論文2002-39SD-2-9

고성능 32-bit DSP 코프로세서의 아키텍쳐 개발

(Development of a High-performance DSP Coprocessor Architecture)

尹 晟 瞰 *, 金 相 郁 *, 裵 晟 日 *, 姜 成 昊 *, 金 容 天 **,
鄭 勝 在 **, 金 相 佑 **, 文 相 焄 **

(Sungchul Yoon, Sungil Bae, Sangwook Kim, Sungho Kang, Yonchun Kim, Seungjae Chung, Sangwoo Kim, and Sanghun Moon)

요 약

이 논문은 저전력 마이크로 컨트롤러의 coprocessor로 동작하는 고성능 DSP의 아키텍쳐 구조를 제안한다. 제안된 DSP 아키텍쳐는 DSP 응용 분야의 기본 수식인 곱의 합을 고속으로 수행할 수 있도록 MAC(Multiply and ACcumulate) 유닛 두 개를 갖는 dual MAC 아키텍쳐 구조이면서, 곱셈기와 덧셈기를 병렬적으로 배치시킨 특징을 갖는다. 그리고 한번에 최대 3개의 명령어를 동시에 수행할 수 있으면서도 명령어 길이는 31 비트로 고정된 3웨이 수퍼스칼라 구조를 갖는다. 현재 상용되고 있는 세 개의 DSP들과의 벤치마크 결과, 제안된 DSP 구조가 가장 좋은 성능을 보여주었다. 또한, 특정 알고리듬에 대해서 성능이 같아도 메모리 사용량에 있어 효율적인 구조라는 것을 보여준다.

Abstract

A new high-performance DSP architecture is proposed, which behaves as a coprocessor of a 32 bit microcontroller. Because the proposed DSP architecture is a dual MAC(Multiply and ACcumulate) DSP architecture, it can process efficiently a number of SOP(sum of product) operations used in many DSP applications. In order to efficiently perform other operations such as pure additions without any restriction, a MAC is composed of a multiplier and a ALU placed in parallel. In addition, it is a 3-way superscalar architecture, which can issue 3 instructions at a time. The benchmark results with 3 other dual MAC DSPs show that the proposed DSP has the best performance. Futhermore, it is proven that the proposed DSP is more efficient in memory usage, although the performance is comparable in some algorithms such as Viterbi decoding and FFT butterfly.

I. 서 론

내장된 프로세서 코어들의 저전력, 고성능에 대한 필요성은 최근 급격히 증가하고 있는 실정이다. 이것은 이동통신 분야의 급속한 발전으로 인하여, 실시간으로 데이터를 처리하면서 전력을 적게 소모하는 프로세서가 더욱 요구되기 때문이다. 여기에 시장진입시간을 줄이기 위해 시스템온칩에 대한 관심이 높아지면서 여러

* 正會員, 延世大學校 電氣電子工學科
(Dept. of Electrical & Electronic Eng., Yonsei University)
** 正會員, 三星電子 CalmRISC 팀
(Samsung Electronic CalmRISC Team)
接受日字:2001年11月8日, 수정완료일:2001年12月24日

가지 프로세서를 서로 합쳐서 하나의 칩안에 넣기 위한 연구도 활발히 진행중이다[1].

이와 같은 요구에 부응하여 저전력 마이크로 컨트롤러에 FPU나 멀티미디어 칩, 그리고 DSP(Digital Signal Processor) 등을 코프로세서로 부착하여 성능을 높이고, 시스템온칩을 실현하는 연구가 활발히 진행 중이다. 디지털 신호 처리는 멀티미디어적인 요소로 인한 데이터길이의 확장과 고속의 데이터 전송의 필요성에 의해 보다 큰 데이터들을 고속으로 처리, 전송해야 하는 요구가 커지고 있다. 그리고 이 또한 이동성을 고려해서 저전력을 고려한 설계가 뒤따라야 하는 것이 추세이다. 저전력을 위해서는 디자인 흐름상의 각 레벨에서 심각하게 고려를 해야 한다. 특히 프로세서 설계에 있어서는, 상위 레벨에서는 명령어 fetch scheme과, 효율적인 버스 사용 등의 상세한 spec. 정의를 통해서 저전력을 위한 아키텍쳐를 구성하는 것부터 하위 레벨에서의 셀 구성 방법 등으로 인해 저전력을 실현하는 것까지 모든 디자인 플로우에서 세심하게 고려를 해야 저전력이 실현되는 것이다.

시스템온칩 접근법은 ASIC 디자이너들에게는 비용, 전력소모, 그리고 시스템 디자인 복잡성 등의 문제 때문에 상당히 매력적인 방법이다. 시스템온칩 방법 중 하나는 마이크로 컨트롤러를 두고 여러가지 응용분야에 따라 그에 해당하는 칩들을 하나의 다이(die)에 구현하는 것이다. 이 방법은 해당 응용에 대한 상당한 유동성을 가지는 것이 장점이지만, 각 칩들간의 통신으로 인한 오버헤드 등이 문제가 될 수 있다. 그리고 또 다른 방법은 한 칩에 응용 분야에 해당하는 명령어 셋들을 추가하고 그에 해당하는 데이터 패스만을 추가하는 방법이 있다. 이 방법은 앞의 방법과는 반대로 각 칩간의 통신 등으로 생기는 오버헤드는 사라지지만, 응용에 따른 유동성 면에서는 앞의 방법보다는 떨어진다는 단점을 가지고 있다.

이 논문에서는 기존 DSP의 성능 저하 요인을 분석하고[2-5], 저전력, 고성능 DSP 아키텍쳐를 소개하고, 그것의 성능 검증을 위해 여러가지 상용 DSP들과의 벤치마크를 수행한 결과를 보였다. 그리고, 32비트 저전력 마이크로 컨트롤러와의 인터페이스 방법에 관하여 소개하였다. 2장에서는 기존 DSP 아키텍쳐의 종류 및 단점을 논하였고, 3장에서는 제안된 DSP 아키텍쳐의 구조와 명령어 집합을, 그리고 4장에서는 다른 세 개의 dual MAC DSP들과의 성능 평가에 대한 논의를 하였고, 5장에서는 32 비트 마이크로 컨트롤러와의 인터페이스 방법을 설명하였으며, 6장에서 결론을 논하였다.

Ⅱ. 기존 DSP 아키텍쳐 구조

마이크로 프로세서와 DSP를 구분하는 기준이 주 응용 분야이다. 다시 말해서, 마이크로 프로세서의 경우에는 주 응용 분야가 여러 가지 범용 프로그램을 사용하기 위한 연산을 주로 하는 반면, DSP의 경우에는 유무선 통신이나 원하는 모양으로 디지털 신호를 처리하는데 주로 사용되며, 특히 DSP는 실시간으로 데이터를 처리해야 하는 분야에 주로 쓰인다.

그리고, DSP의 주요 응용 분야인 필터 구현이나 FFT와 같은 알고리듬들의 기본이 되는 수식이 곱의 합(sum of product)이므로, 곱셈기와 덧셈기를 직렬적으로 연결한 MAC 유닛을 기본 자원으로 사용한다. 또한, 빠른 데이터의 연산을 위해서 데이터와 프로그램을 분리시킨 하바드 아키텍쳐를 쓰는 것이 일반적이며, 한번에 여러개의 명령어를 수행하는 멀티-이슈 아키텍쳐를 갖는 DSP도 많이 연구되고 있다. 멀티-이슈 아키텍쳐를 갖는 DSP는 연산을 위한 데이터 패스와 메모리 연산을 위한 주소를 생성하는 부분이 독립적으로 존재한다. 한번에 수행할 수 있는 곱의 합의 개수를 늘림으로써 성능을 향상시키기 위해 MAC 유닛을 두 개 갖는 dual MAC DSP도 많이 연구가 진행된 상황이다. 기존의 dual MAC DSP 아키텍쳐들은 그림 1의 DSP16000[3-4]의 경우에서 보듯이 대부분 곱의 합 수식의 빠른 처리를 위해 곱셈기와 ALU를 직렬로 배치하였다. 그러나 이 구조는 곱의 합 수식의 처리가 아닌, 단순한 덧셈이나 ALU 연산과 같은 연산에서는 멀티-이슈 프로세서로서의 조건으로는 명령어 실행의 병렬성에 상당한 제약 조건으로 작용한다.

누적기의 구조에서도 대부분의 DSP들은 곱셈기 입출력 레지스터와 누적기를 분리해 놓은 구조를 갖는다[2-5]. 이런 구조는 세 항 이상의 곱셈 연산에서는 별도의 이동 명령이 필요해서 한 싸이클이 소요된다.

dual MAC DSP들보다 성능을 높이기 위해 연산 자원들의 개수를 늘린 VLIW 구조에 대한 연구도 많이 진행되었는데, 그 예가 TI사의 62x 구조이다[6-7]. 그러나 이 구조는 보통의 VLIW 아키텍쳐와 같이 한번에 수행하는 명령어의 길이가 늘어남에 따라, 컴파일러가 복잡해지고, 프로그램의 길이가 증가하게 되는 단점이

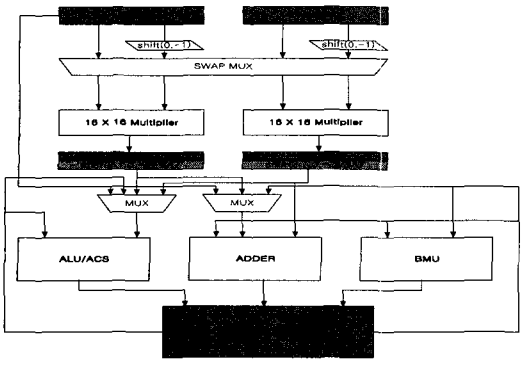


그림 1. 곱셈기와 덧셈기가 직렬로 배치된 예
Fig. 1. The example of serial placement of multipliers and adders.

있다. 프로그램의 길이가 증가하면 자연스럽게 프로그램 메모리를 사용하는 양이 많아지므로 메모리 사용에 있어서 비효율적이다.

Ⅲ. 새로운 DSP 아키텍쳐 구조 및 명령어 집합

1. 전체적인 아키텍쳐 구조

제안된 DSP는 두개의 MAC구조를 갖고, 3-way 수퍼스칼라 구조를 갖는다. 명령어나 데이터는 32비트를 기본으로 하며, 버스도 기본적으로 32비트이다. 그림 2는 제안된 DSP 구조의 전체적인 아키텍쳐 모습을 나타낸 것이다. 실제 연산에 사용되는 DALU(Data Arithmetic Logical Unit), 명령어를 받는 Loopbuffer, 그리고 메모리 연산을 위한 주소를 생성하는 RPU (Ram Pointer Unit)으로 크게 분류된다. DALU는 일반적인 DSP와 마찬가지로 실제 데이터를 처리하는 부분이므로, DSP 성능에 가장 큰 영향을 미치는 부분이라 할 수 있다. 그래서 DALU는 곱셈기 두 개와 40비트 ALU 하나, 그리고 캐리를 사용해 32비트 덧셈을 할 수 있는 두 개의 16bit ALU, 마지막으로 비트별 논리연산을 수행하는 BMU로 구성되어 있다.

dual MAC DSP이므로 메모리에서 데이터를 2개의 워드씩 가져와야 한다. 그렇기 때문에 데이터의 주소를 생성하는 RPU는 두 영역으로 나뉘어져 있다. 바로 X 영역과 Y영역이 그것이다. RPU와 DALU는 각각 병렬적으로 동작할 수 있기 때문에 제어를 위한 제어 레지스터, 그리고 상태를 위한 상태 레지스터가 따로 있다. 루프 버퍼는 코어로부터 받은 명령어를 디코더 단계로

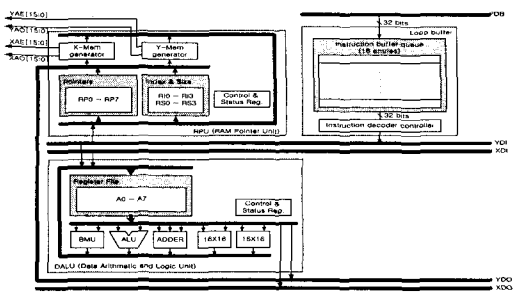


그림 2. 제안된 DSP의 전체적인 블록다이어그램
Fig. 2. Block diagram of the proposed DSP.

넘겨주는 기능과, 루프 명령어가 들어오면, 그 뒤의 명령어를 판단해서 반복 여부를 결정하는 역할을 한다. 16개의 명령어를 저장할 수 있으며 세 단계의 루프를 허용한다. 그러나 복잡성을 피하고 제어로 인한 전력 소모를 적게 하기 위해 하위 단계의 루프는 하나의 명령어만을 반복할 수 있다.

그림 3은 제안된 DSP의 누적기 및 레지스터의 구조를 나타낸 그림이다. 그림에서 보듯이 8개의 누적기를 기본으로 갖고, 곱셈기의 출력 레지스터도 누적기에 포함시켰다. 그리고, 곱셈기의 입력 레지스터와 다른 누적기와 공유되어 있는 것을 볼 수 있는데, 이것은 연속적인 곱셈에서의 성능 강화를 위한 구조이다. 그리고, 40비트 누적기가 10개가 있으므로, 택할 수 있는 경우의 수가 매우 많다. 즉, 누적기의 확장 부분인 AiG 부분, 상위 부분인 AiH, 그리고 하위 부분인 AiL, 그리고 32비트 부분과 24 비트 부분까지 고려하면 선택할 수 있는 경우의 수가 늘어나므로, 이것을 명령어에 직접 인코딩하려면 오퍼랜드에 해당하는 레지스터들의 인코딩에 각각 7 비트가 인코딩되어야 한다. 이 경우 4개의 오퍼랜드 부분만 추가해도 28 비트가 되어 31 비트 명령어 인코딩은 불가능해진다. 그래서, 모든 레지스터의

39 32	31 16	15 0
A0G	A0H(X0H)	A0L(X0L)
A1G	A1H	A1L
A2G	A2H	A2L
A3G	A3H	A3L
A4G	A4H(X0H)	A4L(X0L)
A5G	A5H(X1H)	A5L(X1L)
A6G	A6H(Y0H)	A6L(Y0L)
A7G	A7H(Y1H)	A7L(Y1L)
P0G	P0H(Y0H)	P0L(Y0L)
P1G	P1H	P1L

그림 3. 제안된 DSP의 누적기 및 레지스터 구조
Fig. 3. Accumulators of the DSP.

선택 범위에 따라 따로 집합을 정해서 명령어에 인코딩하였다. 또한, 곱셈 명령에 필요한 오퍼랜드와 같이 입출력이 정해진 데이터 패스에 대해서는 오퍼랜드 인코딩을 생략하는 방법으로 인코딩할 수 있는 비트 필드를 늘렸다.

그림 3을 보면 곱셈기의 입력 레지스터들과 누적기들이 서로 한 레지스터를 공유하는 것을 볼 수 있다. 그리고 P0 곱셈기 결과 레지스터는 Y0와 공유되어 있고, P1 곱셈기 결과 레지스터의 경우는 분리해 놓은 것도 볼 수 있다. 이것은 DSP의 응용분야에서 필터연산 등의 연산에서 두항 이상의 곱셈을 수행하는 경우가 상당히 많은데 이런 경우에 성능향상을 위해 곱셈기의 입력 레지스터와 출력 레지스터 중 하나씩을 공유한 것이다. 그림 4는 이러한 예를 나타낸 그림이다. 기존의 구조를 사용할 때에는 곱셈기 입력 레지스터와 누적기가 분리되어 있었기 때문에 곱셈기의 출력값을 다시 곱셈기의 입력 레지스터로 이동 명령을 사용해서 옮기는데 한 싸이클의 소요가 있었지만, 이 구조를 사용할 때에는 그럴 필요가 없어지기 때문에 위와 같은 연산이 나올 때마다 한 싸이클씩 절약하게 된다. 물론, 곱셈기와 덧셈기를 병렬로 연결함으로써 경우에 따른 데이터의 선택을 위한 별도의 하드웨어가 첨가되어야 하므로 다수의 MUX가 추가되는 문제가 있다.

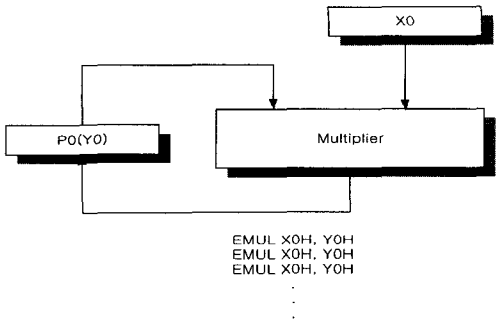


그림 4. 반복곱셈에 유리한 곱셈기의 구조
Fig. 4. Efficient multiplication repetition.

그런데 이와 같은 연산이 응용분야에서 상당한 부분을 차지하므로, 그 이득은 매우 크다. 예를 들면, LMS (least mean square) 필터에서 3항 곱셈이 필요하고, 초월함수 구현 시에 다항식 전개방식으로 구현을 하면 곱셈을 하고, 덧셈을 한 후에 그것을 다시 곱셈을 하는데 이러한 연산들을 수행할 때, 결과 레지스터를 Y0로 쓰면 별도의 이동 명령 없이 효과적으로 연산을 수행할 수 있다. 그러나, 이런 레지스터 공유로 인한 컴파일러를 만드는데 상당한 어려움이 존재할 것으로 예상된다. 파이프라인 된 프로세서에서 다음 명령어와의 데이터 의존성이 있는 것을 체크하는 것은 컴파일러의 역할인데, 이것을 수행하기 위해 컴파일러 제작 시에 더욱 세밀한 작업이 필요하기 때문이다. 그리고 상대적으로 레지스터 자원에 부족 현상이 발생할 수도 있으므로 의존성 체크에 대한, 보다 큰 노력을 기울여야 한다. 제안된 DSP는 고성능이면서, 저전력을 소모하는 현 추세에 맞추기 위해 아키텍처 정립에서부터 구현에 이르기까지 모든 설계 과정에서 고려하였다.

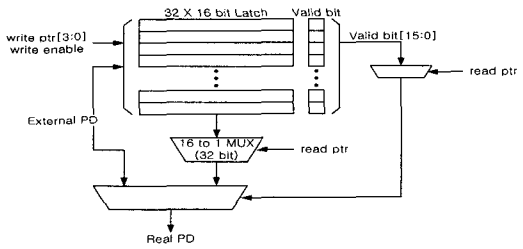


그림 5. 루프버퍼의 블록다이어그램
Fig. 5. Block diagram of loop buffer.

우선, 아키텍처면에서 살펴보면, 명령어를 프로그램 메모리에서 가져오는 횟수를 줄이기 위해 루프명령에 대해서 버퍼를 두었다. 제안된 DSP는 마이크로 컨트롤러의 coprocessor이므로 명령어는 마이크로 컨트롤러를 통해서 전달받는 형식을 취한다. 그러므로, 명령어를 가져오는 횟수를 줄이는 것은 DSP뿐만 아니라, 마이크로 컨트롤러의 전력소모에도 많은 이득을 가져온다. 루프는 세 단계까지 있고, 가장 윗수준 루프는 두개의 루프를 포함할 수 있다. 루프 안에서는 명령어들이 일정한 주기를 갖고 반복된다. 이러한 특성을 살리기 위해 루프 수행시에는 루프버퍼에 저장된 명령어들을 가져옴으로써, 전력 소모를 줄일 수 있다. 그림 5는 루프버퍼의 블록다이어그램을 나타낸 것이다. 그리고 그림 6은 루프버퍼로 명령어를 사용할 때, 늘어난 프로그램 카운터를 알맞게 정정하기 위한 루프버퍼 브랜치 논리회로를 나타낸 것이다. 즉, DSP가 루프버퍼에서 명령어를 가져오는 동안에도 마이크로 컨트롤러의 프로그램 카운터는 계속 증가하고 있는데 루프를 빠져나온 후에는 이 프로그램 카운터는 올바른 값이 아니므로 정정해줄 필요가 있다. 이 때 사용되는 논리회로가 바로 루프버퍼 브랜치 논리회로이다. 이러한 루프 버퍼는 실제적으

로 DSP에게는 명령어 버퍼역할까지 한다. 즉, coprocessor인 DSP는 직접 프로그램 메모리에는 접근을 할 수 없으므로, 코어가 넘겨주는 명령어를 루프 버퍼가 받아서 처리를 하는 구조를 갖는다.

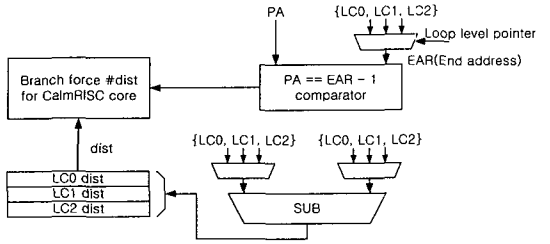


그림 6. 루프 버퍼 브랜치 논리회로
Fig. 6. Loop buffer branch logic.

2. 명령어 집합

명령어 집합은 DSP의 응용 분야에 따라 필요한 연산들을 조사하여 구성하였다. dual MAC DSP에서는 두 개의 워드를 가져와야 하는 경우도 있고, 한 개의 워드를 가져와야 하는 경우도 있으므로, load/store 명령어를 여러 개로 나누었다. 그리고, 해당 레지스터의 종류에 따라서 RPU 레지스터에 하느냐, 아니면 DALU 레지스터에 하느냐에 따라서 각각 분류를 하였다. 그리고 덧셈과 뺄셈, 그리고 곱셈 등의 DSP의 기본 명령들도 여러 개의 부분 집합으로 나누어 구성함으로써, 성능 향상을 꾀하였다. 또한 논리 연산을 위한 명령어도 포함시키고, Viterbi 연산에 필요한 명령어 등 여러 응용 분야에 따른 명령어 집합도 고려하였다. 표 1은 제안된 DSP의 명령어 집합을 나타낸 표이다. 표 1에서 보듯이 매우 다양한 명령어 집합이 있고, 각각 병렬적으로 사용할 수 있는 명령어들은 동시에 수행할 수 있는 구조를 갖는다. 예를 들면, load/store 명령어, ALU 명령어 또는 논리 명령어 등은 같이 수행될 수 있는 것이다.

제안된 DSP의 명령어는 31 비트를 기본으로 한다. 그러나 VLIW 아키텍처처럼 n개의 명령어를 수행할 수 있을 때, 31 × n 비트를 사용하는 구조가 아니라, 31 비트 안에 최대 3개의 명령어를 인코딩한 구조를 갖는다. 그림 7은 이러한 명령어의 인코딩의 예를 나타낸 것이다.

또한, 제안된 DSP 구조가 3-way 수퍼스칼라 구조이기 때문에 최대 3개의 명령어를 31비트 안에 인코딩을 해야 한다. (한 비트는 마이크로 컨트롤러의 명령어와

표 1. 명령어 집합과 mnemonic
Table 1. instruction set and mnemonic.

구분 / 명령어 부분집합	종류	MNEMONIC EXAMPLES
LOAD/ STORE	load/store RPi single load/store dual load/store	ELD dec1, src1 ELD dec1, src1 ELD dec1, src1 ‖ ELD dec2, src2
ALU 명령어	3항 덧셈/뺄셈 2항 덧셈/뺄셈 덧셈/뺄셈 & round 덧셈/뺄셈 & shift dual 덧셈/뺄셈 dual 덧셈/뺄셈 & round 곱셈	EAA(EAS, ESA, ESS) dec1, src1, src2, src3 EAA(EAS, ESA, ESS) dec1, src1, src2 EAAR(EASR, ESAR, ESSR) dec1, src1, src2 EAA(EAS, ESA, ESS) dec1, src1, src2, src3 EAD dec1, src1, src2 ‖ EAD dec2, src3, src4 EADR dec1, src1, src2 ‖ EADR dec2, src3, src4 EMUL X0H, Y0H X0H, Y0L (32가지)
논리 명령어	절대값 negation logical not increment by 1 decrement by 1 round round and not shift minimum maximum Exponent and, or, xor	EABS dec1, src1 ENEG dec1, src1 ENOT dec1, src1 EINC dec1, src1 EDEC dec1, src1 ERND dec1, src1 ENRND dec1, src1 ESLA(ESRA, ESR, ESL) dec1, src1 EMIN dec1, src1 EMAX dec1, src1 EEXP dec1, src1 EAND(EOR, EXOR) dec1, src1, src2
제어 명령어	branch loop	EBRT(EBRF, EBSRT, EBSRF) #target address EREP(EREPS) type, end point, loop count
기타 명령어	insertion extraction	EINSZ(EINS, EINS) dec1, src1, src2, src3, src4 EXTRZ(EXTRS, EXTR) dec1, src1, src2, src3, src4

그림 7. 효과적인 명령어 인코딩의 예
Fig. 7. The example of efficient instruction encoding.

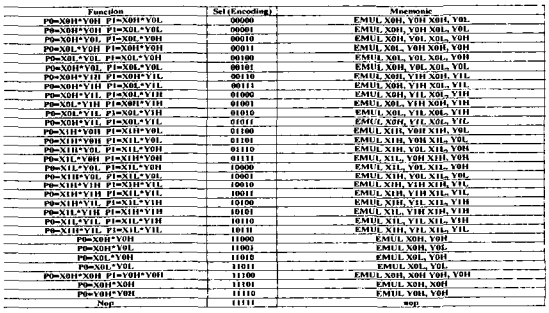


그림 8. 곱셈 명령어의 5 비트 인코딩
Fig. 8. The 5-bit encoding of multiplication instructions.

DSP 명령어를 구분하는데 사용한다.) 이것을 위해 명령어 종류별로 구분하고, 내재된 명령어의 개수에 따라

서 구분하여 효율적인 명령어 인코딩을 수행하였다. 그러나, 31비트 안에 최대 3개의 명령어가 인코딩 되어 있으므로 인해, 이것을 디코딩 하는데 약간의 오버헤드가 있을 수 있다. 그리고 다음절에서 설명할 파이프라인에서 보듯이 디코드 단계가 두 개 필요한 것도 이러한 명령어 인코딩의 특성으로 인한 것이다. 하나의 명령어 안에 곱셈, ALU 연산, 메모리 연산 등의 세 개의 명령어를 포함시킬 수 있다. 그림 7은 명령어 인코딩의 한 예를 보인 것이다. 곱셈 명령은 곱셈기의 입력 레지스터와 출력 레지스터가 두개씩 있고, 곱셈기도 두개가 있기 때문에 레지스터들에 대해서는 직접적인 명령어 인코딩을 할 필요가 없다. 32가지의 다른 곱셈연산을 사용하므로 그것에 대한 인코딩을 5비트로 하였다.

그리고 그림 7에서 Dual Load는 메모리 연산, 즉 load/store에 관련된 인코딩으로써 RPU를 사용하여 수행되는 연산들에 대한 명령어를 나타내었다. 그리고 나머지 비트 필드에는 ALU연산에 관련된 명령어가 인코딩되어 있다. 이와 같이 3-way 수퍼스칼라 명령어를 32비트에 효과적으로 인코딩을 수행하였다. 그림 8에서 보듯이 곱셈에 대한 부분은 5 비트로 인코딩되어 있는데, 이것은 그림 8에서 보는 바와 같이 32가지 곱셈 연산을 수행할 수 있고, 곱셈기의 입력과 출력 레지스터가 정해져 있기 때문에 입출력 레지스터를 명령어 안에 인코딩할 필요가 없다. 그리고, 31 비트 안에 인코딩되는 명령어를 최대한 효율적으로 배치하기 위해 곱셈 명령어들은 5 비트로 표현한 것이다.

3. 제안된 DSP의 파이프라인 구조

앞에서 설명한 3-way 수퍼스칼라 구조는 고성능을 위한 구조이면서도, coprocessor로써의 DSP 입장에서는 최대 3개의 명령어를 한번에 가져올 수 있으므로, 명령어를 가져오는 횟수를 줄임으로써 전력소모와 프로그램 메모리 사용을 줄이는 장점을 갖는다. 실제 설계에서 전력을 많이 소모하는 부분을 살펴보면, 파이프라인을 위한 제어 부분과 데이터 연산을 수행하는 DALU 부분에서 가장 많이 소모한다. 파이프라인에서 전력소모는 브랜치 명령에 의한 파이프라인 지연과 명령어 flush에 따른 것으로, 이 문제를 해결하기 위해 명령어의 조건적 수행(conditional instruction execution)을 도입하였다. 즉, 파이프라인 수행중에 생기는 플래그를 바로 명령어 수행을 위한 조건으로 사용하는 명령어 집합을 도입한 것이다.

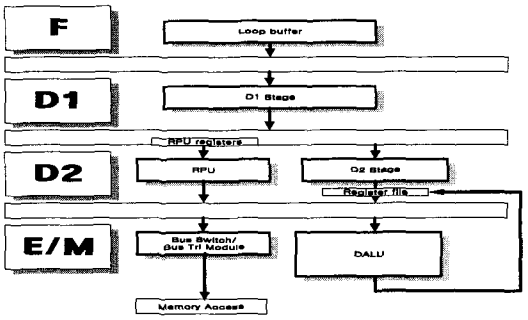


그림 9. 제안된 DSP의 파이프라인 구조
Fig. 9. Pipeline structure of the DSP.

그림 9는 이 DSP의 파이프라인 구조를 나타낸 그림이다. dual MAC구조를 가지면서 3-way 수퍼스칼라 구조를 갖는 DSP이기 때문에 디코드1 단계에서는 명령어를 해석해서 몇 개의 명령어가 들어있는지를 조사하고, 디코드2 단계에서는 명령어 수행에 필요한 각 유닛들의 컨트롤 신호와 레지스터 어드레스를 생성한다. 3-way 수퍼스칼라 구조를 갖기 때문에 메모리 연산을 사용할 때의 주소를 생성하는 역할을 RPU(Ram Pointer Unit)라는 블록으로 사용함으로써 자원문제를 해결하였다. 디코드2 단계에서 RPU를 사용해서 메모리 연산의 주소를 계산함으로써 실행 단계에서 곧바로 연산 동작과 병렬적으로 메모리 연산을 수행할 수 있도록 하였다. 이것은 보통의 아키텍쳐들에서 보는, 메모리 단계와 레지스터 쓰기 단계를 분리하는 일반적인 경향과 다른 구조인데, 메모리 연산 명령어와 ALU 연산 명령어를 동시에 수행할 수 있는 수퍼스칼라 구조에서는 메모리 단계와 레지스터 쓰기 단계를 동시에 수행할 수 있으므로, 두 단계를 합치는 것이 효율적이다. 그리고 각 파이프라인 레지스터에는 파이프라인 실행에 대해서 지연이 발생하는 경우를 위한 신호들이 있고, 이것은 코어에게 알려주기 위한 인터페이스 신호들과 연결되어 지연 발생시 바로 코어에게 알려 코어쪽의 파이프라인도 지연을 시킬 수 있다.

Ⅳ. 마이크로 컨트롤러와의 인터페이스

그림 10은 32 비트 마이크로 컨트롤러와 제안된 DSP의 인터페이스 신호들을 나타낸 것이다. 코어는 COPIR[30:0]을 통해서 DSP의 명령어를 DSP에 전달하고, DSP는 루프 버퍼를 사용해서 그 명령어를 받게 된다. 그리고 코어의 파이프라인에서 지연이 발생하면 각

단계마다 STXEN, STMEN, STWEN 등의 신호가 DSP에 전달되어 DSP의 파이프라인도 지연되게 되고, 반대로 DSP에서 지연이 생길 경우에는 COPXEN, COPMEN, COPWEN 등의 신호가 코어에 전달되어 코어의 파이프라인도 지연되는 것이다. 또한, 예외(exception) 발생시에 코어와 DSP간에 알려주는 신호도 존재한다.

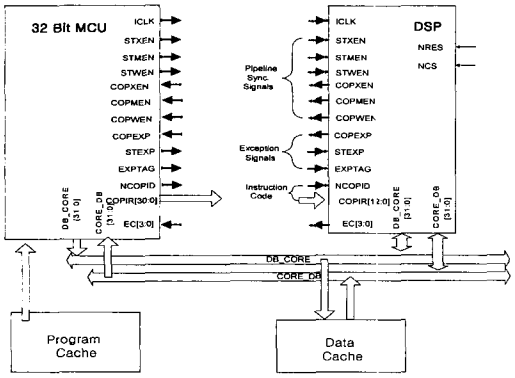


그림 10. 32 비트 마이크로 컨트롤러와 DSP의 인터페이스

Fig. 10. Interface between 32 bit micro-controller and proposed DSP.

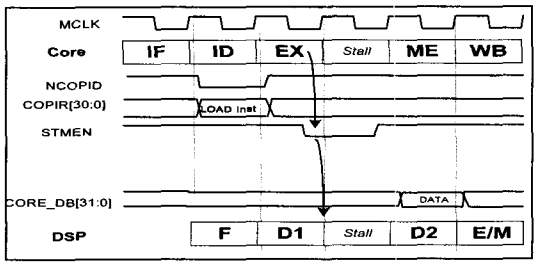


그림 11. CLD 명령어를 사용한 코어와 DSP의 데이터 교환 타이밍도

Fig. 11. Timing Diagram for data exchange between core and DSP using CLD instruction.

제안된 DSP의 코어인 마이크로 컨트롤러의 첫 번째 코프로세서인 FPU와의 인터페이스를 위한 신호들을 대부분 유지하였다. FPU는 기본 명령어가 12비트이고, 16비트까지 확장이 가능한 명령어 인코딩 때문에, COPIR[11:0]와 sysx[3:0]을 두었다. 그래서 12비트 명령어는 COPIR[12:0]을 통해서 코프로세서로 전달하고, 12비트를 넘는 명령어의 나머지 부분은 sysx[3:0]을 통해 전달하였다. 제안된 DSP는 31비트의 명령어 길이를 갖기 때문에 COPIR 신호를 COPIR[30:0]으로 확장시켰고, 여러 개의 코프로세서를 장착시킬 수 있도록 sysx[3:0]은 코프로세서의 번호와 모드 전환을 위한 신호로 사용하기로 하였다. 이를 위해 sysx라는 명령어를 추가하였다.

마이크로 컨트롤러와 DSP간의 데이터 교환이 필요한 경우가 있는데, 이 경우를 위해 CLD 명령어가 존재한다. CLD 명령어는 코어와 코프로세서 사이에 레지스터에 있는 값을 교환할 수 있는 명령어로써 그림 11에서 보듯이 여러 인터페이스 신호를 사용해서 데이터 교환과 파이프라인 동기를 맞추게 된다.

V. 성능 평가

제안된 DSP의 성능을 알아보기 위해 같은 조건을 갖는 다른 세 개의 DSP와의 벤치마크를 수행하였다. 세 DSP 모두 MAC을 두 개씩 갖는 Dual MAC DSP이지만, 아키텍쳐에서는 약간씩 다르다. 표 2는 제안된 DSP를 포함한 네 개의 아키텍쳐를 비교한 표이다.

모두 고정된 명령어의 길이를 갖지만, TMS320C55x만이 가변적인 명령어 길이를 갖는다. 그래서 TMS-320C55x는 명령어 버퍼에 명령어를 32 비트씩 가져와서 저장해 놓은 후에, 명령어의 길이를 판별하여 이 버퍼에서 fetch하는 구조를 갖는다. 명령어가 가변적이므로, 명령어 수행 시간도 일정하지 않다. 그리고 메모리에서 메모리로의 연산이 가능한 것도 TMS320C55x의 특징이다.

TeakTMDSP는 여러 가지 주소 모드(address mode)를 지원하는 것이 장점이지만, 누적기가 4개인 점에서 오는 자원 부족 문제가 심각하다. DSP16000은 누적기는 8개로 많지만, 많은 주소 모드를 지원하지 않고, 곱셈기와 덧셈기가 지극히 직렬적이라는 문제를 가지고 있다. 즉, 덧셈기의 입력 중에서 곱셈기 출력을 제외하면 누적기의 입력은 상당히 제한되어 있다. 보통의 DSP는 곱셈기와 덧셈기를 직렬로 배열하는 것이 일반적이다. 즉, DSP의 주 응용 수식이 곱의 합 형태이기 때문에 곱셈기의 결과가 덧셈기의 입력으로 들어가는 구조가 흔한 것이다. 그러나, 제안된 구조는 명령어마다 곱셈기와 덧셈기를 지나는 경로를 다르게 할 수 있게 하였다. 곱셈기와 덧셈기의 배치를 병렬적으로 하여 곱셈 명령과 덧셈 명령을 동시에 수행할 수 있게 하였으나, 곱의 합 수식을 효과적으로 하기 위해서 곱셈의 결

표 2. 아키텍처 비교
Table 2. Comparison of architectures M : memory.

DSP / 내용	Teak™DSP [2]	DSP16000 [3,4]	TMS320C55x [5]	제안된 DSP
버스	-32bit data bus 3개 -32bit program bus 1개	-32bit data bus 1개 -32bit common bus 1개	-16bit read data bus 3개 -16bit write data bus 2개 -32bit program bus 1개	-32bit data bus 3개 -32bit program bus 1개
명령어 길이	16 bit and 32bit	16 bit and 32bit	8 to 48bit	31bit
명령어 캐쉬	None	124 bytes cache	24 Kbytes cache	124 bytes cache
명령어 버퍼	None	None	64 bytes buffer	64 bytes buffer
MAC구조	-2 multipliers -3 input ALU	-2 multipliers -2 input ALU -3 input ADDER	-2 MACs -2 input ALU	-2 multipliers -3 input ALU -2 input ADDER
곱셈기 입력	16bit registers (x0, x1, y0, y1)	32bit registers (x, y)	Not specified	32bit registers (x, y)
BMU	ALU	Separated BMU	Shifter	Seperated BMU
누적기	4	8	4	10
ALU 입력	abX, p0, p1 XDB	aX, y, p0, p1	CB, DB ACx Shifter output	aX(공유), y, p0, p1
데이터 전송	M to x, y, abX, ALU input	M to x, y, p0, p1, aX	M to M, ACx, MAC input, ALU input, shifter input	M to x, y, p0, p1, aX

표 3. 제안된 DSP와 다른 DSP와의 벤치마크 결과
Table 3. Benchmark result N, M : 싸이클 수

프로그램 목록	Teak™DSP [2]	DSP16000 [3,4]	TMS320C55x [5]	제안된 DSP
real 콘볼루션	N/2	N/2	N/2	N/2
complex 콘볼루션	2N	2N	2N	2N
Biquad(Form I) 5계수 단정밀도	9N	7N	7N	5N
Biquad(Form I) 5계수 배정밀도	20N	13N	16N	12N
Biquad(Form II) 4계수 단정밀도	3N	3N	4N	3N
Biquad(Form II) 4계수 배정밀도	14N	12N	14N	9N
Biquad(Form II) 5계수 단정밀도	8N	5N	5N	3N
Biquad(Form II) 5계수 배정밀도	17N	16N	17N	10N
Lattice Forward FIR	4N	4N	4N	3N
Lattice Inverse IIR	5N	5N	4N	3N
FFT Butterfly	7N/2	7N/2	5N/2	5N/2
LMS단정밀도	M/2+3N	M/2+2N	2M+2N	M/2+N
LMS배정밀도	2M+5N	3M/2+4N	5M+7N	3M/2+3N
Delayed LMS 배정밀도	2M+5N	3M/2+4N	5M+7N	3M/2+3N
Viterbi 디코딩	22N	26N	21N	21N

과를 덧셈의 입력으로 사용할 수 있는 경로도 추가하였다. 이렇게 함으로써, 명령어의 병렬성을 증가시키면서, DSP의 응용 분야에 적합한 연산 능력도 동시에 향상될 수 있는 것이다.

표 3은 제안된 DSP의 성능을 측정하기 위해 수행한 벤치마크의 결과를 나타낸 것이다. real convolution은 실수만으로 계산되는 convolution 연산을 나타내고, complex convolution은 실수와 허수가 복합적으로 계산되는 convolution 연산을 나타낸다. 그리고, Biquad라고 나타난 부분에서, Form I이라는 것은 지연 소자를 공유하지 않는 IIR 필터이고, Form II라는 것은 지연 소자를 공유하는 IIR 필터를 나타낸다. 또한, 여기서 계수라는 것은 필터 계수의 개수를 나타낸다. Lattice 필터는 여러 개의 직렬로 연결되어 필터 연산을 수행하는 필터로서, Forward FIR과 Inverse IIR은 다음 단에 연결되는 입력이 무엇인지를 나타내는 것이다. 그리고 LMS라는 것은 echo-canceler같은 곳에 사용되는 필터의 일종으로서, 결과를 입력으로 다시 넣으면서, 근사치를 구하는 필터이다. 마지막으로, Viterbi 디코딩은 길쌈 부호화한 코드의 에러를 정정하는 알고리듬을 나타낸다. 그리고, 단정밀도는 16-비트 연산을, 배정밀도는 32-비트 연산을 나타낸다. 벤치마크 결과는 다른 세 개의 DSP와 제안된 DSP의 명령어 집합을 숙지한 후, 각 알고리듬들을 최대의 ILP(instruction level parallelism)을 달성하도록 최적화하여 구현한 후에 싸이클 수를 비교한 것이다. 비교 대상인 다른 DSP들은 모두 MAC unit 두 개를 갖는 dual MAC DSP들이다. 표의 왼쪽에 나온 프로그램의 목록들은 DSP 응용에서 주로 사용되는 알고리듬들이고, 이것을 명령어 수준에서 직접 프로그램을 짜서 최적화한 후에 비교한 것이다. 물론 이 성능 비교 작업이 다른 DSP의 매뉴얼을 보고 명령어에 대한 숙지와 사용의 제한점을 이해한 후에 어셈블리 코드를 매뉴얼적으로 생성하여 최적화한 작업이므로, 약간의 오차가 존재할 수 있다. 그러나 그 오차를 최소화하기 위해 장시간에 걸친 최적화 과정을 거쳤다. 표 3에서 보듯이 제안된 DSP에서 모든 알고리듬들에서 같거나 좋은 성능을 가졌다. 이것은 앞에서 설명한 바와 같이 곱셈기의 입력 레지스터와 출력 레지스터를 공유하는 형태의 아키텍처라는 점이 가장 큰 이유로 작용하였다. 즉, DSP가 주로 처리하는 수식인 곱의 합을 효율적으로 처리할 수 있도록 아키텍처를 설계했기 때문에 성능이 좋아진 것이다.

그리고 최적의 bypass 구조를 사용함으로써 파이프

라인에서 의존도(dependency)로 인한 지연을 최소화할 수 있었다는 것도 중요한 이유 중의 하나이다. 그리고, 3-way 수퍼스칼라 구조도 고성능을 위해 필요한 구조였고, 그래서 그것을 택한 구조인 제안된 DSP가 성능이 좋게 나온 것이다. 또한, DSP 아키텍쳐에서는 수퍼스칼라 구조보다는 흔한 VLIW와 비교를 하면, 성능은 별로 차이가 나지 않는다.

그러나 VLIW보다는 프로그램 메모리를 적게 사용하므로, 메모리 효율 면에서는 훨씬 경제성이 있으므로, 모바일(mobile) 응용 분야에서는 제안된 구조가 VLIW 구조보다는 적당하다. 앞에서 제안하였던 DSP 아키텍쳐는 설명한 바와 같이 고성능을 가지면서, 전력을 적게 소모하며, 마이크로 컨트롤러의 coprocessor로 동작하는 DSP 아키텍쳐이다.

Ⅵ. 결 론

4장의 성능 평가에서 나타난 바와 같이, 제안된 DSP 구조는 고성능/저전력 DSP이다. 세가지 dual MAC DSP와의 성능 평가에서는 벤치마크를 위한 프로그램 목록 모두에서 다른 DSP들보다 성능이 좋거나 같은 성능을 보였다. 다음과 같은 사항이 제안된 DSP의 성능을 뛰어나게 한 요인들이라 할 수 있다.

첫째, 연속적인 곱셈에 강한 구조를 갖는다. 누적기 겸 레지스터의 구조에서 누적기와 곱셈기의 입력 레지스터와 공유함으로써, 이 공유된 레지스터는 ALU나 BMU, 그리고 곱셈기의 입력으로 들어갈 수 있음으로 해서 연속적인 곱셈, 예를 들면, 삼항 이상의 곱셈에서 별도의 이동 명령 없이 레지스터의 이름만 바꾸어 줌으로써 수행할 수 있다.

둘째, 곱셈기와 ALU의 병렬적인 배치가 곱의 합 처리 이외의 수식 처리에 성능을 좋게 하였다. 곱셈기와 ALU의 직렬적인 배치는 ALU의 입력이 제한됨으로써 ALU 연산에 제한이 있다. 제안된 DSP는 곱셈기와 ALU, 그리고 BMU 등의 연산 유닛들을 모두 병렬적으로 배치함으로써, 모든 연산 유닛들의 입력 제한을 최소화하였다. 그리고, 앞서 설명한 레지스터 공유 형태로 인해 입출력에 대한 데이터의 변환이 자유롭게 함으로써 이러한 연산 유닛들의 병렬 배치가 더욱 유용해졌다.

셋째, 한번에 최대 3개의 명령어를 처리할 수 있는 3웨이 수퍼스칼라 구조이다. 한번에 단일 명령어를 실행하는 DSP가 아닌, 최대 3개의 명령어를 실행할 수 있는 3 웨이 수퍼스칼라 구조이기 때문에, dual MAC DSP라는 구조와 더불어 성능 향상에 도움이 되었다. 그리고, 그것을 31 비트안에 인코딩했기 때문에, 한번에 가져오는 명령어도 한 개이면 충분한 구조이다. 이점은 명령어를 가져오는 횟수를 줄임으로서 전력소모에도 도움이 되는 구조이다. 그리고 실제로는 dual MAC 명령어인 곱셈 두 개와 ALU 연산 두 개, 그리고 load/store 두 개씩을 수행할 수 있기 때문에, 일반 단일 명령어 이슈 DSP 기준으로 보면, 최대 6개의 연산을 수행할 수 있는 구조이다.

마지막으로, 32 비트 버스를 가진 DSP이지만, 데이터형이 8 비트, 16 비트, 24 비트, 32 비트, 그리고 40 비트 데이터형을 누적기의 선택에 의해 자유롭게 변환되므로, 단정밀도나 배정밀도에서의 연산이 자유롭다. 그리고, 32 비트 데이터의 메모리 연산 뿐만 아니라, 24 비트 데이터의 메모리 연산과 누적기의 데이터의 상위 부분의 벡터화도 지원함으로써 성능향상에 도움이 되었다.

참 고 문 헌

[1] K.-M. Lim, S.-W. Jeong, Y.-C. Kim, S.-J. Jeong, H.-K. Kim, Y.-H.Kim, B.-Y. Chung, and H.-L. Roh. "CalmRISCTM : A Low Power Microcontroller with Efficient Coprocessor Interface", Proc. Intl Conf. Computer Design (ICCD), pp 299-302, October 1999.

[2] TeakTMDSP Core Architecture Spec.

[3] DSP16000 Reference Guide

[4] M. Alidina, G. Burns, C. Holmqvist, E. Morgan, D. Rhodes, S. Simanapalli, M.Thierbach, "DSP16000 : a high performance, low-power dual-MAC DSP core for communications applications", Custom Integrated Circuits Conference(CICC), pp119-122, 1998.

[5] TMS320C55x DSP CPU Reference Guide.

[6] Seshan, N., "High VelociTI processing [Texas Instruments VLIW DSP architecture]", IEEE Signal Processing Magazine , Volume: 15 Issue : 2 , pp86 -101, 117, March 1998.

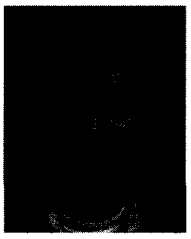
[7] Simar, R., Jr., "Codevelopment of the TMS320C6X VelociTI architecture and

compiler", Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on , pp3145-3148 vol.5, 1998.

저 자 소 개

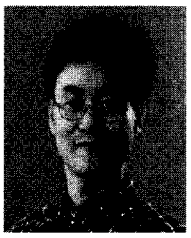
尹 晟 暾(正會員)

2000년 2월 : 연세대학교 전기공학과 (공학사). 현재 : 연세대학교 전기전자공학과 석사 과정 재학 중 <주관심분야> VLSI & CAD, DSP, 테스팅 등임



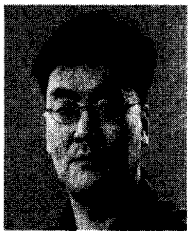
裵 晟 日(正會員)

1998년 2월 : 경북대학교 전자공학과(공학사). 2000년 8월 : 연세대학교 전자공학과(공학석사). 현재 : 연세대학교 전기전자공학과 박사 과정 재학 중. <주관심분야> VLSI & CAD, DSP, 테스팅 등임



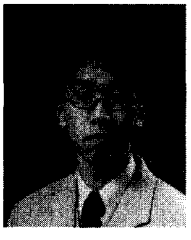
金 容 天(正會員)

1988년 2월 : 연세대학교 전자공학과(공학사). 현재 : 삼성전자 CalmRISC팀. <주관심분야> VLSI & CAD, DSP 등임



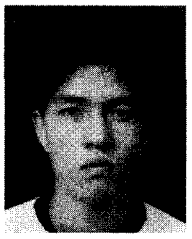
鄭 勝 在(正會員)

1993년 2월 : 연세대학교 전자공학과(공학사). 1995년 2월 : 연세대학교 전자공학과(공학석사). 현재 : 삼성전자 CalmRISC팀. <주관심분야> VLSI & CAD, DSP 등임



文 相 煮(正會員)

1998년 2월 : 경북대학교 컴퓨터공학과(공학사). 2000년 2월 : 서울대학교 컴퓨터공학과(공학석사). 현재 : 삼성전자 CalmRISC팀. <관심분야> VLSI & CAD, DSP 등임



金 相 郁(正會員)

2001년 8월 : 연세대학교 전기공학과(공학사) 현재 : 연세대학교 전기전자공학과 석사 과정 재학 중 <주관심분야> VLSI & CAD, DSP, 테스팅 등임

姜 成 昊(正會員)

1986년 2월 : 서울대학교 제어계측공학과(공학사). 1988년 5월 : Electrical and Computer Eng., Univ. of Texas at Austin(공학석사). 1992년 5월 : Electrical and Computer Eng., Univ. of Texas at Austin(공학 박사). 1989.11~1992.8 : 미국 Schlumberger Inc. Research Scientist. 1992.9~1992.10 : 미국 The Univ. of Texas at Austin, Post Doctoral Fellow. 1992.8~1994.6 : 미국 Motorola Inc., Senior Staff Engineer. 1994.9~1999.8 : 연세대학교 기계전자공학부 조교수. 1999.9~현재 : 연세대학교 기계전자공학부 조교수. <주관심분야> 테스팅, DFT, VLSI & CAD, Design Verification 등임

金 相 佑(正會員)

1996년 2월 : 연세대학교 전자공학과(공학사). 1998년 2월 : 포항공대 전기전자공학과(공학석사). 현재 : 삼성전자 CalmRISC팀. <주관심분야> VLSI & CAD, DSP 등임