論文2000-37SD-11-10

32비트 부동소수점 호환 DSP의 설계 및 칩 구현에 관한 연구
(Study on Chip Design & Implementation of 32 Bit Floating Point Compatible DSP)

禹 鍾 植*, 徐 鎭 瑾**, 林 在 英*, 朴 柱 成**
(Jong Sik Woo, Jin Keun Seo, Jae Young Lim, and Ju Sung Park)

요 약

본 논문은 TMS320C30과 호환되는 DSP(Digital Signal Processor)를 설계하고 구현하는 과정을 다룬다. 구조 설계를 위하여 DSP의 파이프라인 사이클마다 일어나는 일을 정의하기 위한 CBS(Cycle Based Simulator)를 구현하였다. CBS는 특정 명령어가 수행되기 위한 기능블럭의 동작, 제어신호 값, 각종 레지스터 값, 메모리 값 내부 버스의 값들을 제공해 주기 때문에 VHDL 코딩시의 중요한 레퍼런스가 된다. 논리 설계는 VHDL을 사용하였다. 설계된 DSP 검증을 위하여 논리 시뮬레이션 및 하드웨어 에뮬레이션을 하였다. 설계된 DSP는 0.6㎛ CMOS 라이브러리를 이용하여 구현하였다. 칩 복잡도는 45만 게이트이며 칩 크기는 9×9mm^{2}이고 동작 속도는 20 MIPS이다. 제작된 칩을 이용하여 114종 명령에서 109개의 명령어와 13종의 알고리즘을 수행시켜 정상적으로 동작하는 것을 확인하였다.

Abstract

This paper deals with procedures for design and implementation of a DSP, which is compatible with TMS320C30 DSP. CBS(Cycle Based Simulator) is developed to study the architecture of the target DSP. The simulator gives us detailed information such as function block operation, control signal values, register condition, bus and memory values when a instruction is being carried out. RTL design is carried out by VHDL. Logic simulation and hardware emulation are employed to verify proper operation of the design. The DSP is fabricated with 0.6㎛ CMOS technology. The Chip has 450,000 gates complexity, 9×9 mm^2 area, 20 MIPS operation speed. It is confirmed by running 109 instructions out of 114 instructions and 13 kinds of algorithm that the developed DSP has compatibility with TMS320C30.

I. 서 론

DSP는 수학적 계산이 많은 복잡한 알고리즘의 하드웨어 구현과 데이타의 처리를 고속으로 수행할 수 있도록 내부에 특수한 기능블럭을 갖고 있으므로 실시간 멀티미디어 서비스에 많이 응용되고 있다. 일반적으로 DSP는 데이타 처리방식에 따라 부동소수점 DSP와 고정소수점 DSP로 구분된다. 프로세서를 이용하여 데이타를 처리할 경우에 부동소수점 데이타는 고정소수점 데이타에 비하여 아주 정밀하게 수치를 표현할 수 있기 때문에 정밀도가 요구되는 분야에 응용되고 있다. 또한 부동소수점 DSP는 C-Compiler의 결과를 그대로 사용해도 코드 길이 측면에서 큰 무리가 없기 때문에 사용자가 빠른 시간 내에 프로그램을 완성 할 수 있는 장점이 있다. 현재 일반 사용자의 욕구를 만족시키며 섬세하고 정밀한 실시간 멀티미디어 서비스를 제공하는 제품들은 부동소수점 DSP 코어를

* 正會員, ((株)보이소半導體)
(VOISO Semiconductor Co., Ltd.)
** 正會員, 釜山大學校 電子工學科
(Dept. of Electronics Engineering, Pusan National University)
※ 본 연구는 산업자원부, 과학기술부, 정보통신부에서 시행하는 주문형반도체 개발사업의 지원을 받아 수행되었습니다.
接受日字:2000年 4月10日, 수정완료일:2000年 10月17日

내장하는 추세로 가고 있다. 따라서 본 연구에서는 현재 널리 사용되는 TMS320C3X 32비트 부동소수점 DSP와 호환되는 칩을 설계하고 검증하는 것을 목표로 한다.

전체적인 설계 및 칩 제작 흐름에 따라 Ⅱ장에서 설계 DSP의 주요 특징을, Ⅲ장에서 설계에 필요한 구조설계 및 CBS의 제작을, Ⅳ장에서는 제어 블록 설계를, Ⅴ장에서 DFT(Design For testability)를 고려한 회로 설계를, Ⅵ장에서는 DSP의 동작 검증 및 호환성 검증을, Ⅶ장에서는 칩 제작 및 칩 테스트에 관한 내용을, 마지막으로 Ⅷ장에서는 결론을 내리고자 한다.

Ⅱ. 설계 DSP의 주요 특징

목표 DSP는 4단계의 파이프라인으로 동작하며 32비트로 구성된 114종의 명령어 세트를 가진다. DSP의 내부는 중앙처리장치, 곱셈기, Barrel shifter, register files, 버스 등을 포함하고 외부인터페이스를 위해 인터럽터, 입출력 포트, DMA, Timer, Serial port 등으로 구성되어 있다. 본 논문에서 설계될 DSP의 사양을 간략히 정리한다.

1. 파이프라인 구조

목표 DSP의 파이프라인 구조는 program fetch (FETCH), decode(DEC), operand fetch (READ), execute(EXE)의 4레벨로 나누어지며, 매 사이클마다 하나의 명령어를 수행한다.[1]

2. 명령어 세트

설계될 DSP가 수행하는 명령어는 register to register, memory referance, branch/call, delayed branch/call, two/three operand arithmetic/logical, program control instruction 등의 114종으로 구성되어진다.[1,2]

3. 버스

버스는 표 1과 같이 총 14개로 구성된다. 특히 데이타 버스는 2개의 데이타 어드레스 버스와 데이타 데이타 버스로 구분되고 1사이클에 메모리의 내용을 두 번 read 하도록 하여 버스 대역폭[3]을 높였다.

4. 중앙처리 장치

DSP의 중앙처리장치는 40비트 부동소수점, 32비트 고정소수점 곱셈기와 32비트 barrel shifter, ALU를 내장하여 ALU 연산과 곱셈 연산을 1사이클에 병렬로 수행할 수 있다. 또한 다양한 기능을 수행할 수 있는 28-CPU register files와 간접 어드레스 연산을 수행하는 2개의 ARAU(Auxiliary Register Arithmetic Unit)으로 구성된다.

표 1. 버스의 종류
Table 1. Kinds of bus.

구분	버스 종류	할당 비트 수
프로그램	어드레스	24/13bit
	데이타	32bit
데이타	어드레스	24bit
	데이타	32bit
DMA	어드레스	24bit
	데이타	32bit
CPU	데이타	40bit
Register	데이타	40bit
Peripheral	어드레스	24bit
	데이타	32bit

5. 어드레싱 모드

DSP가 데이타를 읽어 오는 방법으로 총 8가지의 방법이 있다.

- general 어드레싱 모드[1]
- three operand 어드레싱 모드
- 병렬 어드레싱 모드
- long immediate 어드레싱 모드
- 조건 분기 어드레싱 모드
- circular 어드레싱 모드
- 비트 반전 어드레싱 모드
- 시스템 스택 어드레싱 모드

6. Peripherals

외부 인터페이스를 위해 serial port, 32bit timer, external interrupt, DMA controller를 포함하고 있다.

Ⅲ. 구조 설계 및 CBS의 제작

수십만 게이트급의 DSP나 ASIC을 설계함에 있어서 검증시간이 많은 부분을 차지하고 있다. 따라서 검증시간을 줄이기 위해서는 게이트 레벨 설계와 같은

구체적인 작업 이전에 설계될 VLSI의 구조, 명령어, 주변과의 인터페이스 등을 종합적으로 검토할 수 있는 방안이 필요하다. 이러한 목적으로 제작된 시뮬레이터는 설계될 DSP의 파이프라인 사이클을 기반으로 모델링 되었다. 이러한 연유로 시뮬레이터를 CBS(Cycle Based Simulator)라고 명명하였다. 본 논문에서 언급되는 CBS는 명령어 동작, 어드레싱 모드, 내부 기능블럭, 파이프라인 구조 등이 포함되어 있으므로 전체 설계과정의 기준이 된다.

제작된 CBS는 어셈블리 코드를 입력으로 하여 파이프라인 사이클마다 DSP의 내·외부 데이타를 출력하는 것을 목표로 하는 GUI(Graphic User Interface)로 제작되었으며 외부인터페이스를 위해 자체적으로 링크 및 어셈블러가 포함되어 있다. 또한 CBS를 작성한 C 언어와 VHDL은 모두 상위 레벨 언어라는 점에서 많은 유사점이 있으므로 CBS를 제작할 때에 미리 VHDL로의 변환을 고려해두어 C 언어에서 VHDL로 변경하는데 많은 시간이 소요되지 않았다.

CBS의 제작방법과 최종 결과인 GUI환경을 아래에 간단히 요약한다.

1. CBS의 제작환경

일반적으로 CBS를 제작할 수 있는 언어는 다양하지만 C언어는 프로그램의 확장성 및 이식성, 고속처리 수행 능력이 탁월하다. 이러한 이유로 CBS의 제작은 TMS320C30 DSP의 확장레지스터(40비트)를 표현할 수 있는 64비트 GCC-C의 Unix 환경하에서 수행되었다.

2. 제작 흐름

DSP의 하드웨어 동작을 고려하여 CBS를 입·출력 신호, 제어신호와 하드웨어모델로 구성하였다.

■입·출력 신호 및 제어신호

DSP의 명령어 동작을 고려하여 DSP 내부의 모든 기능블럭의 입·출력 데이타 및 제어신호를 정의한다.[4]

■하드웨어 모델[2]

이미 정의된 입·출력과 제어신호를 이용해서 원하는 명령어 동작을 수행할 수 있도록 하드웨어 모델의 동작을 기술한다. 본 논문에서 대상으로 삼은 TMS320C30의 기능블록들을 100여 개의 함수로 모델링 했다. 각 모델은 게이트 단계까지 고려하여 제어신호를 정의하고, 제어 타이밍은 명령어 사이클에 기준하여 정의한다.

3. GUI 제작[5]

CBS에서 출력되는 결과는 여러 종류의 버스값, 제어신호와 기능블럭의 입출력 데이타, 레지스터와 메모리의 값 등이다. 이러한 여러 가지 데이타를 텍스터 형태로 출력해서 확인하는데는 많은 어려움이 있다. 이러한 문제점을 극복하기 위하여 MOTIF, X11, Xt, Xm 등의 세부 Library를 사용하여 CBS를 GUI로 제작했다. 그림 1과 같이 GUI는 어떠한 명령어가 Execution, Read, Decode, Fetch 단계에 있는가를 구분하며 상용시뮬레이터에서 제공되지 않는 모든 메모리 값, 버스 값, 제어신호와 CBS 구성에 사용된 모든 모델의 입·데이타 정보를 표현할 수 있다.

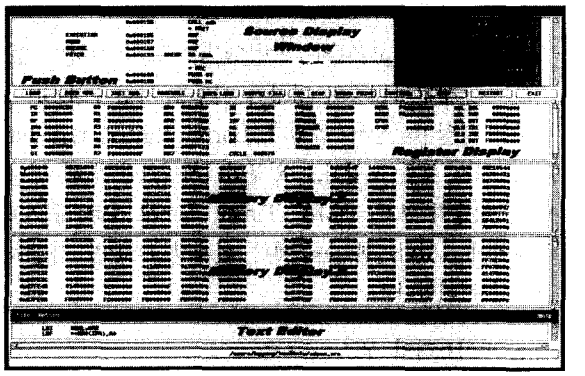


그림 1. CBS의 GUI 화면
Fig. 1. GUI Screen of CBS.

4. CBS의 동작 검증

CBS를 제작 단계에서 오류를 줄이기 위하여 제작 단계마다 철저한 검증을 수행했다. 각 블록을 설계한 후에는 기능블록의 동작 검증을, CBS 완성 후에는 명령어 검증과 여러 명령어를 조합하는 조합 명령어 동작 검증을, 마지막으로 널리 알려져 있는 여러 응용프로그램[6~8]을 이용하여 검증을 하였다. 네 단계의 검증은 상용 시뮬레이터[9,10]와 동일한 환경에서 수행한 후에 DSP 에뮬레이터에서 얻을 수 있는 모든 결과를 비교 기준으로 삼아 약 2만 사이클의 검증을 수행하였다.

Ⅳ. 제어블럭 설계

설계된 DSP는 데이타처리 기능블럭과 제어신호의

발생 및 제어 타이밍을 구현한 제어블럭으로 구성된다. ALU, 승산기, Barrel Shifter 등을 포함하는 기능블럭은 CBS의 하드웨어 모델을 기준으로 VHDL로 변환하였으며, 파이프라인 단계에 따른 3가지의 제어블럭을 생성하여 제어신호를 발생 시킬 수 있도록 하였다.

목표 DSP를 제어하기 위하여 하드와이어와 마이크로프로그램 ROM 방식[11-13]을 결합하였다. DSP의 각 사이클별 파이프라인 동작은 각각 1개의 마이크로 프로그램 ROM으로 구성되며 각 사이클별 세부 동작 제어와 파이프라인 사이의 제어는 2개의 파이프라인 IR(Instruction register) 버퍼를 이용하여 하드와이어적[12]으로 처리하였다. 파이프라인 IR 버퍼는 각각의 파이프라인 상에서 현재 수행되는 명령어를 가리키며, 명령어 수행에 필요한 공동 제어신호를 하드와이어 방식으로 발생시킨다. 각 명령어에 필요한 공통된 제어신호가 거의 없는 제어신호에 대하여 마이크로 프로그램 ROM방식을 사용하여 설계자가 제어신호를 바꾸기가 용이하게 하였다. 마이크로 프로그램 ROM은 파이프라인 제어를 위해 3개의 ROM으로 나누어 구성되었다. 단 fetch 제어신호의 경우는 ROM의 폭이 작아 decode ROM에 포함시켰다. 각 ROM의 폭×길이는 decode의 경우는 26×119, read는 22×116, execute는 33×118로 구성되었다. 이와 같은 ROM필드를 수동으로 코딩할 경우 많은 시간이 소모되고 오류가 발생할 가능성이 많아 마이크로 어셈블러를 별도로 제작하였다.

DSP의 명령 수행은 몇 개의 마이크로 명령어로 구성되어 있고, 마이크로 명령어에 해당하는 제어신호를 ROM 필드에 정의하였다. 이때 ROM 필드에 이진 값을 직접 기입하지 않고 필드 값을 사용하여 이진 데이타 파일로 변환해 주는 프로그램이 마이크로 어셈블러[4,11]이다. 마이크로 프로그램 ROM 테이블의 제작 예를 그림 2에 도시하였다. 먼저 C와 같은 "/* */"를 주석으로 사용하여 각 필드의 의미와 정보를 표시한다. 필드 정의부는 필드 이름 정의 및 필드 폭의 값을 설계자가 정의할 수 있도록 하였다. 사용자 필드정의 영역은 "FORMAT"으로 시작하고 "필드이름(최대필드폭)"형태로 정의하게 하였다.

그림 2에서 사용된 비트 수는 총 16진수이며 명령어각각에 대하여 각 필드의 값을 정의하였다. 마이크로 어셈블러는 어셈블리 형태의 입력 파일을 읽어 배열 형태의 이진 출력 파일을 발생시킨다. 이진 형태의 배열을 VHDL 형태의 ROM으로 표현하기 위해서 VHDL 형태로 변환 시켜주는 ROM 생성 변환기를 제작하여 ROM 테이블을 만들었다. 마이크로 프로그램 ROM 블럭 구조는 그림 3에 나타나 있다. 어드레

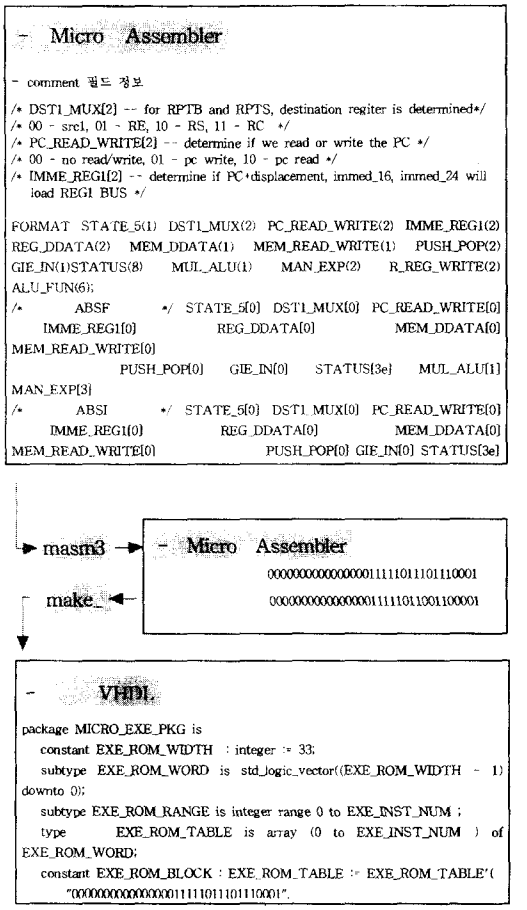


그림 2. 마이크로 프로그램 ROM 제작 예
Fig. 2. Example of microprogrammed ROM.

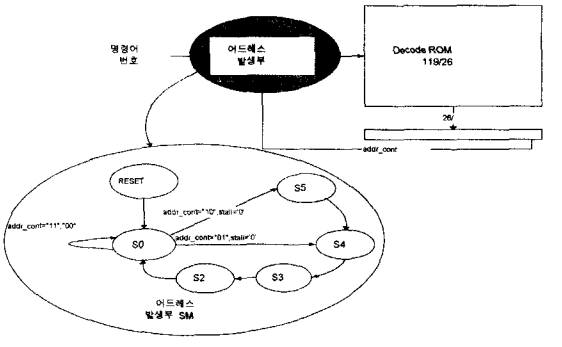


그림 3. 마이크로 프로그램 ROM 구조
Fig. 3. Structure of microprogrammed ROM.

스 발생부[12,13]에 명령어와 Branch 정보가 입력되면 어드레스 발생부의 SM(State Machine)[11~13]은 파이프라인을 고려한 마이크로프로그램 ROM 어드레스를 발생시킨다.

V. DFT를 고려한 회로 설계

설계된 DSP가 칩으로 제작되었을 때, 칩 동작을 테스트 할 수 있는 테스트 회로[14~16]를 첨가하여 테스트를 용이하게 하는 것이 필요하다. 본 연구에서는 테스트를 위하여 설계된 DSP의 외부와 내부에 테스트 회로를 삽입하였다.

DSP의 외부 테스트 회로는 보드레벨 테스트에 사용되는 JTAG(Joint Test Action Group)의 표준안인 IEEE1149.1[15]에 따라 경계 주사회로, TAP 제어기, 명령어 레지스터, 테스트 명령어, 디코더로 설계되었다.

DSP의 내부 테스트는 BIST[16,17] 방식을 이용하여 수행되었다. BIST(Built-In Self Test) 방식은 VLSI 칩 외부에서 테스트 장비를 사용하지 않고 대상회로의 내부에 테스트에 필요한 회로를 삽입함으로써 외부에서 간단한 제어를 통해 내부회로를 테스트하는 방법이다. 대상 DSP 내부의 핵심적인 조합회로는 ALU (Arithmetical and Logical Unit), 곱셈기, 메모리로 각 15,000gates 이상의 면적을 차지하고 있으며 BIST로 구현하면 테스트 시간을 크게 줄일 수 있는 장점을 가진다. 본 논문에서는 BIST 회로의 고장 검출률(fault coverage)을 95% 이상으로 하여 시뮬레이션은 synopsys사의 Test-Compiler 중의 TestSim[18]으로 수행되었다.

BIST는 테스트할 대상 회로의 입력과 출력의 테스트 패턴을 생성하는 TPG(Test Pattern Generator)와 테스트 응답을 분석하고 압축하는 signature analyzer로 구성된다. TPG와 Signature analyzer로 여러 가지 방법이 있지만 본 연구에서는 PRPG(Pseudo Random Pattern Generator)와 MISR(Multiple Input Signature Register)을 사용하였다.[17]

고장 시뮬레이션에서 PRPG 테스트 패턴 생성의 문제점 중의 하나는 90% 정도까지는 비교적 적은 수의 패턴으로 고장 검출이 가능하지만, 그 이상의 고장 검출은 많은 테스트 패턴을 요구한다. 이러한 문제를 해결하기 위해 PRPG의 seed를 갱신하는 방법을 사용하였다. PRPG는 완벽한 무작위 패턴을 만들 수 없으며, 이전의 패턴과 현재의 패턴 그리고 다음 패턴은 서로 연관관계를 가진다. 따라서 검출되지 않은 10% 정도의 고장을 검출하기 위하여 PRPG의 seed를 바꾸면, 새롭게 생성되는 패턴들은 이전의 패턴들과 상관도가 낮아져서 이전의 패턴들이 검출하지 못했던 고장을 검출할 확률도 높아지게 된다. 본 연구에서는 ALU, 곱셈기, ROM, RAM의 테스트 패턴 생성을 위해서 seed를 갱신하였다.

그림 4는 6번의 seed 갱신을 통한 ALU의 고장 검출률을 표현한 것이다. 그림 4에서 95%의 고장 검출률을 달성하는데 필요한 테스트 패턴 수는 4,230개다. 이와 같은 테스트 패턴 수는 seed 갱신을 하지 않은 경우인 13,800개보다 훨씬 작은 수이다. 그리고 고장 검출률 95%이상을 확실하게 보장하기 위해서 5,500개의 테스트 패턴을 사용했으며, 이때의 고장 검출률은 95.35%이다. 한편, seed 갱신 방법을 사용하지 않은 경우는 5,500개의 패턴을 사용하면 94.29%의 고장 검출률을 나타낸다.

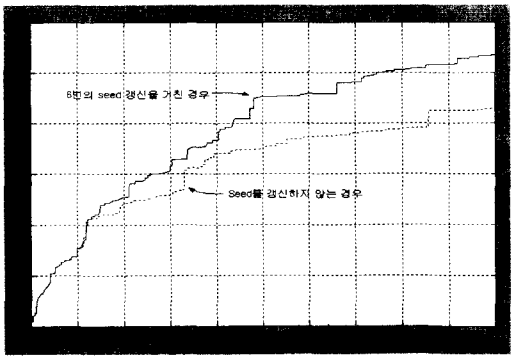


그림 4. Seed가 6번 갱신되는 경우의 ALU 고장 검출률
Fig. 4. Error dectection ratio of ALU in case of seed to be reformed 6'th times.

곱셈기를 위한 테스트 패턴 생성에도 ALU와 마찬가지로 seed 갱신 방법이 사용되었으며, seed가 갱신되지 않은 경우는 95 %의 고장 검출률에 도달하는데 1,813개의 테스트 패턴이 필요하며, 5,000개의 테스트 패턴을 사용하면 95.29 %의 고장 검출률을 낸다. 한편, 2번에 걸쳐서 seed를 갱신하는 경우에는 95%의 고장 검출률에 도달하는데 1,544개의 테스트 패턴이 필요하며, 5,000개의 테스트 패턴을 사용하면 96.34 %의 고장 검출률을 높일 수 있었다.

BIST를 삽입함으로써 야기되는 ALU의 면적 오버

헤드는 11.47 %정도가 되고, 곱셈기도 10 %정도 되는 것을 확인했다. 이러한 오버헤드를 줄이기 위해서 본 연구에서는 PRPG에 해당하는 부분을 경계주사 셀로 대체한다. 그림 5와 같은 경계주사 셀의 시프트 레지스터를 이용해서 ALU와 곱셈기 BIST를 위한 PRPG와 MISR를 설계했다. 그림 5의 시프트레지스터들은 BIST모드가 아닐 경우는 정상적인 경계주사 사슬로 동작하며, BIST 모드일 때는 제어신호에 의해서 ALU와 곱셈기를 위한 PRPG, MISR로 동작하게 된다. 경계주사시프터 레지스터를 이용하여 BIST회로를 설계할 경우 4 % 정도의 면적 오버헤더를 갖는다.

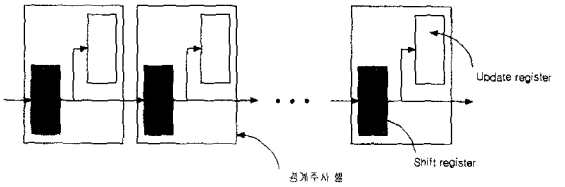


그림 5. 경계주사 사슬의 구조
Fig. 5. Structure of boundary scan chains.

VI. DSP의 동작 검증 및 호환성 검증

1. Co-verifier

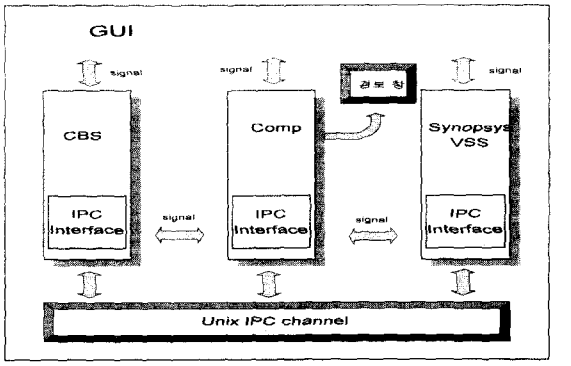


그림 6. Co-verifier의 블록도
Fig. 6. Block diagram of Co-verifier.

DSP의 동작 검증과 호환성 검증을 위해 Co-verifier를 제작[19] 하였다. 그림 6의 Co-verifier는 GUI(Graphic User Interface) 환경을 가지며, C 언어로 모델링된 CBS와 SYNOPSYS 툴의 VSS와 비교기와 IPC 채널로 구성된다. CBS와 VSS와 비교기는 통신을 위해서 IPC(Inter Process Communication)[20] 채널에 연결되었다. VSS[21,22]에서 C언어로 구성된 Co-verifier를 호출하기 위해서 CLI(C Language Interface)[22]을 이용하였다.

IPC의 통신방식에는 메세지 큐, 공유 메모리, 세마포어의 세가지방식이 있으나 사용하기 편하고 대부분

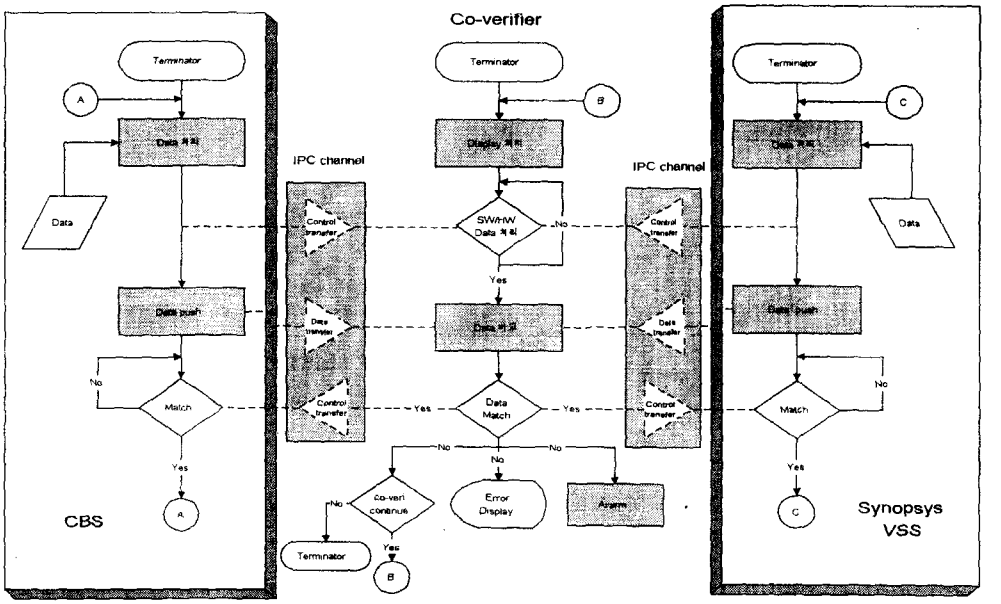


그림 7. Co-verifier의 동기 과정 흐름도
Fig. 7. Syncronization flow of co-verifier.

의 하드웨어에서 지원하는 메세지 큐방식을 사용한다. 그리고 GUI는 C언어와 MOTIF로 구현되었고 프로세서와의 통신을 위하여 signal[20] 방식을 사용하였다.

CBS와 VSS의 정확한 데이타를 실시간으로 비교하기 위해서는 CBS와 VSS의 수행동작을 동기 시켜야 한다.

그림 7은 Co-verifier의 동기과정 흐름도를 나타낸다. CBS와 VSS를 서로 독립적으로 실행시키면서 동일한 테스트 벡터를 입력하고 결과가 나왔을 경우 데이타 처리가 끝났다는 사실을 signal을 통하여 비교기에게 알리고 Unix IPC 채널을 통하여 그 결과 데이터를 메시지 큐에 저장한다. 이 때 비교기에서는 두 모델에서 데이타 처리가 끝났다는 signal이 올 때까지 기다려서 메세지 큐에서 데이타를 읽어 서로 비교한다. 만약 두 결과 데이타가 일치할 경우는 각 모델에 signal을 보내어 다음 동작을 실행하고 일치하지 않으면 경보음과 함께 경고 메시지 창을 화면에 출력한다. Co-verifier의 실행 결과를 그림 8에 나타내었다.

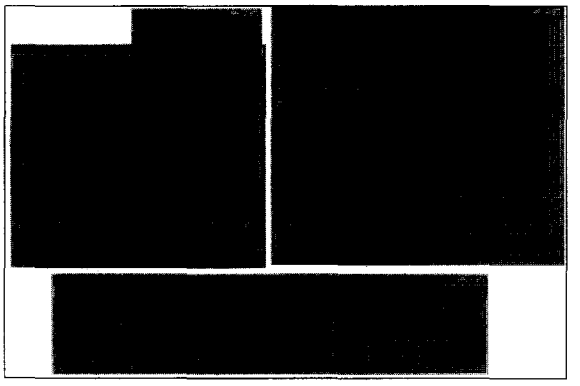


그림 8. Co-verifier를 실행한 화면
Fig. 8. Screen of Co-verifier.

설계자가 눈으로 검증하는 수동적인 검증 방법과 유닉스의 diff 명령어를 이용하는 검증 방법과 Co-verifier를 이용한 검증 방법의 소요시간을 곱셈기의 2538개로 비교하였다. VSS의 출력 파형을 눈으로 확인할 경우 약 1시간 정도 소요되었으며, CBS와 VSS의 출력 파일을 Unix 명령어 diff를 이용할 경우 26분 정도 소요되었다. Co-verifier를 이용한 검증은 5분 정도 소요되어 검증시간에 절약할 수 있었다.

2. 하드웨어 에뮬레이션[23]

VHDL 코드를 이용한 논리 시뮬레이션으로 설계된 DSP의 기능을 검증하기 위해서는 복잡한 알고리즘을 통하여 처리된 많은 샘플 데이타를 분석해야 한다. 본 연구에서는 DSP의 종합적인 기능 검증을 위하여 음성속도 변환을 위한 SOLA(Synchronize Overlap Adding)[24,25] 알고리즘을 수행하였다. 44.1KHz의 샘플링 주파수를 가지는 스테레오 음성 데이타를 10초간 SOLA 알고리즘을 수행하기 위해서는 약 400×106개의 수행 사이클을 필요로 한다. 논리 시뮬레이션을 통하여 10초간의 SOLA 알고리즘 수행 결과 데이타를 얻기 위해서는 SUN-ULTRA1에서 3 주일 이상의 시간이 소요되므로 하드웨어 에뮬레이션을 수행하게 되었다.

하드웨어 에뮬레이션은 하드웨어적으로 DSP를 직접 구현하기 때문에 신호처리 방식이 병렬로 되어 처리속도가 논리 시뮬레이션에 비하여 아주 빠르다.

하드웨어 에뮬레이션을 수행하기 위한 전체 시스템은 그림 9와 같이 FPGA Array Board로 이루어진 Emulator와 Compile 및 P&R 과정을 수행하고 제어할 수 있는 Workstation, Probe 동작을 수행 하는 HP 논리 분석기로 구성된다. 설계될 DSP의 주변 응용회로를 포함한 Target Board를 만들어 하드웨어 에뮬레이터와 데이타 케이블과 클럭 케이블로 연결한다. Workstation과 논리 분석기 간의 통신은 이더넷으로 이루어지며, 트리거 신호를 Workstation에서 논리 분석기로 보내고, Probe된 데이타는 논리 분석기에서 Workstation으로 업로드 된다.

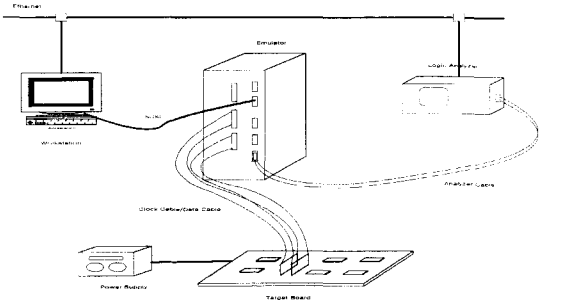


그림 9. Hardware Emulation 시스템
Fig. 9. Systems for Hardware emulation.

이러한 시스템을 이용하여 명령어별 테스트 및 응용알고리즘[5~7]을 수행하여 DSP 에뮬레이션 결과와 직접 비교하였다. 하드웨어 에뮬레이션에 사용된 SOLA 알고리즘의 입력 음성 데이타 양이 많으므로 DSP의 내부에 전체 음성 데이타를 저장할 수 없으며, 하드웨어 에뮬레이터의 데이타 출력도 실시간이 아니므로

PC를 통해 입력과 출력 데이타를 관리하여 동작을 검증하였다. PC와의 Interface는 ISA 방식의 8255 칩과 FIFO chip 으로 구성하여 그 결과를 PC에서 확인 할 수 있었다.

그림 10(a)는 C 프로그램으로 음성 속도 변환 알고리즘을 수행했을 경우의 파형을 나타낸 것이며, 그림 10(b)는 동일한 알고리즘을 하드웨어 에뮬레이션한 결과를 나타낸 것이다. C 프로그램과 DSP의 두 데이타를 비교하였을 때 최하위 비트의 반올림 범위를 벗어나지 않았으며 스피커를 통한 청음 시에 동일한 음질의 사운드임을 확인하였다.

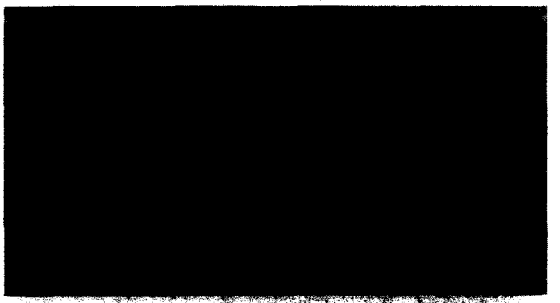


그림 10(a). SOLA 알고리즘의 C 프로그램 실행 파형
Fig. 10(a). Waveform of SOLA by C-programm.

그림 10(b). SOLA 알고리즘 하드웨어 에뮬레이션 파형
Fig. 10(b). Waveform of SOLA by hardware emulation.

3. 호환성 검증[1, 4, 11, 13, 15]

본 논문에서 목표 DSP인 TMS320C30와 호환성 검증을 위하여 모든 명령어의 수행 결과를 상용 시뮬레이터의 결과와 비교하여 개별 명령어 호환 검증을 수행하였다. 개별 명령어의 시뮬레이션 결과만으로 DSP의 완벽한 호환성을 검증할 수 없으므로 설계된 DSP로 표 2에 표현한 다양한 응용 프로그램을 수행하여 상용 시뮬레이터의 수행 결과가 완전히 일치함을 검증하였다. 설계된 DSP를 대상으로 응용프로그램 수행 여부를 검증하는 이유는 설계된 DSP를 실제로 이용할 때, 하나의 명령어가 어떻게 수행되느냐 하는 것보다는 어떤 의미 있는 알고리듬을 구현하고자 하는 것이 목적이기 때문이다. 따라서 다양한 알고리듬에 대해서 검증을 함으로써 설계된 DSP에 대해 더 높은 신뢰를 얻을 수 있고, 성능을 평가받을 수 있는 것이다. 표 2에서 검증용으로 사용된 응용프로그램들의 수행 결과를 분석하면 시뮬레이션에서 수행된 명령어는 총 114종의 명령어 중에서 77종(68%의 명령어 사용률)으로 일반적인 프로세서의 프로그램에 사용되는 통계적인 명령어 사용률[11,13] 25%의 2.5배로 월등히 많았다.

표 2. 호환성 검증용 응용프로그램
Table 2. Application program used to verify compatibility of DSP.

구분	프로그램 실행 사이클 수	프로그램 길이
u-law	350	180
A-law	355	180
FIR	3,120	240
IIR1	1,100	240
IIR2	680	300
LMS	920	220
FFT 64point	3,500	320
ADPCM	26,335	1,800
Viterbi enc/dec	30,300	1,000
SUM	66,660	4,480

Ⅷ. 칩제작 및 칩 테스트

칩 제작을 위한 VHDL 코드의 합성은 ISM(Input Slope Model) 기법으로 만들어진 0.6㎛ Standard Cell Library를 이용하였다.

Back-end 작업과 칩 테스트에 필요한 테스트 벡터는 응용프로그램을 통하여 검증된 DSP를 하드웨어 에뮬레이터에서 수행시켜 결과를 저장하고, 하드웨어 에뮬레이터로부터 만들어진 명령어 코드를 변형하여 생성하였다.

1. 칩 제작

설계된 DSP를 AVANT tool의 asic synthesizer와 메모리 compiler로 게이트 수준으로 합성한 후, 회로의 전체 게이트 수와 SSO(Simultaneously Switching Output)를 고려하여 internal/external power를 계산

한다. 총 게이트 수는 RAM, ROM 모듈을 제외하고 약 120,879게이트였으며, ROM과 RAM을 포함하면 약 45만 게이트였다. 총 45만 게이트의 복잡도를 가지는 회로를 0.6㎛ Cell Based 공정에서 칩 제작하기 위한 원판의 크기는 9×9 ㎟였으며 Package는 174 pin의 PQFP를 사용하였다.

그림 11에서 P&R을 수행한 회로도를 표현하였다. 칩의 내부를 살펴보면 왼쪽/아래 부분이 ALU 블럭, 중간/아래부분이 Multiplier 블럭, 오른쪽에 하나의 segment가 메모리 블럭이다. 그리고 ALU, Multiplier, 메모리 블럭을 제외한 나머지 부분은 왼쪽/윗부분에 flatten되어 있다. 메모리 블럭은 int_ram, int_rom, ext_ram, ext_rom 블럭으로 구성되며, DSP의 외부 메모리 수행 동작 검증을 위하여 칩의 내부에 포함 시켰다.

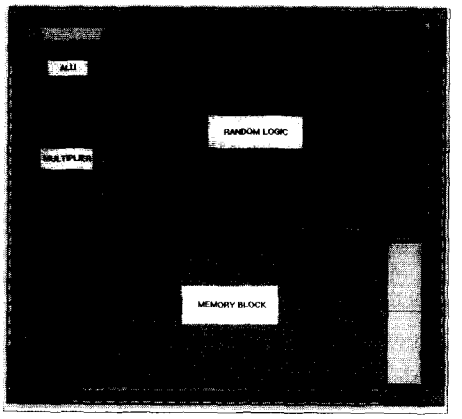


그림 11. 설계된 DSP의 배선 및 배치도
Fig. 11. Layout of designed DSP.

P&R 동작을 수행하고, 원판에서 구현된 배치와 배선의 실제 R,C 값을 추출한 netlist를 이용하여 시뮬레이션을 수행하였다. Post-simulation 은 하드웨어 에뮬레이션에서 제작된 테스트 벡터를 이용하여 회로가 정상적으로 동작함을 확인하였다.

2. 칩 테스트[19]

설계와 검증이 완료된 TMS320C30 호환 DSP는 0.6 μm CBIC 공정 기술을 이용해서 제작되었고, 176 핀 PQFP 타입으로 packaging되었다. 176핀 중에서 150핀 정도는 실제로 DSP에서 function을 위해서 사용되며, 나머지 핀들은 VCC 및 GND 그리고 테스트용으로 사용이 된다.

칩 테스트는 그림 12의 테스트용 보드를 이용하여 회로의 동작검증과 호환성 검증에 사용된 다양한 알고리즘을 칩으로 수행시키고 출력결과를 Tektronix사의 TLA 510 Logic analyzer를 사용하여 각 pin들에 대해서 200 ns마다 probing하여 테스트를 수행하였다.

또한 칩의 기능동작 뿐만 아니라 칩의 전기적인 특성, 물리적인 특성 및 동작 특성분석을 위해 TRILLIUM이라는 Logic Tester 장비를 이용하여 다양한 테스트를 수행하였다. Logic Tester에서 DC 테스트, Timing 및 Level의 noise margin 특성, 속도 시험 등을 수행하여 제작된 칩이 20 MIPS로 175 ㎽(40MHZ, 3.65V)의 전력을 소비하는 것으로 나타났다. Logic tester에 의해 측정된 주요 특성을 요약하면 표 3과 같다.

제작된 칩을 이용하여 시뮬레이션 검증에서 명령어 동작 검증과 응용 알고리즘을 수행시켜 칩이 정상적으로 동작함을 확인하였다.

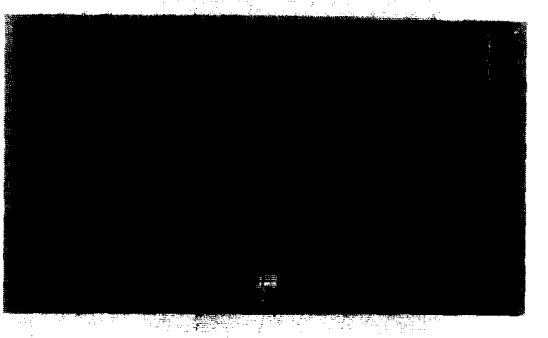


그림 12. 칩 테스트용 보드
Fig. 12. The Board for chip test.

표 3. Logic tester의 측정 분석
Table 3. The measurement of logic tester.

구 분	Min	Max	동작속도
open test	-0.56519V	-0.52569V	40MHz
static current	0.0000A	5.43mA	40MHz
dynamic current	45.41mA	45.41mA	40MHz
leakage current	-10.000uA	0.008uA	40MHz
Time margin	125.000ns	131.943ns	40MHz
Level margin	1.500v	2.980v	40MHz
speed	4MHz	45MHz	-

Ⅸ. 결 론

본 논문에서는 디지털 신호처리의 응용분야에서 많

이 사용되고 있는 TMS320C30 칩과 호환성을 갖는 DSP의 설계 및 칩 구현을 목표로 하였다. 이를 위하여 먼저 메뉴얼, 상용 시뮬레이터를 통하여 DSP 동작을 이해하고 제어신호, 레지스터, 버스, 메모리 값들에 대한 정보를 제공해주는 CBS를 제작하였다. 제작된 CBS를 기준으로 기능블럭과 제어블럭을 설계하였으며, 칩 제작후의 칩 테스트를 위하여 DFT를 삽입하였다. DFT의 삽입은 많은 오버헤드를 요구하므로 본 고에서는 경계주사 셀을 이용한 BIST 회로를 삽입하여 오버헤드를 4% 이하로 낮출 수 있었다. 설계된 DSP가 코어로 활용되기 위해서는 호환성이 보장되어야 하므로 호환성 검증에 많은 노력을 기울였다. 호환성을 검증하기 위해 114개 명령어별 시뮬레이션, ADPCM, Viterbi Encode/Decode, SOLA 등의 13종의 응용프로그램에 대한 시뮬레이션을 수행하였고, 고속의 시뮬레이션 검증을 위하여 Co-verifier를 제작하였다. 응용프로그램을 통하여 검증된 DSP를 IKOS사의 하드웨어 에뮬레이터에서 수행시켜 결과를 저장한 후에 CBS, 하드웨어 에뮬레이터, back-end, 칩 테스트에 사용될 수 있는 테스트 벡터로 변형하여 DSP의 설계 및 검증에 일관성을 유지하였다.

설계된 DSP의 호환성과 신뢰성을 높이기 위하여 음성속도 가변을 위한 SOLA 알고리즘을 하드웨어 에뮬레이션하여 PC에서 청음 테스트를 수행하였다. 청음 테스트의 결과는 C 프로그램으로 처리된 내용과 하드웨어 에뮬레이션 결과가 동일한 음질을 가진 사운드임을 확인할 수 있었다.

칩 제작은 0.6㎛ CBIC 라이브러리를 이용하여 Backend 작업을 수행하였으며 post-simulation에 사용된 테스트 벡터는 하드웨어 에뮬레이션 결과를 이용하였다. 설계된 DSP는 내부메모리를 포함하여 총 45만 게이트를 해당하며 0.6㎛ Cell Based 공정에서 9 X 9㎟ 원판과 174 pin의 PQFP로 제작되었다. 제작된 칩의 성능은 20 MIPS, 175 ㎽(40MHZ, 3,65V)의 소비전력으로 Trillium Logic Tester 장비 상에서 측정되었다.

본 논문에서 언급된 호환 DSP 칩의 설계 및 검증 과정은 호환 프로세서 칩 개발에 대한 시스템적인 접근 방법을 제공하며 명령어 기반의 프로세서 개발에 응용될 수 있는 중요한 기법들을 포함하고 있어서 비슷한 분야의 연구에 도움이 되리라 본다.

X. 참 고 문 헌

[1] "TMS320C3X User's Guide", Texas Instruments, 1994.

[2] "TMS320 Floating-Point DSP Assembly Language Tools User's Guide", Texas Instruments, 1994.

[3] Kai Hwang, "Advanced computer architecture", McGraw Hill, 1995

[4] Rozenblit, "Codesign: Computer-Aided Software/Hardware Engineering", IEEE press, 1995.

[5] 우종식, 박주성외, "32비트 부동소수점 DSP의 Cycle Based Simulator에 관한 연구" 대한전자공학회, 1998. 12

[6] Oppenheim, "Discrete-Time signal processing", Prentice-Hall, 1989.

[7] Sklar, "Digital Communications fundamentals and applications", Prentice-hall, 1988.

[8] Rulph Chassaing, "Digital Signal Processing with C and the TMS320C30", JOHN WILEY & SONS, 1992.

[9] "TMS320C3XC Source Debugger User's Guide", Texas Instruments, 1994

[11] Nikitas Alexandridis, "Microprocessor Based Systems", 1993.

[10] TMS320 Floaing-Point DSP Optimizing C Compiler User's Guide, Texas Instrument, 1999.

[12] D. A. Patterson & J. L. Hennessy, "Computer organization & Design, The hardware/software interface", Morgan Kaufman, 1994.

[13] J. L. Hennessy, "Computer design and construction", Morgan Kaufman, 1995.

[14] 홍성제 외 공저, "테스팅 및 테스팅을 고려한 설계", 홍릉과학출판사, 1998

[15] Test Technology standards committee, "IEEE standard Test Acess Port and Boundary-Scan Architecture", IEEE computer society press, 1990.

[16] D. K. Bhavsar & R. W. Heckelman, "Self

Testing by polynominal Division", Proc. of 1981 IEEE Test conf., pp. 208-216, 1981.
[17] P. H. Bardell & W.H. Moanney & Jacob Savir "Built-In test for VLSI:Pseudorandom Techniques", International Business machines corp., 1997.
[18] Test Compiler, Synopsys, 1997
[19] 부산대, "DSP MACRO 라이브러리 개발에 관한연구 연구보고서", 산자부, 정통부, 과기부, 1998-2000
[20] "UNIX 시스템 프로그래밍", 홍릉과학출판사, 1991
[21] VSS User's Guide, Synopsys, 1997.
[22] VSS Interface , Synopsys, 1997.
[23] "VirtuaLogic Emulation System Version 2.0", IKOS system Inc, 1999.
[24] N. R. Portnoff, "Time scale modification of speech based on short time Fourier analysis", IEEE. Trans.Acoustics. Speech Signal Process, Vol. 29, 1981.
[25] T. F. Quateri & R. J. Mcaulay "Shape invariant time scale & peak modification of speech", IEEE. Trans. Acoustics. Speech Signal Process, Vol. 40, 1992.

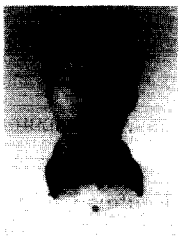
저 자 소 개

禹 鍾 植(正會員)

1968년 10월 21일 부산 출생. 1993년 2월 부산대학교 전자공학과 졸업(학사). 1995년 2월 부산대학교 전자공학과 졸업(석사). 1997년 3월~현재 부산대학교 전자공학과 박사과정. 1997년 10월~현재 (주)보이소반도체 대표이사. 주관심분야 : DSP 설계, DSP Application, ASIC 설계, 영상 신호처리 및 구현

徐 鎭 瑾(正會員)

1971년 4월 25일 경남 울산 출생. 1997년 2월 부산대학교 전자공학과 졸업(공학사). 2000년 2월 부산대학교 전자공학과 졸업(공학석사). 1997년 11월~2000년 7월 부산대학교 반도체 설계 교육센터 연구원주관심분야 : VLSI 설계, DSP Application



林 在 英(正會員)

1975년 10월 1일 : 경남 함양 출생. 1998년 2월 부산대학교 전자공학과 졸업(공학사). 2000년 2월 부산대학교 전자공학과 졸업(공학석사). 2000년 3월~현재 (주)보이소반도체 기술연구소 연구원. 주관심분야 : VLSI 설계, DSP Application

朴 柱 成(正會員)

1953년 12월 19일 경남 진주 출생. 1976년 : 부산대학교 전자공학과 졸업(학사). 1978년 KAIST 전기 및 전자공학과 졸업(석사). 1989년 Univ. of Florida EE (Ph. D.). 1978년 3월~1991년 2월 ETRI 반도체연구단 직접회로 연구부 실장, 연구위원 역임. 1991년 3월~현재 부산대학교 전자공학과 부교수, 부산대학교 컴퓨터 및 정보통신연구소 연구원. 1997년 11월~현재 부산대학교 반도체 설계 교육센터장. 2000년 6월~현재 부산대학교 전자공학과 학과장. 주관심분야 : DSP 설계, ASIC 설계, 음성/사운드 신호처리 및 구현