병렬처리 컴퓨터를 이용한 논리 시뮬레이션 기법

鄭 然 模
慶熙大學校 電子工學科

Ⅰ. 서론

복잡한 VLSI 칩을 설계하기 위해서는 비용과 시간이 많이 든다는 것은 잘 알려진 일이다. 칩을 생산하기 전에 가능한 모든 오류를 찾아내는 것은 필수적인 일로서 지금까지 이를 위한 도구로서 시뮬레이션(simulation)이 사용되어져 왔다. 시뮬레이션은 설계를 검증하는 단계로서 가장 힘든 과정중의 하나이다. 회로가 복잡해지고 집적도가 높아짐에 따라 이를 위한 시간이 엄청나게 길어지고 있으므로 이를 위한 해결책으로서 미국 등 선진국에서는 병렬처리 컴퓨터를 이용한 고속 시뮬레이션을 위한 연구가 활발히 전개되고 있다.

본고의 제 2 장에서는 처리 수준 및 지연 처리에 따른 시뮬레이션의 종류를 알아보고 게이트 수준(gate-level)의 논리 시뮬레이션의 특징을 조사해 보기로 한다. 이어서 제 3 장에서는 고속 시뮬레이션을 위해서 병렬처리 컴퓨터를 사용하는 경우에 고려할 사항, 기존의 알고리듬의 종류와 특징에 대해서 알아본다. 이들 알고리듬을 대단위 병렬처리 SIMD 컴퓨터(Massively Parallel SIMD Computer)로써 CM-2와 MP-1상에서의 실제 수행 능력의 평가에 대해 제 4 장에서 다룬다. 마지막으로 시뮬레이션의 이용 방향 및 향후 연구과제에 대해서 알아보기로 한다.

Ⅱ. 시뮬레이션의 종류 및 특징

VLSI 설계 과정에서 시뮬레이션 결과들은 특정한 벡터(또는 파형)를 회로 설계에 입력함으로서 만들어지며, 이러한 결과들은 그 설계의 기능적 정확성과 설계에 대한 타이밍을 검증하기 위해서 사용된다. 신뢰성있는 회로 설계를 위해서는 컴퓨터 시뮬레이션에 많은 시간이 투자된다. 시뮬레이션에는 여러가지 종류가 있는데 설계되는 회로의 성질, 크기, 설계 목적, 처리 수준에 따라서 원하는 시뮬레이터를 사용하게 된다.

1. 처리 수준에 따른 종류

시뮬레이션을 낮은 수준(level)에서 부터 대략적으로 구분하면 다음과 같다.

- 회로 시뮬레이션 : 필터 또는 OP AMP와 같은 회로의 설계를 위한 아날로그 시뮬레이션
- 스위치 시뮬레이션 : 트랜지스트를 기초로한 시뮬레이션
- 논리 시뮬레이션 : AND, OR 등과 같은 게이트에 기초를 둠
- RTL 및 기능별 시뮬레이션 : 회로의 구성 요소를 하나의 단위로 한 동작적 시뮬레이션

위의 구분에서 수준이 낮을수록 시뮬레이션 결과가 설계에 가깝고 정확하지만 시간이 많이 걸리는 단점이 있는 반면에, 수준이 높을수록 시뮬레이션 시간이 적게 걸리지만 정확도가 떨어진다. 덧붙여서 위 수준을 복합적으로 사용한 혼합 수준(mixed-level)과 아날로그와 디지탈을 함께 고려한 혼합 모드(mixed-mode) 시뮬레이션 등이 있다. 이 중에서 일반적으로 가장 많이 사용되는 것은 본고에서 설명하고자 하는 게이트 수준의 논리 시뮬레이션이다.

2. 논리 시뮬레이션의 특징

논리 시뮬레이션에서 고려되는 일반적인 특징(또는 가정)에 대해서 언급하고자 한다.

- 고정된 게이트 지연 (fixed gate delay) : 시뮬레이션 동안에 각 게이트는 입출력에 상관없이 고정된 지연(delay)를 가지고 있다.
- 비선점(non-preemption) : 한 시뮬레이션 사이클내에서 각 게이트상의 시뮬레이션은 다른시뮬레이션에 의해 영향을 받지 않는다.
- 예견 능력(lookahead capability) : 각 게이트의 지연이 고정되고 입력된 이벤트가 정해지면 출력되는 이벤트 시간 및 값을 미리 예상할 수 있다.
- 미세 그레인 연산(fine-grain operations) : 각 게이트는 아주 간단한 테이블 참조(table lookup) 연산과 지연 합산만을 처리한다.

위의 네 가지는 본 고에서 다루는 시뮬레이션 기법을 구현하기 위한 것이고 경우에 따라서는 비선점 및 예견 능력에 대한 가정을 다소 변경할 수 있다. 예를 들면 VHDL의 inertial delay를 다루기 위한 경우를 들 수 있다.

3. 지연 처리에 따른 종류

지연처리에 따른 논리 시뮬레이션 기법으로서는 크게 compiled-code 시뮬레이션과 event-driven 시뮬레이션으로 나눌 수 있는데, 지금까지는 간편성 때문에 compiled-code 시뮬레이션이 많이 사용되어 왔다. Compiled-code 시뮬레이션이란 NOT, AND, OR, NAND 등과 같은 함수의 계산을 지원할 수 있는 C와 같은 고급 프로그래밍 언어를 사용하는 시뮬레이션으로서 회로는 이러한 함수의 연결로써 표현된다. 오직 조합적(combinational)이고 동기적(synchronous)인 회로의 함수적 시뮬레이션에만 사용한다. 빠르고 간단한 논리 시뮬레이션에 사용되며 논리 게이트의 다중 지연(multi-delay)은 고려하지 않은 zero-delay나 unit-delay 논리 시뮬레이션만이 수행된다.

이러한 다중 지연 처리 문제을 보완하기 위해서 사용되는 기법이 event-driven 시뮬레이션이다. 시뮬레이션 시간과 값의 쌍으로 이루어진 이벤트(event)를 기초로 이루어지는 event-driven 시뮬레이션은 zero-delay, unit-delay는 물론 다중 지연 논리 시뮬레이션을 위해서 사용된다. 그 과정은 이벤트가 있는 게이트만 수행하는 선택적 추적 접근(selective trace approach) 방법을 사용하므로 아주 효과적이다. 그러나 이벤트의 수가 많아짐에 따라서 수행 시간이 엄청나게 걸린다.

본 고에서는 병렬처리 컴퓨터 상에서 event-driven 논리 시뮬레이션에 대해서 설명하고자 한다.

Ⅲ. SIMD 병렬처리 컴퓨터상에서 시뮬레이션

1. 병렬처리 컴퓨터

고속의 처리를 위해 사용되는 병렬처리 컴퓨터란 수십에서 수만개의 처리기로 구성된 컴퓨터를 말한다. 병렬처리 컴퓨터는 병렬성(parallelism)을 기준으로 하며 자료의 병렬성(data parallelism)을 이용한 SIMD(Single Instruction - Multiple Data stream)컴퓨터와 기능의 병렬성(function parallelism)을 이용한 MIMD(Multiple Instruction - Multiple Data stream)으로 나눌 수 있다.

Event-driven 논리 시뮬레이션을 위해서 병렬처리 컴퓨터를 사용하고자 할 때는 다음 몇 가지를 고려하는것이 좋다. 첫째, VLSI 설계에는 수 많은 게이트들이 있으며 이에 비례하여 하나의 처리기에 몇개의 게이트를 어떠한 방법으로 분할하여 할당해야 하는지를 결정해야 하는 회로의 분할(circuit partitioning) 및 부하 균형 (load balancing) 문제가 발생한다. 이를 위해서는 보다 많은 처리기를 가진 병렬처리 컴퓨터를 사용하면 이러한 문제가 다소 해결할 수 있다. 둘째, 앞에서 언급한 바와 같이 각 게이트는 미세 그레인 연산을 하므로 처리기가 아주 많은 경우에 각처리의 기능이 다소 간단하고 값싼 것을 사용해도 된다. 세째, 회로상의 각 게이트 및 블락은 병렬로 처리되지만 서로간의 처리에 따르는 동기화 문제(synchronization) 를 처리할 수 있어야 한다.

병렬처리 컴퓨터를 이론적으로 고려하면 대단히 높은 속도를 낼 수 있을 것 같으나 실제적으로는 그렇지 못하다. 그 이유는 어떻게 많은 처리기를 효율적으로 서로 통신하면서 사용하느냐에 따라서 그 수행 능력이 달라지기 때문이다. 지금까지 위에서 열거한 사항들을 고려하여 시뮬레이션을 위해 적당한 병렬처리 컴퓨터를 선정함에 있어서 먼저 SIMD와 MIMD를 비교하여 보도록 하자. 현시점에서 볼때 일

반적으로 SIMD 컴퓨터는 MIMD 보다 많은 처리기를 가지며 각 처리기의 기능도 아주 간단하여 값도 싸다. 또한 MIMD에서는 각 처리기가 독립적으로 수행하므로 필요시에 프로그램상에서 동기화를 위한 조치를 별도로 취해야하지만 SIMD 컴퓨터에서 모든 처리기는 하나의 제어에 의해서 일률적으로 하나의 명령어에 의해서 수행되므로 동기화가 쉬우며 프로그래밍 또한 비교적 쉽다. 즉 MIMD 컴퓨터에서 처리기 서로간은 비동기적(asynchronous)으로 처리되지만 SIMD 컴퓨터에서는 동기적(sybchronous) 처리를 기초로 하고 있다. 그러나 현재 이용 가능한 MIMD컴퓨터가 SIMD 컴퓨터보다 훨씬 큰 국부 기억장치 용량을 가진다는 장점이 있다.

따라서 본 고에서는 SIMD 컴퓨터중에서도 대단위 병렬처리기(massively parallel processors)를 가진 Thinking Machine사의 CM-2(Connection Machine)와 MasPar사의 MP-1를 고려하였다. CM-2는 65,536개까지의 처리기를 가질 수 있으며 각 처리기(또는 PE : processing element)는 1-bit 처리와 부동 소숫점 처리가 가능한 혼합 형태를 이룬다. 반면에 MP-1은 16,384개까지의 처리기를 가질 수 있으며 각 처리기는 4-bit RISC 처리기를 사용하고 있다.

2. SIMD 병렬처리 시뮬레이션 기법

병렬처리 SIMD 컴퓨터상에서 이용할 수 있는 event-driven 시뮬레이션 기법에 대해서 설명하기 전에 몇가지를 먼저 정의하여야 한다. 여기서 말하는 이벤트(event)란 이진값(0 또는 1)과 시뮬레이션 시간의 쌍으로 이루어진 것을 말하는데 디지탈 회로에서 발생하는 모든 파형은 이벤트로 표시될 수 있다. 즉 시뮬레이션을 위해서 주어지는 입력 파형이나 시뮬레이션 결과를 나타내는 출력 파형도 이벤트들의 모임으로 나타낼 수 있다. 시뮬레이션에서 각 게이트들 사이에는 이벤트를 주고 받으며 이벤트를 통해서 시뮬레이션을 수행한다. Event-driven 시뮬레이션에서 각 게이트는 바로 이전에 출력한 이벤트와 같은 값을 가지는 이벤트는 생성하지 않으므로 효율적인 시뮬레이션을 할 수 있다.

각 게이트는 그 게이트에 들어온 입력 이벤트를 처리하기 이전까지 일시적으로 보유하는 이벤트 큐(Event Queue)를 가진다. 각 게이트는 LVT(Local Virtual Time)를 가지는데 이는 이벤트 큐에 있는 처리되지 않는 이벤트의 시뮬레이션 시간 중에서 가장 작은 것을 말한다. 또한 전체 회로에서 처리되지 않은 이벤트의 가진 가장 작은 시뮬레이션 시간을 GVT(Global Virtual Time)이라고 한다. 즉 GVT는 전체 LVT중의 최소값이라고 할 수 있다. SIMD 컴퓨터에서는 GVT의 계산이 아주 쉬운 장점이 있다.

모든 게이트는 똑 같은 시뮬레이션 과정을 반복적으로 수행한다. 즉 i) 각 게이트에서 가진 가진 이벤트를 이용한 LVT 계산, ii) 이벤트를 생성할 게이트 선정, iii) 새로운 이벤트 생성 및 전달, iv) 전달 받은 이벤트의 저장의 과정을 말한다. 이러한 반복적인 과정을 시뮬레이션 사이클(Simulation Cycle)이라고 부른다.

지금까지 설명한 개념들을 이용하여 SIMD 컴퓨터상에서 병렬처리 event-driven 시뮬레이션의 기법에 대해서 설명하고자 한다. 각 기법은 위의 시뮬레이션 사이클의 네 단계중에서 두 번째 단계인 어떻게 이벤트를 생성할 게이트를 고르느냐에 따라서 구분된다. 그러므로 아래에서는 두번째 단계의 차이점에 대해서만 설명하고자 한다.

1) 동기적(Synchronous) 시뮬레이션

이벤트를 생성하는 모든 게이트는 GVT에 의해서 동기화된다. 즉 위 두번째 단계에서 전체 LVT를 이용하여 GVT를 구한다음 GVT와 같은 LVT를 가진 게이트만 이벤트 생성에 참여한다. 전체 시스템을 통틀어서 가장 적은 LVT를 가진 게이트들만 이벤트를 생성하게 함으로 이벤트를 보내는데 있어서 선행 배반(precedence violation)을 하지 않도록 한다. 여기서 말하는 선행 배반이란 게이트에서 계속해서 내보내는 이벤트들의 시뮬레이션 시간은 같거나 증가하여야 함에도 불구하고 감소하는 경우를 말한다.

2) 보수적(Conservative) 시뮬레이션

Chandy-Misra 기법[1,2]이 가장 대표적인 보수적 시뮬레이션 기법이므로 소개하기로 한다. 동기적 시뮬레이션 보다는 훨씬 효과적인 방법으로서 이벤트를 생성하는 게이트를 고르기 위해서 각 게이트에게는 입력 대기 규칙(input waiting rule)을 적용하여 이율 배반을 방지한다. 입력 대기 규칙이란 게이트의 각 입력 포트마다 그 포트로 가장 최근에 도착한 이벤트의 시뮬레이션 시간을 가지도록 한다. (이 시뮬레이션 시간을 해당 포트의 클락 시간이라고 부른다) 이 경우에 각 게이트에서는 LVT와 같고 그 게이트의

모든 클락 시간의 최소값과 같거나 작은 시뮬레이션 시간을 가진 이벤트를 보유한 게이트를 선정한다. 이러한 이벤트는 수행되어 다음 게이트에 전달되어도 선행 배반을 하지 않는다.

3) 낙관적(Optimistic) 시뮬레이션

낙관적인 시뮬레이션의 대표적인 예가 Time-Warp[7,8] 이므로 이를 설명하기로 한다. 위의 두 가지 기법과는 다른 방법으로서 새로운 이벤트를 보내기 전에 선행 배반여부를 검사하는 것이 아니라, 들어온 모든 이벤트는 수행을 하여 새로운 이벤트를 만들어 보낸뒤에 선행 배반임이 밝혀지면 선행 배반이 일어난 지점부터 수행을 다시 시작한다 (이를 rollback이라고 부른다.) 대신에 rollback이 언제 일어날지 모르기 때문에 각 게이트에서는 GVT부터 현재의 수행시점까지 실행을 기록해 둔다. 이 경우에 잘못 보내진 이벤트에 의해서 파생된 영향(side-effect)을 취소(cancellation)해야하는데 이를 위한 두 가지의 방법이 있다. 첫째는 rollback에서부터 수행해 올라가면서 그 파생된 영향을 취소하는 방법(lazy cancellation)과 발견과 동시에 그 이전에 파생된 영향을 모두 취소하고 rollback 시점부터 시작하는 방법(aggressive cancellation)이다.

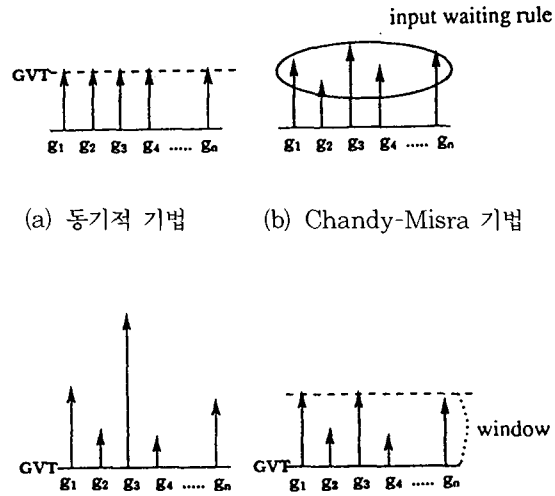
Time-Warp에서는 rollback을 위한 overhead는 있지만 각 게이트가 가지고 있는 이벤트를 수행하여 보내면 rollback이 일어나지 않는한 그만큼 이득인 것이다. 어차피 SIMD에서는 게이트 수행할 것이 없으면 쉬고 있어야 하므로 이때에 rollback 과정을 하면된다.

4) 이동 시간 윈도우(Moving Time Window)

위에서 언급한 Time-Warp에서 GVT보다 시뮬레이션 시간이 많이 앞서가는 게이트가 있으면 이로 말미암아 rollback 발생할 가능성이 높다. 이동 시간 윈도우는 가장 작은 LVT를 가진 게이트와 큰 LVT를 가진 게이트 사이의 LVT 격차가 어느 정해진 시간 간격(Time Window)를 넘지 않도록 하는 기법이다. 이때 이 간격은 시뮬레이션이 계속됨에 따라 크기는 일정하지만 앞뒤의 시간은 계속 변하게된다.

위에서 언급한 네 가지 기법을 나타내면 그림 1과 같다.

그외의 시뮬레이션 기법으로서 token-driven 시뮬레이션 기법[5,6]이 있다. 이는 일종의 compiled-code 시뮬레이션을 병렬처리 컴퓨터상에서 수행을 위해 병렬화(parallelization)를 시킨 기법으로서 지연



(a) 동기적 기법 (b) Chandy-Misra 기법

(c) Time Wrap 기법 (d) 이동시간 윈도우

(gi는 게이트를 나타냄)

그림 1. 시뮬레이션 기법

(즉 시뮬레이션 시간) 이 필요없으므로 시뮬레이션 값만이 이용될 뿐이다. 숫자를 할당받은 토큰은 게이트로부터 여러 개의 값을 운반한다. 토큰에 숫자를 할당한 이유는 시뮬레이션을 위해서 값들을 동기화하기 위한 것이다.

3. 구현을 위한 고려 사항

위에서 언급한 기법들은 원래 MIMD나 Network 상에서 논리 시뮬레이션이 아닌 다른 목적을 위해서 만들어진 것으로서 SIMD 컴퓨터상에 구현하기 위해서 다소 변형된다. 그러므로 위 기법들과 사용하고자 하는 SIMD 컴퓨터의 구조를 완벽하게 이해를 하면 아주 좋은 결과를 얻을 수가 있다. 다음 몇 가지 사항을 고려해 보자. 첫째 SIMD 컴퓨터의 메모리 크기는 일반적으로 적으므로(예를 들면 CM-2, MP-1의 처리기의 국부 기억장치 용량은 각각 8K, 16K 바이트이다) 이벤트 큐를 위해서 사용시 효율적으로 사용해야 한다. 둘째 SIMD에서는 모든 명령어가 동기화되므로 입출력 및 연산을 위해서 table lookup 등과 같이 가능한 간단한 연산을 사용해야 한다.[3] 세째 SIMD 컴퓨터에서 처리기의 수는 제한되지만 회로에 따라서는 게이트의 수를 예측할 수 없으므로 아주 큰 회로에 대해서는 하나의 처리기에 여러개의

게이트가 할당될 경우를 대비해서 컴퓨터의 가상 처리기(Virtual Processor)의 개념이 가능해야한다. 이 경우에는 수행 능력이 기대 이하로 떨어 질 수 있음을 알아야 한다.

Ⅳ. 수행능력의 평가 방법

제시된 시뮬레이션 기법의 테스트를 위해 벤치마크 회로로서 ISCAS89 회로 및 32 bit array multiplier를 사용하였으며 SIMD 컴퓨터로서 32,768개의 처리기를 가진 CM-2와 8,192개의 처리기를 가진 MP-1을 사용하였다.

위의 기법들에 대한 수행능력의 평가는 여러 가지로 할 수 있다. 첫째, 회로의 검증을 위한 시뮬레이션 사이클의 수(number of simulation cycles)를 들 수 있다. 동기적 시뮬레이션 기법이 가장 많은 시뮬레이션 사이클 수를 요구하며 낙관적 시뮬레이션 기법이 가장 진보적이므로 적은 수를 요구한다. 둘째, 한 시뮬레이션 사이클에서 평균 몇개의 게이트가 이벤트를 처리하는지를 나타내는 병렬성(parallelism)을 말할 수 있다. 세째, 가장 중요한 기준으로써 입력 벡터를 설계에 주었을때 시뮬레이션을 위해서 필요한 전체 시간을 들 수 있다. 동기적 기법은 병렬성이 낮으므로 느리며, 낙관적 기법은 rollback 처리등으로 병렬성은 높지만 한사이클내에서 수행해야 하는 시간이 많기 때문에 보수적 방법보다 느리다.

시뮬레이션 사이클의 수, 요구되는 전체 시뮬레이션 시간은 적을 수록 좋으며, 병렬성은 클수록 좋다. CM-2와 MP-1에서의 시뮬레이션을 위한 속도 차이를 계산해 본 결과 동기적, 보수적, 낙관적 기법이 각각 1.81, 2.01, 2.52배정도 CM-2에서보다 MP-1에서 빨리 수행할 수 있었다.

Ⅴ. 결론

하나의 처리기를 사용한 처리 속도는 거의 물리적 한계에 도달하고 있으나, 아주 복잡하고 큰 VLSI 설계의 시뮬레이션 등을 위해서 보다 빠른 컴퓨터 이용에 대한 요구는 갈수록 크지고 있으므로 이를 해결하는 방법으로서 병렬 처리 컴퓨터의 이용은 필수적이라고 할 수 있다.

병렬처리 컴퓨터중에서 MIMD에서 보다 SIMD에서 구현할 경우에 쉽게 GVT를 계산할 수 있고 프로그래밍이 쉬우며 많은 수의 처리기를 사용함으로서 이점을 얻을 수 있다. 그러나 SIMD 컴퓨터에서 단점은 적은 크기의 기억장치 용량과 입력 및 출력 포트의 결합에 비례하는 메시지 전달 순서의 필요, 처리기간의 느린 통신 등이 있다.

향후 연구 과제로서 Thinking Machine사가 최근에 발표한 CM-5를 사용한 위 시뮬레이션 기법의 구현을 고려해 볼만하다. CM-5는 적은 국부 기억장치의 용량과 처리기간의 느린 통신 등과 같은 CM-2의 단점을 대폭 보완하여 내놓은 차세대형 병렬처리 컴퓨터로 알고 있다. 병렬처리 컴퓨터를 이용한 시뮬레이션 기법을 VHDL을 위해 보다 빠른 시뮬레이터로 사용할 수 있도록 하는 것도 바람직 하다고 본다. 시뮬레이션 자체의 단점이라고 할 수 있는 모든 입력 벡터에 대해서 다 고려할 수 없기에 새로 나온 정형 검증 기법[9]에 대해서 병렬 컴퓨터에서의 구현을 고려해 볼만하다.

參考文獻

[1] K. M. Chandy and J. Misra, *Parallel Program Design, A Foundation*, Adison-Wesley, 1988.

[2] K. M. Chandy and J. Misra, "Asynchronous distributed simulation via a sequence of parallel computations," *Communications of the ACM*, 24(11), pp. 198-206, April 1981.

[3] M. J. Chung and Y. Chung, "Data parallel simulation using Time Warp on the Connection Machine," *Proceedings of the 26th Design Automation Conference*, pp. 98-103, June 1989.

[4] M. J. Chung and Y. Chung, "Efficient parallel logic simulation techniques for the Connection Machines," *Proceedings of the Supercomputing'90*, pp. 606-614,

November 1990.

[5] M. J. Chung and Y. Chung, "A distributed token-driven technique for parallel zero-delay logic simulation on massively parallel machines," *Proceedings of 1991 International Conference on Parallel Processing*, pp. 391-394, August 1991.

[6] J. R. Engelsma, M. J. Chung, and Y. Chung, "Distributed token-driven logic simulation on a shared-memory multiprocessor," *Proceedings of the 6th Workshop on Parallel and Distributed Simulation*, pp. 197-198, January 1992.

[7] D. R. Jefferson, "Virtual Time," ACM Trans. *Programming Languages and Systems*, 7(3):404-425, July 1985.

[8] D. R. Jefferson, "Fast concurrent simulation using the Time Warp mechanism," *Proceedings of the 1985 Multiconference on Distributed Simulation*", pages 63-69. SCS, January 1985.

[9] P. Camuarati and P. Prinetto, "Formal verification of Hardware Correctness: Introduction and Survey of Current Research, "*Computer*, pp. 8-19, July 1988.

筆 者 紹 介

鄭 然 模

1957年 10月 7日生

1980年 2月 경북대학교 졸업(이학사)

1982年 2月 한국과학기술원 졸업(공학사)

1992年 2月 미국 미시간주립대학교 졸업(공학박사)

1982年 2月 ~ 1987年 3月 경제기획원 전산처리관

1992年 3月 ~ 현재 경희대학교 전자공학과 조교수

주관심 분야 : VLSI 설계 및 CAD, 병렬 처리, 컴퓨터 구조, 소프트웨어 공학.