論文 96-21-2-12

상위 수준의 마이크로프로세서 호환성 검증 환경 구현

正會員 이 문 기*, 김 영 완*, 서 광 수*, 손 승 일*

A Compatibility Verification Environment for HDL-modeled Microprocessors

Moon Key Lee*, Young Wan Kim*, Kwang Soo Seo*,
Sonh Seung IL* *Regular Members*

要 約

본 논문은 HDL(Hardware Description Language)로 기술된 마이크로프로세서와 기존의 마이크로프로세서와의 호환성을 검증하는 환경에 대하여 기술한다. 호환성 검증은 새로 설계된 마이크로프로세서가 기존의 마이크로프로세서에서 동작하는 OS(Operating System)를 이진 코드의 변형 없이 실행시키는 것을 보여줌으로써 이루어진다. 제안된 검증 환경은 가상 시스템과 그래픽 사용자 인터페이스 모듈로 구성된다. 각각의 모듈은 서버 클라이언트 모델에 바탕을 두고 독립적으로 설계되며, 두 모듈의 정보 교환을 위한 통신부가 존재한다. 본 논문에서는 이러한 검증 환경의 구축 방법에 대하여 기술하고 x86 계열의 마이크로프로세서 호환성 검증 환경을 실험 결과로 제시한다.

ABSTRACT

This paper describes the simulation environment that verifies whether a new microprocessor described with HDL is compatible with an existing microprocessor. The compatibility verification is done by showing that the new microprocessor executes the OS(Operating System) program used in the existing microprocessor without any modification of its binary code. The proposed verification environment consists of a virtual system and a graphic user interface (GUI) module. Each module is independently designed based on server-client model and there exists a communication part for information interchange between the two modules. This paper describes the method of constructing the verification environment and presents the compatibility verification environment of the x86 microprocessor as the simulation result.

*연세대학교 전자공학과
Department of Electronic Engineering Yonsei University
論文番號:95254-0801
接受日字:1995년 8월 1일

I. 서 론

집적회로가 복잡해짐에 따라 집적회로의 설계 기

간과 설계 비용은 크게 증가하였다. 특히 수백만 개의 트랜지스터로 구성되는 대규모의 집적도를 가지고 있는 마이크로프로세서의 설계 검증은 설계 기간 및 설계 비용에서 상당 부분을 차지하므로 매우 중요하다고 할 수 있겠다[1][2].

초기의 집적회로 설계 방식은 회로의 규모가 작아 트랜지스터 수준, 혹은 논리 게이트 수준에서 이루어질 수 있었으며, 논리 시뮬레이션 만으로의 검증이 가능하였다[3][4]. 그러나 백만개 이상의 트랜지스터로 구성되는 마이크로프로세서를 트랜지스터나 논리 게이트 수준으로 구성할 경우, 회로 설계와 회로에 대한 논리 시뮬레이션에 많은 시간이 소요된다. 대규모 집적회로인 마이크로프로세서를 설계할 때는 이러한 점을 보완하기 위하여 [그림 1]의 방법을 사용한다. 먼저, 상위 수준에서 회로의 동작을 기술하는 HDL (Hardware Description Language)을 이용하여 하드웨어를 표현하고, 논리회로 합성(Logic Synthesis)을 통해 논리 게이트 수준의 회로를 얻는 것이다[5].

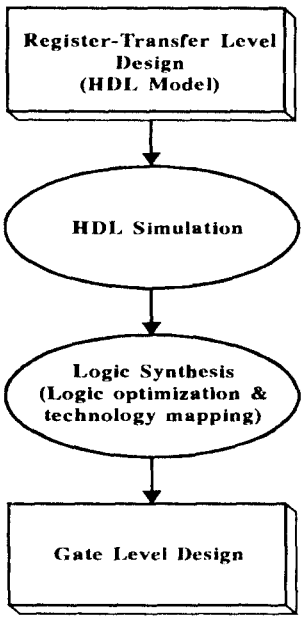


그림 1. 설계 흐름
Fig. 1 Design Flow

일반적으로 HDL을 이용하여 기술된 마이크로프로세서의 검증은 각 명령에 대한 테스트 벡터(Test Vector)나 작은 크기의 응용 프로그램을 이용하여 시뮬레이션 한다[3][4]. 그러나 기존에 발표된 마이크로프로세서와 호환되는 명령을 가지는 마이크로프로세서의 설계시에는 위의 방법 외에 보다 발전된 형태로써, 실제 사용되는 응용 프로그램 자체를 이용하여 검증하는 방법이 필요하다[6][9].

본 논문에서는 오퍼레이팅 시스템과 실제 사용되는 응용 프로그램을 이용하여, 기존의 마이크로프로세서와 호환되는 마이크로프로세서를 검증하는 검증 환경을 구성하는 방법을 제시한다. 마이크로프로세서 호환성 검증 환경에선 HDL로 기술된 마이크로프로세서 전체를 시뮬레이션하기 위하여 시스템의 필수적인 주변 장치인 메모리 등을 HDL로 모델링하고, 부가적인 요소들을 빠른 시뮬레이션을 위해 C 언어로 기술한다.

논문의 뒷 부분에선 제안한 검증 환경 구성 방법을 통해 x86 계열의 마이크로프로세서와 호환되는 마이크로프로세서를 상위수준에서 검증하는 검증 환경을 제시한다.

II. 마이크로프로세서 호환성 검증 환경 구성

마이크로프로세서 호환성 검증 환경은 마이크로프로세서의 HDL 모델을 시뮬레이션하기 위한 가상 시스템(Virtual System)과 시뮬레이션 과정에서 마이크로프로세서의 내부 상태 및 모니터 출력 결과를 보며 전체 시뮬레이션을 제어할 수 있는 GUI(Graphic User Interface) 모듈로 나누어진다. 시뮬레이션 수행 시간의 단축과 호환성 검증 환경 개발의 편리를 위해 두 모듈은 독립된 프로그램으로 개발되며, 두 모듈간의 정보 교환을 위한 통신부(Communication Part)가 존재한다.

두 모듈을 독립적으로 설계함으로써 GUI 모듈의 하드웨어 의존도를 낮출 수 있으며, 이로 인해 다른 마이크로프로세서 호환성 검증 환경을 개발할 경우 이미 개발한 마이크로프로세서 호환성 검증 환경을 다시 이용할 수 있다는 장점이 생긴다.

검증 환경은 마이크로프로세서의 기능을 검증할 시험 프로그램으로 기존의 마이크로프로세서에서 동작하는 오퍼레이팅 시스템(Operating System)을 이진 코드의 변형 없이 이용한다. 오퍼레이팅 시스템뿐만

이 아니라, 오퍼레이팅 시스템이 수행되는 동안에는 기존의 응용 프로그램들을 수행시켜 봄으로써 마이크로프로세서의 호환성 검증 및 기능 검증을 수행하게 된다.

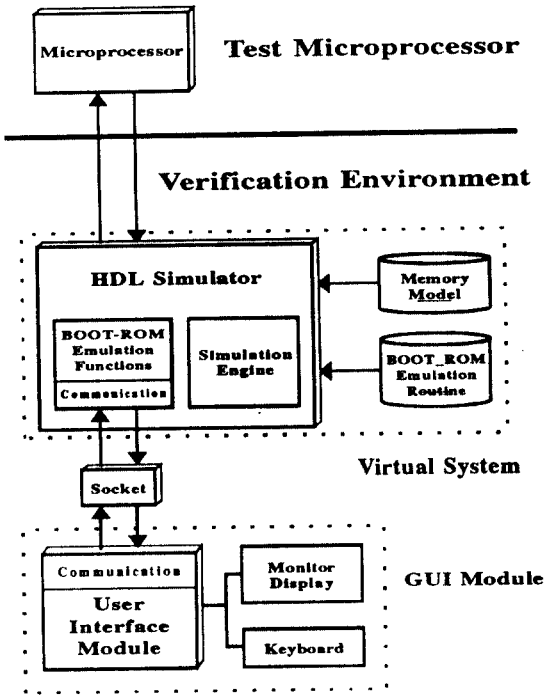


그림 2. 검증 환경의 전체 구성
Fig. 2 Verification Environment

2.1 가상 시스템

가상 시스템은 메모리 모듈, BOOT-ROM 에뮬레이터 및 모니터, 키보드 모듈, 디스크 모듈로 구성되어진다. 모델링한 디스크에는 시뮬레이션에 필요한 오퍼레이팅 시스템과 응용 프로그램들을 저장하도록 한다. 또한 검증하고자 하는 마이크로프로세서가 가상 시스템에 추가됨으로써 가상 컴퓨터가 구성되어 시뮬레이션이 가능하게 된다.

BOOT-ROM 에뮬레이터는 시스템의 기본적인 동작을 담당하는 부분으로 다음과 같은 기능을 가지게 된다. 첫째, 오퍼레이팅 시스템을 메모리에 로드(Load)하고 실행(Execution)시키는 부팅(Booting) 기능을 가진다. 둘째, 모니터(비디오 카드), 키보드, 디스크 등 기본 입출력 장치를 제어하는 루틴을 가진다. 이 외에도 실제 시스템의 ROM에는 POST(Power On Self Test) 루틴 및 시스템의 구성 요소를 진단하는 루틴이 있으나 이는 호환성 검증을 위해 필요한 부분이 아니므로 에뮬레이션 기능에서 제외하였다[10-13].

BOOT-ROM 에뮬레이터에서 기본 입출력을 위한 기능은 인터럽트(Interrupt) 형태로 제공되며, 시뮬레이션 수행시 수시로 호출되므로 속도 향상을 위해 특별히 취급되어야 하는 부분이다. BOOT-ROM을 HDL로 작성하고, BOOT-ROM의 내용을 실제 사용되는 기계어로 작성한다면, 기본 입출력 루틴이 인스트럭션 수준에서 매번 시뮬레이션되어야하므로 시뮬레이션 수행 시간이 크게 증가한다. 따라서 시스템 버스와 연결되는 부분은 HDL로 작성하고, 각 기능 수행에 필요한 루틴들을 C로 작성하여 절차형 언어 인터페이스(Procedural Language Interface)[14]를 이용해 각각의 루틴을 인터페이스하도록 하면 전체 시뮬레이션 시간을 단축시킬 수 있다.

BOOT-ROM 인터럽트 서비스 에뮬레이션 과정은 [그림 3]과 같다. 마이크로프로세서가 인터럽트 서비스를 호출하는 인스트럭션을 만나면 프로그램 카운터는 인터럽트 서비스 루틴으로 점프하게 되며, 점프한 곳에는 인터럽트 핸들러 에뮬레이션 루틴이 존재

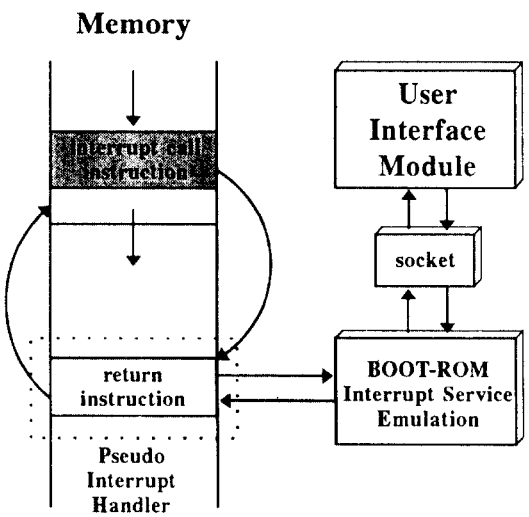


그림 3. BOOT-ROM 인터럽트 서비스 에뮬레이션 과정
Fig. 3 BOOT-ROM Interrupt Service Emulation Process

하게 된다. 에뮬레이션 루틴 중 ROM 코드는 인터럽트 서비스를 종료하는 인스트럭션만을 가지고 있으며, 실제 수행 루틴은 모두 HDL과 C언어를 이용하여 작성된 루틴이다. 이 루틴은 인터럽트 핸들러가 호출되었음이 감지되면, ROM 코드인 인터럽트 서비스 종료 인스트럭션이 수행되기전에 시뮬레이터에 의해 수행됨으로써, 인터럽트 서비스 기능을 에뮬레이션 할 수 있는 것이다.

기본 입출력 기능을 수행하기 위해 필요한 함수들을 [표 1]에 나타내었다.

표 1. BOOT-ROM 에뮬레이션을 위한 함수들
Table 1. Functions for a BOOT-ROM emulation

분 야	함 수 명	
디스크	$get_parameter(); $set_drive_parameter(); $read_status();	$clear_buffer();
	$read_sector(); $write_sector(); $name_disk();	
	$peek();	$poke();
비디오 카드 (모니터)	$crt_name();	$putchar();
	$poke_imageb();	$poke_imagew();
	$send_image();	
키보드	$keyboardhit();	$get_keyboard();

키보드 모듈과 모니터 모듈, 디스크 모듈은 오퍼레이팅 시스템에 따라 다르게 구성되어진다. PC 시스템에서 동작하는 DOS의 경우 키보드 및 모니터를 제어하기 위해 ROM-BIOS 서비스 루틴을 그대로 이용하므로, C 언어를 이용해 구현된 에뮬레이션 루틴이 수행되어 빠른 시뮬레이션을 가능하게 한다. UNIX를 오퍼레이팅 시스템으로 채택한 경우 ROM의 기본 입출력 서비스는 부팅과 하드웨어의 분석에 이용되고, 오퍼레이팅 시스템의 부팅 후에는 UNIX 자체의 코드를 사용하게 된다. 따라서 이 경우에는 키보드, 모니터 등의 입출력 장치를 오퍼레이팅 시스템이 이용할 수 있도록 입출력 콘트롤러(Controller)를 모델링하여야 한다.

2.2 GUI(Graphic User Interface) 모듈

GUI 모듈은 시뮬레이션 과정에서 사용자의 입력을 받아 HDL 시뮬레이터에게 전달해 주거나 HDL 시뮬레이터로부터 받은 정보를 화면에 나타낸다. HDL 시뮬레이터로부터 받는 정보로는 비디오 메모리의 내용, 내부 레지스터의 값, 현재 수행 중인 명령어, 포괄적인 시뮬레이션 상황 등이며, 사용자가 입력한 키값을 스캔(SCAN)코드와 아스키(ASCII)코드로 변환하여 HDL 시뮬레이터에게 보내준다.

이 모듈의 외형을 나타낸 것이 [그림 4]이다.

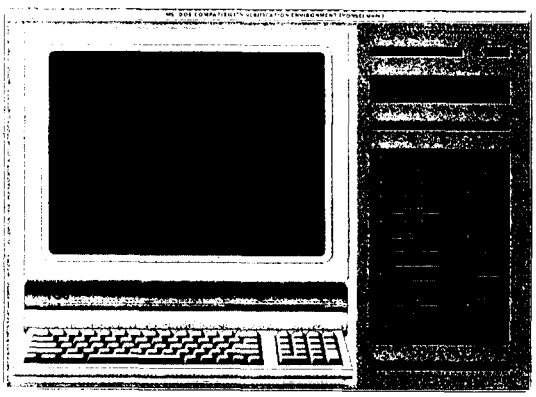


그림 4. GUI 모듈의 외형
Fig. 4 Appearance of the GUI Module

GUI 모듈은 [그림 4]에서 보이듯이 시스템의 기본적인 동작에 필수적인 키보드와 모니터 외에 시뮬레이션 진행 과정 및 결과를 보다 쉽게 관찰할 수 있도록 하여주는 윈도우(Window)를 가지고 있다. 이는 시스템의 초기화가 어느 정도 진행되었는지를 알려 주는 윈도우, 현재 수행되고 있는 명령(Instruction)을 알려주는 윈도우, 마이크로프로세서의 내부 레지스터들의 값을 나타내 주는 윈도우 등으로 구성된다.

사용자를 위한 그래픽 환경을 구성하기 위해 X/Motif 라이브러리를 이용하고, 모든 코드를 C로 작성하며, 서버(Server)-클라이언트(Client) 모델에서 서버로 동작하게 된다[9][10].

2.3 모듈간의 인터페이스

GUI 모듈은 HDL의 절차형 인터페이스 프로그램을 이용하여 C 언어로 작성됨으로써, HDL 시뮬레이터의 한 프로그램으로 동작할 수 있다. 하지만 이러

한 경우에 시뮬레이터와 GUI 모듈이 동시에 한 컴퓨터에서 실행되므로 많은 메모리를 사용할 뿐만 아니라 실행 속도 또한 시뮬레이터만 돌리는 것보다 상당히 저하되는 문제점이 발생한다. 이를 피하기 위해 본 검증 환경에서는 GUI 모듈과 가상 시스템을 독립적인 프로그램으로 동작하도록 하며, 두 모듈의 통신을 위한 인터페이스 모듈을 둔다.

두 모듈의 통신을 위한 인터페이스 모듈은 GUI 모듈과 같이 C 언어만을 사용하여 작성되며, 유닉스(UNIX)의 소켓 라이브러리를 이용하여 두 모듈의 통신을 담당하도록 한다[10]. 따라서 두 모듈이 서로 다른 컴퓨터에서 작동할 수 있게 되어 효율적인 메모리 사용 및 전체 시뮬레이션의 실행 속도를 향상시킬 수 있다.

GUI 모듈과 가상 시스템의 통신에서 GUI 모듈은 서버(Server)로 동작하고 가상 시스템은 클라이언트(Client)로 동작한다.

Ⅲ. x86 계열의 마이크로프로세서 호환성 검증 환경

x86 계열의 마이크로프로세서 호환성 검증 환경은 퍼스널 컴퓨터(Personal Computer)를 모델링한 가상 시스템과 시뮬레이션을 제어할 GUI 모듈로 나누어진다. 두 모듈은 각각 독립된 프로그램으로 동작하며 서로의 정보를 교환하기 위한 통신부를 두었다. 가상 시스템은 HDL 시뮬레이터와 PC의 ROM-BIOS를 에뮬레이션하는 BOOT-ROM 에뮬레이터, 1 MBytes의 메모리 모듈, DOS 프로그램과 시뮬레이션을 위한 응용 프로그램들을 포함하는 5.25 인치 디스크 및 3.5 인치 디스크 모듈로 이루어지며, 입출력 기능은 BOOT-ROM 에뮬레이터에서 담당하도록 하였다.

DOS 및 대부분의 응용 프로그램은 ROM-BIOS의 입출력 루틴을 자주 호출한다. 따라서 입출력 모듈을 별도로 설계하는 경우에는 ROM-BIOS의 입출력 루틴을 기계어로 작성해야 하므로 입출력 루틴이 호출될 때마다 해당 루틴의 기계어가 반복해서 시뮬레이션 되는 문제점이 있다. 그러므로 입출력 기능을 키보드 콘트롤러, 비디오 콘트롤러 등의 입출력 모듈에서 수행하지 않고, C 프로그래밍 언어를 이용하여 작성한 BOOT-ROM의 입출력 루틴에서 수행하도록 하여 ROM-BIOS의 인스트럭션 시퀀스(sequence)가 반복해서 수행되어지는 문제점을 해결하였다.

시뮬레이션이 시작되면, 가상 시스템은 디스크로부터 MS-DOS 시스템을 읽어 부팅을 시작하며, 부팅이 끝난 후엔 도스 응용 프로그램들을 실행할 수 있다.

전체적인 시뮬레이션은 [그림 5]와 같은 순서로 진행된다.

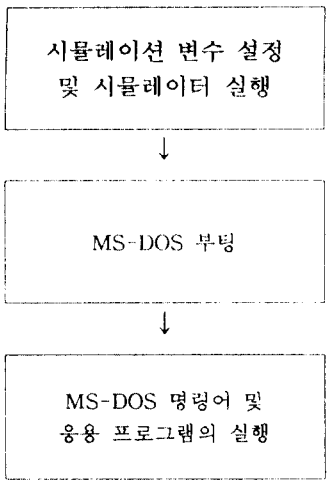


그림 5. 시뮬레이션 수행 과정
Fig. 5 Simulation Flow

호환성 검증 환경에서는 MS-DOS 버전 3.3, 5.0, 6.0 각각에 대해 테스트할 수 있도록 각 프로그램이 들어 있는 5.25 인치 및 3.5 인치용 디스크를 파일로 만들었으며, 변수 지정 단계에서 만들어진 6개의 MS-DOS 디스켓 파일 중 하나를 선택하도록 하였다. 변수 지정은 MS-DOS 버전과 디스크 용량 외에도 GUI 모듈이 실행되는 워크스테이션의 IP(Internet Protocol) 어드레스(Address)와 마이크로프로세서 내부 신호의 파형 표시 여부 등을 지정해 주도록 하였다.

MS-DOS의 부팅 과정에선 진행 과정을 GUI 모듈의 초기화 윈도우에 백분율로 표시한다.

MS-DOS 명령어 및 응용 프로그램 실행 과정은 MS-DOS가 부팅된 상황에서 MS-DOS의 프롬프트(prompt)가 나타나면 실제 PC에서 키보드를 통해 명령을 입력하는 것과 같은 방법으로 임의의 명령어를 실행하는 순서로 진행된다. 한 프로그램의 실행이 끝

나면 또다른 명령 혹은 프로그램을 연속적으로 실행시킬 수 있으며, 이 과정에서 사용한 명령 및 응용 프로그램은 [표 2]와 같다.

표 2. 시뮬레이션에 사용된 도스 명령어와 응용 프로그램
Table 2. MS-DOS commands and application programs used in simulation

분류	MS-DOS 명령, 응용 프로그램
MS-DOS 부팅	'TIME', 'DATE' 'RAMDRIVE.SYS'
MS-DOS 내부 명령어	'DIR', 'DEL', 'REN', 'COPY', 'CLS', 'PROMPT', 'MKDIR', 'CHDIR', 'RMDIR', 'TIME', 'DATE'
MS-DOS 외부 명령어	'CHKDSK', 'FORMAT', 'MEM', 'DEBUG', 'LABEL', 'TREE', 'MORE', 'SYS'
응용 프로그램	'TETRIS', '오목', 'TANGMAN', 'HANOI', 'QSORT', 'BASEBALL'

[그림 6]은 'TETRIS' 게임의 실행 화면으로 실제 PC의 모니터에 출력되는 결과와 같음을 확인할 수 있다.

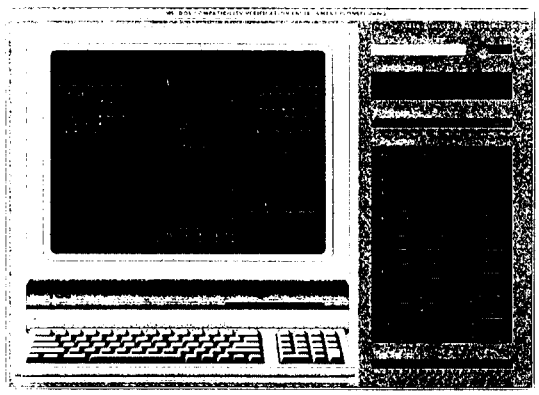


그림 6. 'TETRIS' 프로그램의 수행 화면
Fig. 6 The picture of executing 'TETRIS'

시뮬레이션은 SunOS 4.1.3을 OS(Operating System)로 사용하는 워크스테이션(Workstation)에서 수행되었다. GUI 모듈과 가상 시스템이 서로 다른 컴퓨터에서 분산 처리될 경우 한 대의 컴퓨터에서 수행되는 경우보다 수행 속도가 최대 51% 단축되었으며, 이는 각 모듈의 CPU 점유율에 기인함을 알 수 있었다. 한 대의 컴퓨터에서 수행될 경우 GUI 모듈과 가상 시스템의 CPU 점유율은 45 퍼센트 내외를 기록하였으며, 두 대의 컴퓨터를 이용한 경우 각 모듈의 CPU 점유율은 90 퍼센트 이상을 기록하였다.

Ⅳ. 결 론

본 논문은 HDL로 설계된 마이크로프로세서와 기존의 마이크로프로세서와의 호환성을 검증하는 환경에 대해 기술하였다.

검증 환경은 개발의 편리와 시뮬레이션 수행 시간의 단축을 위해 GUI 모듈과 가상 시스템을 서버-클라이언트 모델을 지원하는 독립적인 프로그램으로 설계하였다. 가상 시스템의 핵심 부분인 BOOT-ROM 에뮬레이터 및 입출력 디바이스는 HDL과 C를 인터페이스하여 빠른 시뮬레이션 속도를 얻을 수 있었다. 또한 제안한 호환성 검증 환경에서는 오퍼레이팅 시스템을 수행시킴으로써, 수행된 오퍼레이팅 시스템을 이용하는 응용 프로그램들을 마이크로프로세서의 기능 검증에 이용할 수 있는 부차적 효과를 얻을 수 있었다. 따라서 이용 가능한 테스트 프로그램의 수를 증가시킬 수 있었으며, 기존의 시스템에서 실행되는 코드 전체를 변형 없이 이용함으로써 호환성 검증 결과의 신뢰성을 높일 수 있었다.

x86 마이크로프로세서 호환성 검증 환경의 실험 결과는 각각의 모듈을 서로 다른 워크스테이션에서 분산 처리하는 것이 한 대의 워크스테이션에서 시뮬레이션 한 경우보다 최대 51%의 수행 시간이 단축됨을 보여 준다.

일대 일의 서버-클라이언트 모델은 일대 다의 서버-클라이언트 모델로 바뀔 수 있다. 이는 하나의 서버에 여러 클라이언트가 연결되는 형태로 각각의 클라이언트는 독립적으로 동작하며, 서버의 제어를 받게 된다. 일대 다의 서버-클라이언트 모델을 구성하면 시뮬레이션 관찰자는 동시에 진행되는 여러가지 시뮬레이션을 관찰할 수 있으며, 하나의 서버를 사용한다는 면에서 시스템의 자원을 절약하는 잇점을 얻게 된다.

參 考 文 獻

1. 이정엽, "HDL 모델 마이크로프로세서의 MS-DOS 호환성 검증 환경, 구현," 연세대학교 대학원 전자공학과 석사학위 논문, 1995
2. CADENCE, *Cadence Design Automation Show '93*, 1993
3. Moon Key Lee, "32 bit YONSEI SPARK RISC Microprocessor," Journal of The Research Institute of ASIC Design, Vol. 1 No. 1, 1994
4. Raj Jain, *The Art of Computer Systems Performance Analysis*, John Wiley & Sons, 1988
5. Eli Sternheim, Rajvir Singh, Rajeev Madhavan, Yatin Trivedi, *Digital Design and Synthesis with Verilog HDL*, 1993
6. M. Gupta and B. Zivkov, "A Structured Verification Approach for MIPS Microprocessors: A Case Study of the R4200," digest of papers COMPCON94, Feb 28-Mar 4. 1994
7. 이문기, VLSI 기술을 이용한 RISC 마이크로프로세서 개발에 관한 연구, 상공부 중간 보고서(I), 연세대학교 ASIC 설계 공동 연구소, 1991
8. 이문기, VLSI 기술을 이용한 RISC 마이크로프로세서 개발에 관한 연구, 상공부 중간 보고서(II), 연세대학교 ASIC 설계 공동 연구소, 1992
9. 장훈, "메모리 관리 유닛/캐쉬 제어기 평가 시스템 설계 및 제작," 연세대학교 대학원 전자공학과 석사학위 논문, 1994
10. Nucgaek Tusgerm, *PC Intern System Programming*, 1992
11. Ray Duncan, *Advanced MS-DOS*, Microsoft Press, 1986
12. 한성학, IBM-PC 기술 사전, 집문당, 1989
13. FORCE Computers, *SPARC CPU-2CE TECHNICAL REFERENCE MANUAL*, FORCE Computers, 1992
14. Cadence, *Verilog Programming Language Interface*, Cadence, 1994
15. Marshall Brain, *MOTIF Programming: The Essentials ... and More*, Digital Press, 1992
16. Paula M. Ferguson, MOTIF Reference Manual for OSF/Motif Release 1. 2, O'Reilly & Associates, Inc, 1993
17. Adrian Nye, Xlib Reference Manual for Version 11, O'Reilly & Associates, 1992
18. Adrian Nye, Xlib Programming Manual for Version 11, O'Reilly & Associates, 1992
19. W. Richard Stevens, UNIX Network Programming, Prentice-Hall, 1991

이 문 기(Moon Key Lee)정회원

1941년 8월 23일생

1967년 : 연세대학교 전자공학과 강사

1969년 : 광운공과대학교 전자공학과 전임 강사

1970년 : 경희대학교 전자공학과 조교수

1973년 : 경희대학교 전자공학과 부교수겸 학과장

1976년 : University of OKLAHOMA 연구소 연구원

1980년 : 한국전자기술 연구소(현 ETRI) 설계자동화(CAD) 국책연구과제 수행 책임 연구원

1983년 : 금성 반도체 주식회사 비상임 기술고문

1984년 : 기아 산업 주식회사 종합연구소 비상임 기술고문

1989년 : 연세대학교 부설 아식설계 공동연구소 부소장

1990년 : 연세대학교 전자공학과 학과장

1995년 : 대한 전자공학회 회장

현재 : 연세대학교 교수, 연세대학교 부설 아식설계 공동연구소 소장

※주관심 분야 : 마이크로프로세서 설계, 디지탈 영상처리 칩 설계, 설계자동화 등

김 영 완(Young Wan Kim) 정회원

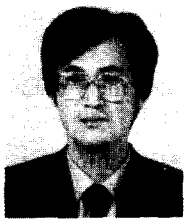
1971년 2월 21일생

1994년 2월 : 연세대학교 전자공학과 졸업

1996년 2월 : 연세대학교 전자공학과 졸업(공학석사)

현재 : 삼성전자 연구원

※주관심 분야 : 설계 자동화 등



서 광 수(Kwang Soo Seo) 정회원
1960년 9월 21일생
1983년 2월 : 경희대학교 전자공학과 졸업
1985년 8월 : 연세대학원 전자공학과 졸업(공학석사)
1985년~1992년 2월 : LG 반도체(주) 선임연구원
현재 : 연세대학교 전자공학과 박사 과정 재학중
※주관심 분야 : 상위수준합성, 데이타 압축, 모듈생성 및 Smart 메모리 아키텍처등

손 승 일(Sonh Seung IL) 정회원
1965년 10월 24일생
1989년 2월 : 연세대학교 공과대학 전자공학과 졸업
1991년 8월 : 연세대학원 전자공학과 졸업(공학 석사)
현재 : 연세대학원 전자공학과 박사 과정 재학중
※주관심 분야 : VLSI & CAD 및 마이크로프로세서 설계 등