 각각의 입력 노드에 따른 출력 신호의 반응이 다르게 나타날 수 있는데, 이는 pin-to-pin delay로서 표시하기도 한다. 지연시간의 결정은 실험에 의한 측정 또는 회로 시뮬레이션의 사용으로 해결하게 된다.

논리 시뮬레이션에서의 지연시간 모델은 실지 회로에서 나타나는 여러 가지 시간 지연 현상을 추상적으로 모델한 것이다. 따라서, 사용자는 논리 소자의 지연시간을 올바르게 표시할 수 있는 지연시간 모델을 선택하고, 회로 시뮬레이션 등의 방법에 의하여 정확한 지연시간 모델식의 계수값을 결정하여야 한다.

3. 해석 방법

논리 시뮬레이션의 해석 방법은 사건 구동 방식(event-driven method)가 대표적이다. 여기서, 사건(event)이라 함은 한 노드의 신호가 논리 상태를 변화한 경우를 말한다. 이제, 한 노드에서 사건이 발생하면 그 노드를 입력으로 하고 있는 논리 소자들을 추적하고(selective-trace), 추적된 논리 소자들 각각에 대하여 발생된 입력 사건에 대한 출력 노드의 신호값을 논리 소자의 기능과 지연시간을 고려하여 계산한다(event driven). 이후, 출력 노드에서 새로운 사건이 발생하면 이상의 과정을 반복하여 더 이상의 새로운 사건이 발생하지 않을 때까지 해석 과정을 반복한다. Selective-trace and event-driven 방법을 효과적으로 처리하기 위하여 발생된 새로운 사건들은 TQ(time queue)에 저장한다. 결국, 사건 구동 방식은 TQ에 등록되어 있는 사건들을 하나씩 가지고 와서 관련 소자들을 selective-trace and event-driven 방법으로 해석하고 새로이 발생된 사건들은 TQ에 등록하는 과정은 반복 수행한다. 그림 3은 SR-FF에 사건 구동방식을 적용한 예를 보여준다.

사건 구동 방식 이외에 논리 시뮬레이션에서 채택하고 있는 해석 방법들로 time-first method, compiled-code approach, parallel process를 이용



0	S(1)	R(0)	C(0)	
	A(X)	B(X)	F(X)	G(X)
1	A(1)	B(1)		
10	C(1)			
11	A(0)			
12	F(1)			
13	G(0)			
15	S(0)	R(1)		
16	A(1)	B(0)		
17	G(1)			
18	F(0)			
20	S(1)	C(0)		
21	A(0)*	A(1)*	B(1)	

〈Fig.3〉 Example of the event-driven method. (a) SR-FF : unit delay. (b) Waveforms. (c) Time-queue.

한 방법 등이 있다.

IV. 회로 시뮬레이션

회로 시뮬레이션은 저항, 커패시터, 인덕터, 반도체 소자 등의 기본 회로소자들로서 구성된 회로의 해석을 목적으로 한다. 회로 시뮬레이션의 대표적인 예로서는 SPICE, ECAP, ASTAP 등이 있는데, 이들은 모두 다양한 해석 기능을 가지고 있다. 회로 시뮬레이션은 DC 해석, 과도시간 해석, 주파수 해석, distortion 해석, fourier 해석, 잡음 해석, 온도 해석, sensitivity 해석 등의 회로 설계자가 원하는 거의 모든 해석 기능을 가지고 있다. 회로 시뮬레이션은 약 1000개 이하의 회로 소자들로 구성된 회로를 1% 이내의 정확도를 가지고 해석할 수 있다. 문제점으로는 대규모 회로의 해석이 어렵고, 계산 시간이 회로소자(또는 회로 노드) 숫자, n의 지수 승으로 증가하여(이론적으로는 n^3, 실제적으로는 $n^{1.5\sim2}$), 회로의 크기가 증가할 수록 계산 속도가 느려진다는 것이다. 이 장에서는 회로 시뮬레이션의 해석 방식, 해의 정확도를 높이는 방법, 회로 시뮬레이션의 문제점을 개선한 시간 시뮬레이션에 대하여 설명하고자 한다.

1. 해석 방식

앞에서 언급한 회로 시뮬레이션의 해석 기능들 중, 해석 방식의 관점에서 기본이 되는 해석 기능들은 (1) DC 해석, (2) 주파수 해석, (3) 과도시간 해석이 있다. DC 해석은 회로의 DC 정상 상태를 해석하는 경우로서, 회로내의 모든 커패시터를 open하고 모든 인덕터를 short시킨 후에 남은 선형/비선형 저항 성분들에 의한 회로의 동작을 해석한다. 주파수 해석은 어떤 특정한 DC 상태(즉, DC-q point)에서 모든 회로 소자들을 선형화한 후에 phasor analysis에 의하여(small signal) AC 정상상태를 구하는 경우이다. 과도시간 해석은 주어진 입력 파형에 대한 회로 내의 모든 노드 전압 파형을 전해석 시간 동안 해석하는 경우를 말한다. 결국, 과도시간 해석은 해석 기법의 관점에서 DC 해석과 주파수 해석을 포함하게 되며, 실지로 가장 많은 해석 시간이 소요된다. 과도시간 해석은 비선형상미분방정식 형태의 회로 방정식의 형성과 해석을 수행한다. 회로 방정식에서 비선형 성질은 주로 반도체 소자들에서 나타나는 비선형 소자 특성에 따라 나타나며, 미분항들은 커패시터나 인덕터에 의하여 나타난다. 회로 시뮬레이션에서의 회로 방정식의 형성과 해석을 위하여 사용하고 있는 방법들은 (1) MNA(modified nodal analysis), (2) 수치 적분, (3) Newton-Raphson 선형화법, (4) LU decomposition, (5) time-step control, (6) companion model appoarch 등이 있다.

여러 가지 회로 방정식 형성 방법들 중에서 nodal analysis는 노드 전압들을 회로 변수로 하여 각각의 노드에 KCL을 적용한 후, KVL과 소자 특성식을 단계적으로 적용한 nodal equation을 얻는 방법이다. 이 방법은 매우 간단하고, nodal element stamp를 사용하면 쉽게 computer implementation할 수 있다. 그러나, 전압 전원, 또는 impedance-type element, current controlled element가 있는 경우에는 nodal equation을 직접 형성할 수 없는 문제점이 발생한다. 이러한 문제점을 해결하기 위하여 nodal analysis를 적용하기 어려운 회로 소자들마다 소자 전류를 회로 변수로 추가 선택하고, 해당되는 소자 특성식은 추가적인 회로 방정식으로 채택하는 방법이 MNA이다. MNA는 모든 형태의 회로에 대하여 쉽게 implementation을 할 수 있는 nodal analysis의 변형이다.

회로 방정식이 형성되면 그 형태는 다음과 같은 비선형 미분 방정식이 된다.

$$F(x(t), \dot{x}(t), t) = 0, \quad with\ x_0 = x(0) \quad (1)$$

여기서 $x \in R^n$는 n개의 회로 변수 벡터이고, F는 n개의 비선형 방정식을 의미한다. (1)을 풀기 위하여서는 수치 적분법이 적용된다. 수치 적분이란 해석시간을 여러 개의 time-point로 나누고, 매 time-point마다 미분을 근사하여 미분 방정식을 해석하는 방법이다. 여러 가지 종류의 수치 적분법

중에서 SPICE에서 사용하고 있는 수치적분법은 Euler-backward(EB)법 또는 trapezoidal법이다. (1)에 시간 $t=t_{n+1}$에서 EB법을 적용하면,

$$F(x_{n+1}, \frac{x_{n+1}-x_n}{h_n}, t_{n+1}) = F(x_{n+1}) = 0 \quad (2)$$

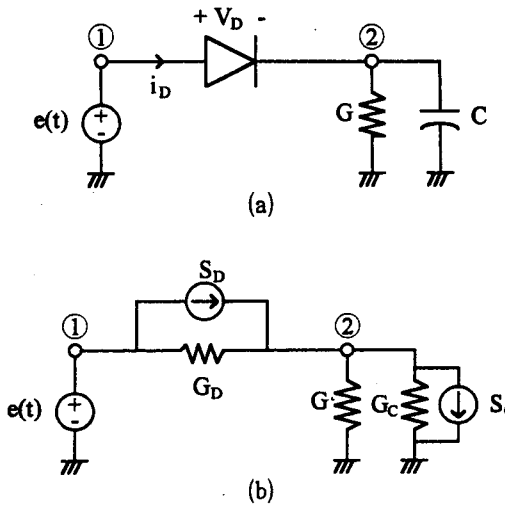
여기서, $x_{n+1} = x(t_{n+1})$이고 $h_n = (t_{n+1} - t_n)$이다. (2)는 x_{n+1}를 미지수로 하는 비선형 방정식이 된다. (2)의 비선형 함수를 Taylor 전개하여 1차 항까지만을 취하는 Newton-Raphson 선형화법을 적용하면 다음과 같은 선형 방정식이 구하여진다.

$$F(x_{n+1}) \cong J(X_{n+1})x_{n+1} + [F(X_{n+1}) - J(X_{n+1})X_{n+1}] = 0 \quad (3)$$

여기서, X_{n+1}은 x_{n+1}의 가정치이며, $J(X_{n+1})$은 X_{n+1}에서의 Jacobian matrix이다. 식(3)의 선형 방정식을 풀기 위하여 LU 분해법이 적용된다. LU 분해법은 주어진 행렬을 두 개의 삼각 행렬로 바꾸어 행렬 방정식의 해를 구하는 방법이다.

$$x_{n+1} = X_{n+1} - J(X_{n+1})^{-1} * F(X_{n+1}) \quad (4)$$

회로 시뮬레이션에서는 매 time-point마다 이상의 수치 해석법들을 반복 적용하여 가면서 해석시간을 증가하여, 전 해석시간 동안의 (1)의 해를 구하게 된다. 수치 적분과 Newton-Raphson 선형화법을 효과적으로 적용하기 위하여서는 방정식을 형성하기 전에 미리 각각의 소자에게 수치적분법과 Newton-Raphson 선형화법을 적용한다. 이러한 방법을 companion-medel 방식이라 하는데, 이는 computer implementation을 쉽게 하기 위한 방법이다. 그림 4는 diode-RC 회로에 대하여 companion 방식에 의하여 수치적분 (EB법)과 Newton-Raphson 선형화법을 소자별로 적용한 결과를 보여준다. 선형화된 회로에 대한(reduced) MNA 식은 다음과 같다.



〈Fig.4〉 Diode RC circuit. (a) Original circuit. (b) Linearized companion circuit.

$$\begin{bmatrix} 1 & 0 \\ -G_D & (G+G_D+G_C) \end{bmatrix} \begin{bmatrix} v_{1,n+1} \\ v_{2,n+1} \end{bmatrix} = \begin{bmatrix} e(t_{n+1}) \\ S_D - S_C \end{bmatrix} \quad (5)$$

여기서, $i_D = I_S(\exp\left(\frac{V_D}{V_T}\right)-1)$, $F_D = \exp\left(\frac{V_{D,n+1}}{V_T}\right)$, $G_D = \frac{I_S}{V_T}F_D$, $S_D = I_S(F_D - 1) - G_D V_{D,n+1}$, $G_C = \frac{C}{h_n}$, $S_C = -G_C v_{2n}$이며, $V_{D,n+1} = (V_{1,n+1} - V_{2,n+1})$로서 $V_{D,n+1}$의 가정치이다.

이제, 한 time-point 해석이 끝난 후에 다음 time-point로 넘어가기 위하여서는 timestep, h_n의 결정이 필요하다. Timestep 조절은 회로 시뮬레이터의 성능을 결정하는 아주 중요한 부분이다. Timestep을 너무 크게 잡아주면 계산속도는 증가하지만 해의 정확도가 떨어지며, 반대로 timestep을 너무 작게 잡아주면 해의 정확도는 증가하지만 계산속도가 느려진다. Timestep의 결정은 (1) Newton-Raphson 선형화법에서의 반복회수에 의하여 timestep을 결정하는 iteration timestep control법과 (2) 매 time-point에서의 계산된 수치 적분 오차(LTE)에 의한 LTE time-step control법이 있다. 이 중에서, LTE법이 보다 정확한 결과해

를 보여주고 있으며, 일반적인 회로 시뮬레이션에서 사용하고 있는 방법이다.

계산 시간의 관점에서 보면, (1) Newton-Raphson 선형화법을 적용하기 위한 비선형 함수의 계산과 (2) 선형 방정식을 풀기 위한 LU 분해 계산에서 대부분의 계산 시간이 소요된다. 비선형 함수의 계산은 회로 소자의 수에 비례한 계산 시간을 보여주고 있으며, LU 분해의 계산 시간은 회로 변수의 3 지수승에 비례한다. 단, sparse matrix technique를 사용한 경우에는 회로 변수의 1.5내지 2 지수승에 비례한다. 회로의 크기가 작으면 비선형 함수의 계산이 상대적으로 비중이 커지며, 회로의 크기가 크게되면 LU 분해의 계산이 상대적으로 비중이 커지게 된다.

2. 해의 정확도

회로 시뮬레이션을 사용하는 경우에는 계산 시간이 많이 소요되더라도 정확한 결과해를 얻기 위한 것이 주목적이다. 회로 시뮬레이션에서 결과해에 영향을 미치는 요소들로는 (1) 반도체 소자의 모델식과 모델 변수에서의 오차, (2) 기생소자들에 의한 오차, (3) 해석방식에서의 알고리즘 오차가 있다. 회로 시뮬레이션의 사용자는 무엇보다도 먼저 적합한 반도체 소자의 모델식을 선정하여야 한다. 일반적으로, 반도체 소자들은 물리적인 해석적 모델과 제작 공정과 연관된 모델 변수값에 의하여 소자 특성이 규정된다. 반도체 소자들의 모델식은 저항 성분을 고려한 static model과 커패시터, 인덕터 성분을 고려한 dynamic model으로 구분되며, 일반적으로 (1) BJT는 Gummel-Poon model, (2) MOSFET는 Schiman-Hdges model과 BSIM model 등을 기초로 한 여러 가지 실험적인 모델들이 개발되어 사용되고 있다. 이제, 반도체 소자 모델식이 결정되면 해당되는 모델식의 모델 변수값을 정확히 알아야 한다. 이 과정을 model parameter extraction이라 하는데, 반도체 제작 공정과 매우 밀접한 연관 관계가 있다. 따라서, 일반적인 회로의 설계자가 반도체 제작 공정에 대한 정보 없이 반도체 소자 모델식과 모델 변수값을 결정할 수 없다. 오늘날, 대부분의 반도체 회사들은 자체적으로 반도체 소자 모델식과 보델 변수값에 대한 정보를 회로 설계자에게 공급하고 있다. 여기서 주의할 점은 같은 반도체 소자라 할지라도 반도체 회사에 따라서 반도체 제작공정의 차이 때문에 모델 변수값들은 다르다. 회로 설계자는 우선 IC 제작을 의뢰할 반도체 회사를 선정한 후, 그 회사에서 제공하는 반도체 소자에 대한 모델식과 모델 변수를 가지고 설계에 임하여야 한다. 최근에는, 소자의 크기가 작아짐에 따라 기존의 소자 모델식으로는 부정확한 결과를 얻는 경우가 많아지고 있다. 이 경우에는, 물리적인 해석 모델식보다는 경험과 실험에 의한 근사식 또는 table 형태의 모델식이 반도체 소자 모델로서 사용되고 있다. 새로운 형태의 모델식을 사용한 경우에는 최소한도 1차 미분값까지는 연속이 되도록 모델식을 결정하여야 회로 시뮬레이션의 해석에서 문제가 발생하지 않는다.

해석 방식에서의 오차는 미분의 근사식인 수치적분법과 비선형 함수의 선형화 과정인 Newton-Raphson 방법에서 발생한다. 이 외에, 컴퓨터의 word-length 제한에 따른 round-off 오차도 결과해에 영향을 미치게 된다. 그러나, 이상의 해석 방식에서 오차가 발생할 경우에는 시뮬레이션 프로그램의 내용을 바꾸어야 하므로, 일반적인 회로 설계자로서는 이를 해결할 방법이 없다. 다만 부분적으로나마 이 문제를 해결하기 위해서는 option 변수값을 바꾸어 주는 것이 한 방법이다. 회로 설계자가 쉽게 접근할 수 있는 option 변수들로는 (1) 상대 오차, (2) 절대 오차, (3) 수치 적분 방법, (4) time-step control법 등이 있다. 여기서, 참고하여야 할 사항으로는 이상의 option 변수값들을 보다 엄격히 바꾼다 하여 해의 정확도가 개선되지는 않는 것이 일반적인 현상이다. 예로서 상대 오차를 0.1%(default값)에서 0.01%로 바꾼다고 하여 해의 정확도는 개선되지 않고, 오히려 해석 불능의 현상이 나타날 가능성이 있다. 이는, round-off error 때문에 컴퓨터의 word-length가 증가하지 않는 한 해결할 수 없는 문제이다. 따라서, option 변수를 바꾸는 경우는 해석 불능의 현상(too small timestep 또는 DC nonconvergence)이 일어

날 경우에 option값은 조금 여유있게 바꾸어 허용 오차를 조금 증가시켜서 해석을 수행하는 경우이다.

3. 시간 시뮬레이션

회로 시뮬레이션은 해는 정확하게 구할 수는 있지만, 계산 속도와 회로의 크기의 관점에서 제약이 주어진다. 따라서 어느 정도 해의 오차를 허용하는 대신에 계산 속도와 회로의 크기에 있어서 잇점을 얻을 수 있는 방법으로 시간 시뮬레이션이 있다. 시간 시뮬레이션에서 사용하고 있는 방법들은 대부분의 전자 회로에서 보여 주고 있는 temporal sparsity, structural sparsity, multirate behavior 등의 성질을 이용하여, (1) 반도체 소자 모델식의 간소화, (2) bypass scheme, (3) 수치 적분 및 시간격 조절법의 간소화, (4) 회로 분할 등의 방법들을 사용한다. 이상의 방법들 중에서, 계산 시간과 회로 크기를 가장 효과적으로 개선할 수 있는 것은 회로 분할에 의한 해석 방식의 변형이다.

회로 분할에 기초를 둔 해석 방식들은 선형·비선형·파형 이완 방식(linear/nonlinear/waveform relaxation)과 ITA(iterated timing analysis)가 대표적이다. 이완 방식은 반복 선형 해석 방식의 일종인 Gauss-Jacobi법, Gauss-Seidel법, SOR (Successive over relaxation)을 회로 시뮬레이션의 각 단계에 적용하는 방법이다. 예로서, 반복 선형 해석 방식을 회로 시뮬레이션에서의 선형 방정식의 해석 단계에 적용하면 선형 이완 방식이 되고, 비선형 방정식의 해석 단계에 적용하면 비선형 이완 방식이 되고, 미분 방정식의 해석 단계에 적용하면 파형 이완 방식이 된다. ITA는 비선형 이완 방식에 selective-trace and event-driven 방식을 혼합하고 있다. 이상의 방법들 중에서, 현재 널리 사용되고 있는 것으로는 Gauss-Seidel법에 의한 비선형 이완 방식, ITA, 파형 이완 방식들이다.

회로 분할에 기초를 둔 해석 방식의 공통적인 문제점으로는 해의 수렴 속도이다. 이상의 회로 분할 방법들은 모두 반복법을 채택하고 있으므로, 수렴 조건에 의하여 해석할 수 있는 회로의 형태가 매 노드마다 grounded capacitor를 가지고 있는 MOS 논리 회로에 국한된다. 초기의 시간 시뮬레이션에서는 노드 분할에 근거한 회로 해석을 수행하였으나, 이 경우 floating capacitor/resistor 등에 의하여 해의 수렴 속도가 현저히 떨어진다. 시간 시뮬레이션에서의 회로 분할은 해석 결과에 매우 중요한 영향을 미친다. 최적의 회로 분할은 전기적으로 독립성을 가지도록 회로 블록을 나누는 것이다. 여기서, 전기적 독립성이란 high input impedance와 low output impedance를 의미한다. 즉, 분할된 회로 블록들이 자체적으로 독립적인 기능을 가지고, 외부와는 단지 입.출력 신호 정보만을 교환하도록 회로 분할을 수행하는 것이다. 아직까지 효과적인 자동 회로 분할법은 개발되고 있지 않다. 다만, MOS 논리 회로의 경우 DC path를 중심으로 회로 분할을 수행하는 방법이 있다. 따라서, 회로 분할은 회로 설계자가 담당하여야 할 영역이다.

이상의 방법들 이외에, variable precision 방법이 시간 시뮬레이션에 유용한 것으로 나타나 있다. 이 방법은 매 time-point마다 미리 변화할 수 있는 전압의 범위를 설정한 후, 이를 위한 시간격을 조절하는 방법이다. 즉, 기존의 LTE 시간 조절법에 상대되는 개념으로 전압 단위로 시간격을 조절하는 방법이다.

결론적으로, 시간 시뮬레이션은 회로의 크기, 계산 속도 등의 관점에서 회로 시뮬레이션에 비하여 잇점을 얻을 수 있다. 그러나, 해석 방식의 수렴 조건에 의하여 해석할 수 있는 회로의 형태가 MOS 논리 회로로 국한되어 있으며, 해의 정확도도 10% 이상의 오차를 나타낸다. 회로 설계자가 시간 시뮬레이션을 사용하고자 할 때에서, 해석하고자 하는 회로의 형태, 적합한 해석 방식의 선택 등을 고려하여 시간 시뮬레이션을 동작시켜야 한다.

V. 맺 음 말

이 글에서는 IC 설계를 위한 시뮬레이션에 관하

여 언급하였다. IC 설계를 위한 시뮬레이션은 IC 설계 및 제작단계에 맞추어 여러 가지 다양한 종류들로 구분된다. 시뮬레이터들은 일반적인 설계환경에 맞추어 설계가 되었으며, 나름대로의 문제점들을 가지고 있다. 따라서, 여러 가지 시뮬레이터들 중에서 IC설계자는 설계 단계와 설계 목적에 적합한 시뮬레이터를 선택·사용하여야 한다. 효과적인 시뮬레이터의 사용을 위하여는 정확한 설계 환경의 입력과 더불어 시뮬레이터 내부의 해석 방식에 관한 올바른 지식이 있어야 한다, 회로 설계자가 시뮬레이터의 사용시 고려하여야 할 점으로 (1) 회로의 형태, (2) 회로의 크기, (3) 회로 분할의 타당성, (4) 정확한 회로 기능 및 구조의 기술, (5) 신호의 의미, (6) 정확한 모델식의 선정과 모델 변수값의 결정, (7) 기생소자들의 처리, (8) 결과해의 의미, 정확도 및 분석, (9) timing 정보의 분석, (10) 시뮬레이터 입력 해석 변수들의 의미, (11) 해석 방식의 장·단점 등이 있다.

이 글에서 언급하고 있는 논리와 회로 시뮬레이션은 주로 하위 레벨의 회로 설계에 사용된다. 상위 레벨의 시스템 설계를 위한 시뮬레이션으로는 VHDL 시뮬레이션, DSP 시뮬레이션 등의 기능 시뮬레이션이 있다. 최근의 시뮬레이션 분야의 연구 경향은 보다 빠른 시뮬레이터의 개발, 보다 정확한 시뮬레이터의 개발, 설계 환경의 변화에 부응하는 시뮬레이터의 개발 등이 있다. 이를 위하여 연구가 진행되고 분야는 회로 분할 방법, 능동 기능 시뮬레이션, 설계 환경의 정확한 모델 개발, 기생 소자의 해석, 안정된 해석 알고리즘 등이 있다.

끝으로, IC설계는 IC설계자의 능력 배양과 올바른 CAD장비의 사용이 조화를 이루어야 한다. 이를 위하여서는 IC 설계 교육 방법의 개발, 활성화된 MPC작업, CAD tool의 개발 및 사용 등에 대한 연구들이 요구된다.

참 고 문 헌

[1] N. H. E. Weste and K. Eshraghian, "Principles of CMOS VLSI design", 2nd ed., Addison-Wesley, 1993.

[2] J. Maver, M. A. Jack and P. B. Denyer, "Introduction to MOS LSI design", Addison-Wesley, 1983.

[3] R. A. Salch and A. R. Newton, "Mixed-Mode Simulation", Kluwer Academic Pub, 1990.

[4] "HSPICE user's manual", Meta-software, 1992.

[5] L. W. Nagel, "SPICE2 : A computer program to simulate semiconductor curcuits", Memo ERL-M520, U. C. Berkeley, May, 1975.

[6] "SILOS II user's manual", Simucad, 1988.

[7] "HILO2 user's manual", Gen Rad, 1984.

[8] A. R. Newton and A. L. Sangiovanni-Vincentelli, "Relaxation-Based Electrical Simulation", IEEE Trans. on CAD, vol, CAD-3, pp. 308~331, Oct. 1984.

저 자 소 개

李 起 焌

1955年 1月 23日生
1978年 2月 서울대학교 공과대학 공업교육학과 전자전공 졸업(공학사)
1981年 2月 한국과학기술원 전기 및 전자공학과 졸업(공학석사)
1986年 2月 한국과학기술원 전기 및 전자공학과 졸업(공학박사)

1986年 9月～1990年 9月 충남대학교 전자공학과 조교수
1990年 10月～현재 충남대학교 전자공학과 부교수

주관심분야 : VLSI-CAD, simulation