집적검증 기법을 채용한 하드웨어/소프트웨어 동시검증
(Hardware/Software Co-verification with Integrated Verification)

이 영 수 † 양 세 양 ††
(Youngsoo Lee) (Seiyang Yang)

요 약 SOC(System On a Chip)에 대한 설계에서 설계 생산성을 향상시키기 위해서 가장 시급히 해결해야 할 과제가 하드웨어뿐만 아니라 소프트웨어까지도 함께 동시검증(co-verification)하여야 함으로서 설계검증에 과도하게 투입되는 비용과 시간을 줄이는 것이다. 본 논문에서는 이러한 설계검증 생산성을 효과적으로 높이기 위한 방법으로 HW/SW 동시검증을 수행할 수 있는 대표적인 두 방법들인 동시-시뮬레이션(co-simula tion)과 동시-에뮬레이션(co-emulation)을 강하게 결합한 새로운 검증 방법인 집적 동시검증(integrated co-verification) 방법을 제안하였다. 또한, 상용화된 동시검증 툴인 Seamless CVE와 물리적 프로토타이핑 보드를 함께 사용하여 구성한 ARM/AMBA 플랫폼 기반의 집적 동시검증 환경을 직접 구성하고, 이를 이용하여 제안된 검증기법의 유용성을 실험적으로 확인하였다.

키워드 : 시스템 설계, 시스템 검증, 하드웨어/소프트웨어 동시설계, 하드웨어/소프트웨어 동시검증

Abstract In SOC(System On a Chip) designs, reducing time and cost for design verification is the most critical to improve the design productivity. This is mainly because the designs require co-verifying HW together with SW, which results in the increase of verification complexity drastically. In this paper, to cope with the verification crisis in SOC designs, we propose a new verification methodology, so called integrated co-verification, which tightly combine both co-simulation and co-emulation in unified and seamless way. We have applied our integrated co-verification to ARM/AMBA platform-based co-verification environment with a commercial co-verification tool, Seamless CVE, and a physical prototyping board. The experiment has shown clear advantage of the proposed technique over conventional ones.

Key words : system design, system verification, HW/SW co-design, HW/SW co-verification

1. 서 론

최근 반도체 제조 및 기술은 무어의 법칙(Moores Law)에 따라 급속한 발전을 이루고 있고, 시장에서는 보다 작고, 빠른, 보다 다양한 기능을 가진 제품을 요구하고 있다. 이러한 기술적 진보와 시장에서의 다양한 요구는 고속의 대용량 회로 설계를 점차 일반화 시켜가고 있는 반면, 대용량 고속 회로의 설계 시작부터 완료시점까지 소요되는 개발 기간의 기대치는 점차 줄어들어 설계자들의 부담을 가중시키고 있다. 이러한 상황은 디지털 로직, 메모리, 디지털/아날로그 혼용 로직 및 사용자 로직 등이 하나의 실리콘 내에 집적될 수 있는 SOC (System On a Chip) 기술이 보편화 되면서 더욱 심화되고 있다.

특히 SOC 기술에 따른 시스템 설계에는 HW와 SW를 동시에 설계하는 HW/SW 동시 설계(co-design) 방식이 주로 이용되고 있는데, 이 설계방식은 설계생성 단계에서의 설계복잡도를 크게 증가시켜 설계생성 후에 검증에 필요한 시간과 비용을 크게 증가 시키는 결과를 초래하고 있다. 이와 같은 HW/SW가 동시에 존재하는 시스템의 설계 과정은 크게 시스템 설계생성 단계와 시스템 검증 단계로 나눌 수 있다. 시스템의 복잡도가 그다지

· 본 연구는 부분적으로 반도체설계교육센터(IDEC)의 설계 툴 지원과 부산대학교 컴퓨터 정보통신연구소의 지능형 단말기 연구 센터(정보통신부 지원 IT 분야 기술인력 양성센터)의 연구비 지원에 의하여 이루어졌습니다.

† 학생회원 : 부산대학교 컴퓨터공학과
ysoolee@hyowon.pusan.ac.kr
†† 정 회 원 : 부산대학교 컴퓨터공학과 교수
syyang@hyowon.pusan.ac.kr

논문접수 : 2001년 12월 14일
심사완료 : 2002년 3월 7일

크지 않았던 과거에는 설계 전체 과정에 투입되는 비용 중 설계생성 단계에 투입되는 비용이 설계검증 단계에 투입되는 것보다 큰 비중을 차지 하였다. 그러나 IP(Intellectual Property)의 사용과 논리합성(logic synthesis)이나 아키텍춰 합성(architecture synthesis)과 같은 설계생성 과정을 매우 신속하게 수행할 수 있는 다양한 방법들이 사용되면 될수록 생성된 설계에 대한 검증을 수행하는데 투입되어지는 시간과 비용은 더욱 더 증가 되는 결과를 초래하여, 현재와 같은 시스템 설계의 복잡도에서 전체 시스템 설계의 전과정에서 시스템 설계 검증을 위하여 전체 시스템의 설계 시간과 비용의 70%를 설계 검증에 투입해야 하는 심각한 설계 검증 위기 상황을 초래하고 있다. 더욱이, 설계검증 생산성을 높일 수 있는 효과적인 방법론이 제시되지 않는다면 가까운 미래에 시스템 수준 검증을 위하여 투입되어야 하는 비용과 시간은 더욱 크게 증가할 것으로 예상되고 있다[1].

시스템 설계 검증에 현재까지 가장 보편적으로 사용되는 방법은 동시-시뮬레이션(co-simulation)이다[2, 3]. 동시-시뮬레이션은 사용의 편이성, 강력한 디버깅 지원, 유연성, 매우 빠른 컴파일 속도 등의 장점들이 있지만, 최근의 수백만 게이트급 이상의 시스템수준 설계 검증에서는 매우 낮은 시뮬레이션 성능과 인서킷(in- circuit) 능력의 결여 등의 문제점으로 동시-시뮬레이션만을 이용하여 시스템에 대한 SOC 설계검증을 성공적으로 수행하는 것은 극히 현실적이지 못한 것으로 인식되고 있다[4, 5, 6, 7]. 반면, 동시-에뮬레이션(co-emulation)은 고속의 검증 속도와 인서킷 능력의 보유 등의 장점들이 있으나, 디버깅의 어려움, 사용의 어려움, 매우 긴 컴파일 시간, 높은 가격과 유지비용 등의 문제점들을 가지고 있다[8, 9]. 본 논문에서는 이와 같은 시스템 수준 설계 검증의 대표적인 두 가지 방법들인 동시-시뮬레이션과 동시-에뮬레이션을 강하게 결합시킨 새로운 시스템 검증 방법인 집적 동시검증(Integrated Co-verification) 방법을 제시한다. 이 새로운 검증방법은 동시시뮬레이션과 동시-에뮬레이션의 장점만을 모두 취합할 수 있음으로서, 고속의 검증실행 속도, 편이성, 강력한 디버깅 지원, 인서킷 능력, 유연성 등의 장점들을 매우 낮은 비용으로도 얻을 수 있다. 따라서 이와 같은 집적 검증 방법의 이용은 시스템 수준 검증 생산성을 크게 높일 강력한 해결책 중의 하나가 될 수 있다.

앞으로 본 논문은 다음과 같이 구성된다. 우선 2장에서는 집적 동시검증 기법에 대하여 설명하고 이 기법의 장점들을 언급하며, 3장에서는 이와 같은 집적 검증기법을 최근 내장형 시스템(embedded system)에 제일 많이 사용되는 ARM사의 ARM/AMBA 플랫폼 기반의 SOC를 설계하는 과정에 적용한 집적 동시검증 기법을 제안하며, 4장에서는 이를 이용한 실험결과를 분석하고, 마지막으로 결론을 5장에서 언급한다.

2. 집적 동시검증 기법

집적 검증(Integrated Verification) 기법은 전통적인 검증 방법인 소프트웨어 기반의 시뮬레이션과 하드웨어 기반의 에뮬레이션을 강하게 결합시켜서 검증 과정에서 시분할(time division)적으로 시뮬레이션과 에뮬레이션을 1회 이상 번갈아 가면서 수행하는 검증 방법이다. 이를 위해서는 시뮬레이션에 이어서 순간적 전환(instant switching)을 통하여 에뮬레이션이 수행될 수 있도록 하거나, 에뮬레이션에 이어서 순간적 전환을 통하여 시뮬레이션이 수행될 수 있도록 하는 것이 필요한데(이를 순간적 방식 전환이라 칭하기로 함), 이와 같은 순간적 방식 전환은 검증 대상에 대한 상태 정보(status information)(검증 대상에 존재하는 모든 레지스터나 메모리의 값. 레지스터는 플립플롭이나 래치들로 구성됨)가 전달되면 가능해 진다[10].

본 논문에서 제시할 집적 동시검증(Integrated Co-verification) 기법은 임의의 프로세서를 명령어 수준에서 소프트웨어적으로 모델링한 ISS(Instruction Set Simulator)와 임의의 HDL 시뮬레이터로 구성되는 임의의 동시-시뮬레이터(co-simulator)와 임의의 상기 임의의 프로세서 칩과 1이상의 FPGA로 구성되는 동시-에뮬레이터(co-emulator)를 연결하는 HW/SW 통합검증 각 부분의 HW/SW 인터페이스를 사용함으로써 동시-시뮬레이션과 동시-에뮬레이션이 하나의 틀(frame-work)에서 동작할 수 있는 환경을 만들어 주게 된다. 이와 같은 인터페이스는 동시-에뮬레이터와 동시-시뮬레이터에서 수행되는 설계검증 대상 모델들간의 디버깅 정보들의 교환을 가능하게 하고 서로 다른 시스템 수준 설계 검증 방식 전환(동시-시뮬레이션에서 동시-에뮬레이션으로 혹은 동시-에뮬레이션에서 동시-시뮬레이션으로의 전환)을 횟수에 상관없이 순간적으로 가능하게 한다. 이와 같은 자유로운 검증 방식들 간의 전환을 통하여 집적 검증방법은 다음과 같은 장점들을 제공함으로서, 효과적이며 신속한 검증을 가능하게 한다.

2.1 고속의 동시검증 실행 속도

시스템 검증 수행의 특정 기간동안에 매우 빠른 동시검증 실행속도를 얻고자 할 경우에는 동시-에뮬레이터 상에 하드웨어적으로 모델링된 검증 대상을 수행시킴으로써 손쉽게 이를 달성할 수 있다. 만일 이에 앞서서 동

시시뮬레이션이 수행되고 있었다면, 인터페이스를 통해서 동시-시뮬레이터의 최종 수행 정보를 동시-에뮬레이터로 순간적으로 이동시킴으로써 동시-시뮬레이터의 낮은 검증속도를 해결하면서 동시-에뮬레이터로써 동시-시뮬레이터에서 수행 중이던 검증을 연속적으로 빠르게 수행하는 것이 가능해진다.

2.2 동시디버깅 능력의 향상

디버깅 과정에서 검증대상에 존재하는 많은 HW 상의 신호선들 값이나 SW 상의 변수들의 값과 프로세서의 레지스터나 메모리의 내용들을 일정기간 탐침(probing)하고자 하는 경우에는 동시-시뮬레이션을 통하여 수행하는 것이 편리하며, 이 동시-시뮬레이터를 이용해서 검증 과정에서 HW 상의 어떠한 신호선들이나 SW 상의 변수들의 값과 프로세서의 레지스터나 메모리의 내용들에 대한 신속한 탐침이 가능하다(이를 100% 가시도(visibility) 확보라고 이야기할 수 있음). 만일 현재의 시스템 검증 수행이 동시-에뮬레이터에서 이루어지고 있다면 인터페이스를 통하여 동시-에뮬레이터의 최종 수행정보를 동시-시뮬레이터로 순간적으로 이동시킨 후 탐침을 원하는 이들 신호선들이나 변수들, 레지스터나 메모리에 대한 탐침을 통하여 동시디버깅(co-debugging)을 수행 하는 것이 가능하다.

2.3 사용의 편이성

제시된 집적 동시검증 기법은 시스템 설계 엔지니어나 검증엔지니어의 입장에서 현재 사용 중인 검증 방법을 그대로 사용할 수 있도록 하면서 현재의 검증 방법에 의해 제기되고 있는 문제점들을 해결할 수 있도록 한다. 따라서 시스템 수준 검증을 수행하는 설계 엔지니어나 검증 엔지니어는 검증 환경의 변화를 최소화하면서 검증 생산성을 높일 수 있게 된다.

2.4 인서킷 능력

제시된 집적 동시검증 기법은 동시-에뮬레이션과 같은 하드웨어 기반의 검증 시스템을 이용 할 수 있으므로 자연스럽게 인서킷 능력을 가지게 된다. 이와 같은 인서킷 능력을 활용하면 타켓 환경으로부터 입출력을 직접 공급받고 공급해줄 수 있으므로 실제 타켓 환경에서의 실동작(real operation)의 검증까지 가능해진다.

2.5 낮은 검증비용

제시된 집적 동시검증 기법을 채용하기위해 설계 엔지니어들이 현재 사용하고 있는 검증환경 및 검증 시스템을 모두 바꿔야 하는 것이 아니며, 현재의 검증환경은 그대로 유지 하면서 사용중인 검증시스템에 새로운 검증 시스템을 통합하면 되므로 이와 같은 집적 검증방법 환경을 매우 낮은 비용으로 구성할 수 있다.

3. ARM/AMBA 플랫폼 기반 집적 동시검증 기법의 구현

본 장에서는 ARM/AMBA 플랫폼 기반의 집적 동시검증 기법에 대한 구체적인 구현을 설명한다. 집적 동시검증 기법에서 가장 중요한 요소인 동시-시뮬레이션과 동시-에뮬레이션 결합을 위해서는 다음과 같은 컴퍼넌트들을 사용하였다.

- . Seamless CVE [11]
- . ADSv1.1 [12]
- . ARM Multi-ICE [13]
- . ARM/FPGA 프로토타이핑 보드 [14]
- . MagicDebugger [15]

그림 1 ARM/FPGA 프로토타이핑 보드

Seamless CVE는 동시-시뮬레이션을 수행하게 되며, ARM/ FPGA 프로토타이핑 보드(그림 1)를 이용해서 동시-에뮬레이션을 수행하게 된다. 또한 Magic Debugger는 검증 브릿지(verification bridge)로서 동시-에뮬레이터에 존재하는 FPGA에 구현된 사용자 HW 부분과 동시-시뮬레이터에서의 HDL 시뮬레이터에서 소프트웨어적으로 수행되는 사용자 HW의 모델 간에 상태정보를 순간적으로 교환시키는데 이용되어진다. ARM/AMBA 플랫폼 기반의 집적검증을 하기위해 또 하나 추가되어야 할 것이 앞에서 언급한 동시-시뮬레이터와 동시-에뮬레이터가 하나의 틀에서 동작할 수 있도록 하는 동시-시뮬레이션/동시-에뮬레이션 인터페이스 부분이다. 이 인터페이스는 소프트웨어와 하드웨어의 각각에 대해서 따로 존재하고 기본적으로 Solaris OS 환경을 기반으로 하는 Seamless CVE와 Windows 기반의 동시-에뮬레이터를 연결할 수 있는 다중 서버/다중 클라이언트의 통신구조를 가지고 있다. 집적 동시검증 기법의 SW 인터페이스인 SRDI(Software Remote Debugger Interface)는 Seamless CVE의 SW 디버거인 X-ray 디버거와 Multi-ICE를 타겟환경으로 한 AXD

(ARM Extended Debugger)를 연결시키는 역할을 한다. SRDI는 동시-에뮬레이션에서 동시-시뮬레이션으로 혹은 동시-시뮬레이션에서 동시-에뮬레이션으로의 전환을 가능하게 하고, 서로 다른 검증단계로 전환할 때마다 현재 검증방법의 최종수행정보(레지스터값, 메모리값, 변수값 등)와 디버깅 정보(Breakpoint, Watchpoint 등)를 전달한다(그림 2).

동시-에뮬레이션에서 동시-시뮬레이션으로 전환할 경우 윈도우기반의 SRDI 클라이언트는 AXD에서 레지스터들의 값, 변수들의 값, breakpoint 정보, watch point 정보, 메모리 값 등을 Solaris OS 기반의 SRDI 서버로 전송하고, 그 전송된 값을 이용해 Seamless CVE의 X-ray Debugger를 동시-에뮬레이션의 마지막 수행단계와 동일한 상태(동시-에뮬레이션을 통해 변화된 레지스터나 변수, 메모리값, breakpoint, watch point 정보를 동시-시뮬레이터가 가지도록 함)로 만든 다음 동시-시뮬레이션을 수행한다.

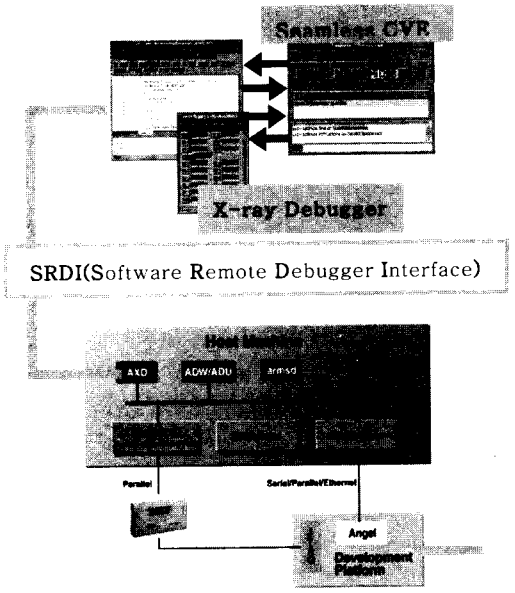

그림 2 SRDI(Software Remote Debugger Interface)

이 방법은 동시-시뮬레이션에서 동시-에뮬레이션으로 전환할 때도 마찬가지이다. Solaris 기반의 SRDI 클라이언트는 Seamless CVE의 X-ray Debugger의 최종수행정보와 디버깅 정보를 Windows 기반의 SRDI 서버로 전송하고 서버는 전송된 정보를 이용해 AXD를 동시-시뮬레이션의 마지막 수행단계와 같은 상태로 만들어 줄 명령어 파일을 생성하고, AXD에서 이 명령어 파일을 수행함으로써 동시-시뮬레이션을 통해 변화된 값들을 동시-에뮬레이터에서 가지게 하고 연속해서 동시-에뮬레이션을 수행한다. 집적 검증 기법의 HW 인터페이스인 HRDI는 Seamless CVE의 HDL 시뮬레이터인 ModelSim과 동시-에뮬레이터의 HW 부분을 결합시켜 동시-시뮬레이션과 동시-에뮬레이션간의 전환을 가능하게 한다(그림 3).

HRDI의 기본적인 기능은 SRDI와 유사하나 동시-에뮬레이션에서 동시-시뮬레이션으로 전환할 경우 동시-에뮬레이션의 사용자 HW 부분의 최종 상태정보를 얻기 위하여, 또한 동시-시뮬레이션에서 동시-에뮬레이션으로 전환할 경우 동시-에뮬레이션의 사용자 HW의 상태정보를 시뮬레이션에서의 상태정보와 같게 만들기 위하여서 MagicDebugger[10]를 사용한다. 이 상태정보는 Windows 기반의 HRDI 클라이언트를 통해 Solaris 기반의 HRDI 서버로 전송되고, HRDI 서버는 전송된 디버깅 정보를 가지고, 동시-시뮬레이션 수행에 필요한 함수나 명령어를 생성한 후 ModelSim의 FLI (Foreign Language Interface)를 이용해서 동시-시뮬레이션을 수행하도록 한다.

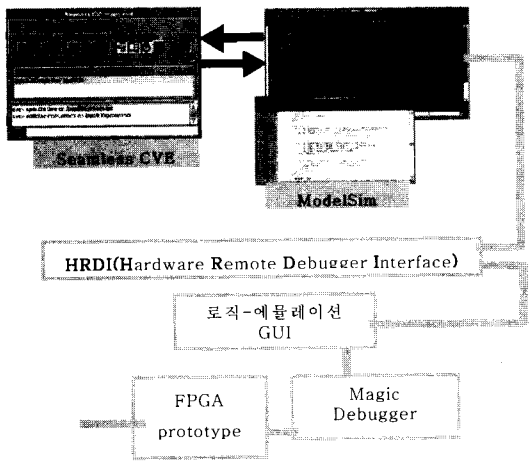

그림 3 HRDI(Hardware Remote Debugger Interface)

동시-시뮬레이션에서 동시-에뮬레이션으로 전환할 경우에는 Solaris 기반의 HRDI 클라이언트는 Seamless CVE의 ModelSim의 FLI를 통해서 최종 수행정보를 얻어내고 이를 Windows 기반의 HRDI 서버로 전송한다. 이 정보는 Windows 기반의 HRDI 서버를 통해 Magic

Debugger로 전송되어져서 동시-에뮬레이터의 HW 부분의 상태 값을 변화시키는데 사용되어진다. ARM/AMBA 플랫폼 기반의 집적 동시검증기법에서 중요한 역할을 하는 부분이 바로 동시-시뮬레이션과 동시-에뮬레이션의 인터페이스인 SRDI와 HRDI이다. 이 두 인터페이스는 앞에서 살펴본 바와 같이 집적 동시검증 기법에서 동시-시뮬레이션과 동시-에뮬레이션이 하나의 플랫폼에서 동작할 수 있도록 각 검증기법의 SW와 HW 부분을 연결해주는 역할을 함으로써 동시-에뮬레이션과 동시-시뮬레이션이 가지고 있는 장점만을 취해서 보다 강력하고 능률적인 디버깅 환경을 제공해 준다. 특히 사용자가 집적 동시검증 환경에서 직접적으로 접하게 되는 GUI 환경인 동시-시뮬레이션 GUI(Seamless CVE의 GUI)(그림 4)와 동시-에뮬레이션 GUI(그림 5)가 SW와 관련된 윈도우들(소스코드 윈도우, 변수 윈도우, 레지스터 윈도우, 메모리 윈도우)과 HW와 관련된 윈도우(파형보기 윈도우)로써 구성 상에서 거의 동일함으로, 동시-시뮬레이션에서 동시-에뮬레이션으로 또는 동시-에뮬레이션에서 동시-시뮬레이션으로의 전환 시에 사용자에게 일관성 있는 검증 환경을 제공하게 되는 장점이 있다.

4. 실 험

앞에서 구현한 ARM/AMBA 플랫폼 기반의 집적 동시검증 기법의 성능을 실험적으로 검증하기위해서 이미지 프로세싱의 하나인 윤곽선 검출(edge detection)의 설계를 통해 동시검증을 수행하였다. 사용된 검증 환경의 구성은 그림 2 및 그림 3과 같다. 검증에 사용된 윤곽선 검출 예제는200*200의 256칼라의 .BMP 이미지 파일의 윤곽선을 Sobel 윤곽선 검출 알고리즘을 통해 추출해서 윤곽선 정보만을 가진 새로운 이미지 파일을 생성하는 것을 목적으로 한다. 윤곽선 검출을 동시-시뮬레이션, 동시-에뮬레이션에 사용하기 위해 SW와 HW 부분으로 나누면 다음과 같다.

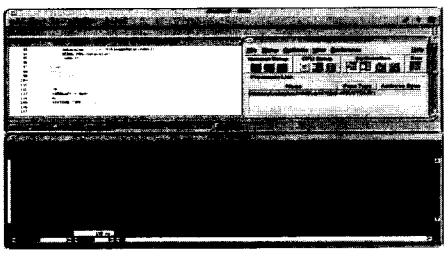

그림 4 Seamless CBE의 디버깅 GUI

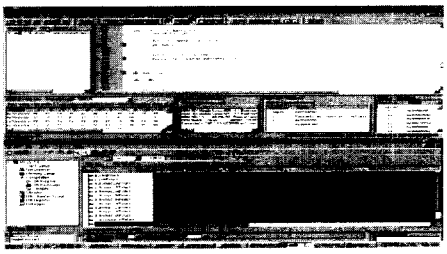

그림 5 동시-에뮬레이션의 디버깅 GUI

* SW 부분 : 윤곽선인지 아닌지 판단되어질 현재 픽셀 값과 윤곽선 검출에 사용될 4개의 픽셀 값을 읽어와서 HW에 전달하는 부분을 담당한다. 그리고 HW에서 윤곽선 검출이 끝난 후, 윤곽선인지 아닌지에 대한 정보를 넘겨주면 그 정보를 보고 윤곽선이면 현재 픽셀 그대로의 값을 윤곽선이 아니면 흰색으로 값을 바꿔서 새로운 이미지를 생성한다.
* HW 부분: SW로부터 받은 픽셀들을 가지고 Sobel 알고리즘을 수행하여 현재 픽셀이 윤곽선인지 아닌지를 가려낸다.

위와 같이 SW와 H/W 부분으로 명확하게 나누어진 윤곽선 검출 로직을 가지고 세가지 검증환경에서 실험을 수행 하였다. 실험 환경에 사용된 컴퓨터는 SUN UltraSparcIIi(프로세서 440MHz, 주기억장치 128MB)와 IBM PC(프로세서 펜티움 650MHz, 주기억장치 384MB)로 각각 Seamless CVE와 ARM AXD와 Multi-ICE를 수행하는데 사용되었다.

세 가지의 검증환경은, 첫째는 동시-시뮬레이터인 Seamless CVE 이용한 동시-시뮬레이션 방식이고 둘째는 ARM/FPGA 프로토타이핑 보드를 이용한 동시-에뮬레이션 방식, 그리고 마지막으로 ARM/AMBA 기반의 집적 동시검증 기법을 사용해서 검증하였다. 실제 윤곽선 검출에서 가장 많은 시간을 차지하는 부분이 각 픽셀에 대한 윤곽선 검증을 하는 부분이므로 ARM/AMBA 플랫폼기반의 집적 검증 실험에서는 4만 픽셀(200*200=40,000)에 대한 윤곽선 검증을 수행하는 동안에는 동시-에뮬레이션을 수행하고, 초기화 단계와 윤곽선 검증이 제대로 수행되는지 확인하기 위해 몇 개의 픽셀에 대한 윤곽선 검출을 하는 과정, 그리고 윤곽선 검출이 끝난 뒤 새롭게 생성된 이미지를 메모리에 저장하는 부분에서는 동시 시뮬레이션을 수행하였다. 그리고 윤곽선 검출 알고리즘에 의도적으로 오류를 넣어서 잘못된 윤곽선 검출 결과가 나타나도록 해서 현재 픽셀의 윤곽선 검출을 수행하는 동시-에뮬레이션 중간

에 동시-시뮬레이션을 수행함으로써 오류를 일으키는 부분을 찾아 낼 수 있도록 하였다. 표 1은 위 세가지 검증방법을 통해 얻은 결과를 보여주고 있다(표 1에 언급된 세가지 방법에서의 유효 클럭속도(두번째 컬럼)는 전체 task를 수행하는데 소요된 수행시간(세번째 컬럼)으로 전체 task를 수행하는데 소요된 총 클럭 사이클수를 나눈 것이다).

표 1 윤곽선 검출 검증 수행 결과

검증방법	유효 클럭속도	수행시간
동시 시뮬레이션	7.6KHz	8시간
동시 에뮬레이션	10MHz	22초
ARM/AMBA 기반의 집적 동시검증 기법	7.3MHz	30초

표 1에서도 알 수 있듯이 윤곽선 검출을 동시-시뮬레이션만으로 검증했을 경우 수행 시간이 상당히 오래 걸린다는 것을 알 수 있다. 만약 사용될 이미지의 크기가 200*200보다 훨씬 더 커지게 된다면 윤곽선 검출이 반영된 최종 이미지를 얻기 위해서는 상당히 오랜 시간동안 기다려야 할 것이다. 사실 실험에 사용된 윤곽선 검출은 작은 시스템에 속하므로 동시-시뮬레이션을 통해서도 결과를 얻을 수 있지만, 실시간운영체제(RTOS)나 통상의 응용 프로그램이 수행되는 일반적인 시스템은 복잡도가 상당히 큼으로 동시-시뮬레이션을 통한 신뢰성 있는 시스템 검증은 거의 불가능하다. 이에 반해 동시-에뮬레이션과 ARM/AMBA 플랫폼 기반의 집적 동시검증 기법은 동일한 클럭 속도로 검증 했을 때 수행시간의 차이가 크게 차이가 나지 않음을 알 수 있다. ARM/AMBA 플랫폼 기반의 집적 동시검증 기법의 수행시간이 약간 더 걸리는 것은 동시-시뮬레이션을 수행하는 과정이 포함되어 있기 때문이다. 표 1을 통해 집적검증 방법이 강력한 디버깅 환경을 제공하면서도 수행속도 또한 다른 순수 에뮬레이션 기반의 검증 방법에 비해 크게 떨어지지 않음을 알 수 있다.

이와 같은 실험을 통하여 쉽게 추론할 수 있는 것은, 본 논문에서 제시된 집적동시검증 방법의 수행속도는 동시 에뮬레이션의 수행속도와 매우 근접될 수 있을 뿐만 아니라 100% 가시도를 통하여 디버깅 용이도는 동시 시뮬레이션과 같게 할 수 있다는 것이다. 이와 같은 능력은 설계되는 시스템의 크기가 커지는 경우와 거의 무관하게 이루어지게 된다. 따라서 설계되는 시스템의 크기가 커지는 경우에도 수행속도는 동시 에뮬레이션에는 매우 근접되지만 동시 시뮬레이션에 비해서는 작은 시스템 설계의 경우보다 그 격차가 더욱 크게 벌어지게 됨을 쉽게 알 수 있으며, 이와는 반대로 디버깅 용이도는 동시 시뮬레이션과 같이 설계되는 시스템의 크기가 커지는 것과는 무관하여 100% 가시도를 유지하는 것이 가능하다는 것이다. 따라서 본 논문에서 제안된 집적동시검증 방법의 유용성은 설계대상이 되는 시스템의 크기가 커지면 커질수록 더욱 높아질 수 있음을 쉽게 알 수 있다.

5. 결 론

본 논문에서는 시스템수준 설계 검증의 대표적인 방법들인 동시-시뮬레이션과 동시-에뮬레이션을 HW/SW 인터페이스를 통해 강하게 결합시킨 집적 동시검증 기법을 제시하였다. 이와 같은 집적 동시검증 기법은 소프트웨어 기반의 검증 방법과 하드웨어 기반의 검증 방법의 장점들만을 모두 취합함으로써 고속의 검증 실행속도, 편이성, 강력한 디버깅 지원, 인서킷 능력, 유연성 등의 장점들을 매우 낮은 비용으로도 모두 가질 수 있다. 특히, 제시된 집적 동시검증 기법을 채용하기 위해 시스템수준 설계 엔지니어들은 현재 사용하고 있는 검증 환경 및 검증 시스템에 새로운 집적 동시검증 시스템을 통합하면 되므로 최소한의 시간과 비용을 투입하면서도 설계검증 생산성을 크게 높일 수 있게 된다. 이로 인하며 집적검증방법은 시스템 설계 검증의 문제를 해결할 수 있는 매우 강력한 방법이 될 수 있다.

참 고 문 헌

[1] Richard Foster, A Design Style to Simplify IP Integration and Verification, White paper, VLSI Technology, Inc. (http://www.vlsi.com), 1999.

[2] R. Klein, Miami, a Hardware/Software Co-verification System, in Proc. 7th IEEE Rapid Systems Prototyping Workshop, pp. 173-177, 1996.

[3] M. Stanbro, Getting to the Bottom of HW/SW Co-verification performance Claims, Computer Design, Vol. 37, No. 12, pp. 65-67, Dec. 1998.

[4] J. Babb, R. Tessier, M. Dahl, S. Hanano, D. Hoki, and A. Agarwal, Logic Emulation with Virtual Wires, *in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, June. 1997.

[5] J. Gateley et al., UltraSPARC-I emulation, in Proc. of Design Automation Conference, 1995.

[6] R. Russell et al., Taking Co-verification To the

Limit ARM Creates a Virtual Prototypes of its Latest Core Running Windows CE, White paper, Mentor Graphics (http://www.mentor.com/), 2001.
[7] S. Chaudhuri et al., Hardware/Software Co-verification of CDMA ASIC Designs, White paper, Mentor Graphics(http://www.mentor.com), 2001.
[8] J. Lach et al., Efficient Error Detection, Localization, and Correction for FPGA-based Debugging, in Proc. of Design Automation Conference, 2000.
[9] M. Kudlugi et al., A Transaction-Based Unified Simulation/Emulation Architecture for Functional Verification, in Proc. of Design Automation Conference, 2001.
[10] S. Yang, Rapid Debugging Method in Rapid Prototyping Apparatus for Embedded Systems, PCT Patent Pending, 2001.
[11] Seamless CVE Datasheet, Metor Graphics (http://www.mentor.com/), 2001.
[12] ARM Developer Suite User Manual, (http://www.arm.com/), 2001.
[13] ARM Multi-ICE User Manual, ARM, (http://www.arm.com/), 2001.
[14] 김동운 외 7인, Customer SoC 및 Application Software 개발을 동시에 진행할 수 있는 Platform 개발, 대한전자공학회 2001 SOC Design Conference 논문집, 2001.
[15] MagicDebugger User Manual, Sevits Technology, Inc. (http://www.sevits.com), 2001.

이 영 수

1989년 부산대학교 컴퓨터공학과 졸업(공학사). 2002년 부산대학교 대학원 컴퓨터공학과 졸업(공학석사). 2002년 1월 (주)삼성전자 재직. 관심분야는 ASIC 설계 및 검증, VLSI-CAD

양 세 양

1981년 고려대학교 전자공학과 졸업(공학사). 1985년 고려대학교 대학원 전자컴퓨터공학과 졸업(공학석사). 1990년 메사추세츠대학교 대학원 전기컴퓨터공학과 졸업(공학박사). 1990년 ~ 1991년 Microelectronics Center of North Carolina 선임연구원. 1991년 ~ 현재 부산대학교 컴퓨터공학과 재직(현 부교수). 관심분야는 하드웨어기반 설계검증, HW/SW 동시검증, 논리합성, 테스팅