ARM-7 코어를 이용한 Dolby Pro Logic 복호기의 실시간 구현

정회원 이 창 우*, 이 상 근**, 조 재 문**

Real-time Implementation of Dolby Pro Logic Decoder Using ARM-7 Core

Chang Woo Lee*, Sang Keun Lee**, Jae Moon Jo** *Regular Members*

요 약

기존의 2 채널 stereo의 한계를 극복하고 음의 입체감을 향상시키기 위해서 2 채널 이상의 다채널로 부호화하는 기법 중에서 Dolby Pro Logic 방식은 음의 입체감이나 분리도 측면에서 매우 우수한 방법으로 고 음질을 요구하는 Hi-Fi 음향 기기에 많이 채용되고 있다. 본 논문에서는 Dolby Pro Logic 복호기를 ARM-7 코어를 사용하여 구현하기 위해서 최적의 ARM-7 code를 작성하였다. 이를 위해서 고정 소수점 연산으로 code를 구현하여 최적화를 수행하고 이의 수행 시간과 정밀도를 측정하여 구현한 code의 타당성을 입증하였다.

ABSTRACT

In order to enhance multi-channel audio signals, Dolby Pro Logic is widely used especially for the Hi-Fi audio system, since it can provide highly stereophonic effects and a nice separation of multi-channel sound. This paper describes an implementation of Dolby Pro Logic decoder with ARM-7 core. The code is modified for the fixed point operation and optimized. For the verification of the code, the operation time and the precision are estimated thoroughly. As a result, it is verified that Dolby Pro Logic decoder can be implemented with ARM-7 core operating at 54 MHz.

I. 서 론

기존의 2 채널 스테레오의 한계를 극복하고 음의 입체감을 향상시키기 위해서 여러가지 연구가 행하여져 왔다. 그중에서 DVD (digital video disk) 나 HDTV (high definition television) [1] 의 표준 음향 부호화 방식으로 결정된 Dolby AC-3 방식[2,3]이나 MPEG-2 오디오 방식[4]은 5.1 채널 (left, center, right, left surround, right surround, low frequency effect) 의 음향 신호를 효율적으로 압축하는 방식이다. 이에 비해서 Dolby Pro Logic 방식[5]은 기존 stereo 방식에서 사용되는 2 채널 (left, right) 로부터 4 채널 (left, center, right, surround) 을 만들어서 음의 방향성과 분리도를 높여서 음의 입체감을 향상시키려는 기법이다. 이 방식은 2 채널로부터 4 채널을 만들기 때문에 기존의 stereo sound track을 사용할 수 있어서 stereo 방식의 음향 기기와 호환성이 뛰어나고 재생되는 음의 입체감이 좋은 장점이 있다. 특히 최근에 개발되는 HDTV를 위한 음향 복호기는 호환성의 유지를 위해서 Dolby AC-3 방식 뿐 아니라 Dolby Pro Logic 방식으로 부호화된 음향 신호도 모두 복호할 수 있도록 설계하는 것이 일반적이다.

* 가톨릭 대학교 컴퓨터 · 전자 공학부 (lcw@www.cuk.ac.kr),
** 삼성전자 중앙연구소 정보미디어 연구소 (lsk@rnd.sec.samsung.co.kr)
논문번호 : 99009-0108, 접수일자 : 1999년 1월 8일
※ 본 연구는 삼성전자와 가톨릭대학교 1998년 교비 연구비 지원으로 수행되었습니다.

본 논문에서는 RISC (reduced instruction set computer) 코어를 사용한 Dolby Pro Logic 복호기의 ASIC (application specific integrated circuit) 구현에 대해서 고찰한다. RISC 코어나 DSP (digital signal processor)를 이용하는 방식은 프로그램을 기본으로 구현되기 때문에 모든 회로를 hard-wired logic으로 구현하는 방식에 비해서 기능 변화가 용이한 장점이 있다. RISC 코어로는 ARM (advanced RISC machine) - 7 코어[6]를 사용하였다. ARM-7은 Advanced RISC Machines 사에서 개발한 32 bit 범용 RISC 코어로서 음향 복호화를 위한 충분한 정밀도를 보장한다. 32 bit 고정 소수점 곱셈 (fixed point multiplication)을 위한 multiplier와 address 연산등을 쉽게 하기위한 barrel shifter를 내장하고 있다. 또한 코어의 크기와 소모하는 전력이 작은 장점이 있고 주변 기기와의 interface가 용이하다. 그러나 3장에서 밝히는 것과 같이 ARM-7을 이용해서 실시간으로 Dolby Pro Logic 복호를 구현하려면 code를 시간 측면에서 최적화해야 한다. ARM-7을 이용한 Dolby AC-3 복호기는 이미 개발되어 동작이 검증되었다[7].

특히 Pro Logic 복호기의 전체 기능을 ARM-7으로 구현하기 위한 code 최적화 작업을 집중적으로 연구하였는데 이를 위해서 먼저 고정 소수점 (fixed point) 연산을 위한 ARM-7 code 구현 방법에 대해서 연구하고 ARM-7 simulator를 이용한 복호 (decoding) 시간을 측정하였다. 이를 토대로 ARM-7 code를 최적화 하였고 에러 해석과 동작 시간 검토를 통하여 구현한 code를 검증하였다.

Ⅱ. Dolby Pro Logic

2.1 Dolby Pro Logic 부호화기

Dolby AC-3나 MPEG-2 오디오와 같은 음향 부호화 기술이 발달하기 이전에 2 채널 음향 정보로부터 2 채널 이상의 다채널로 부호화하는 기법 중에서 Dolby Pro Logic 방식[5]은 1987년에 처음 상용화된 방식으로 음의 입체감이나 분리도 측면에서 매우 우수한 방법으로 고 음질을 요구하는 Hi-Fi 음향 기기에 많이 채용되고 있다. 특히 Dolby Pro Logic 방식은 기존의 stereo 방식과 완전히 호환성을 유지하는 특성이 있다.

Dolby Pro Logic surround 방식을 구현하기 위해서는 4 채널 (left, right, center, surround) 의 음향 신호를 2 채널로 부호화하기 위한 Pro Logic 부호화기와 부호화된 2채널 (Lt,Rt) 로부터 4 채널을 복호화하기 위한 Pro Logic 복호기가 구현되어야 한다.

먼저 그림 1에 Dolby Pro Logic 부호화기의 블록도를 도시하였다. Dolby Pro Logic 부호화기는 Left, center, right, surround의 4 개 채널의 입력을 받아 Lt, Rt의 2 개의 Dolby Pro Logic과 호환 가능한 채널을 생성한다. 이때 center 채널과 surround 채널은 각각 3 dB 만큼 level이 낮추어져서 left 채널과 right 채널에 섞이게 되는데 center 채널 신호는 left, right 채널에 모두 더해지는 방향으로 surround 채널 신호는 left와 right 채널의 신호에 위상이 반전되어 더해지게 된다. 특히 surround 채널은 front 채널인 left, center, right 신호와의 분리도를 좋게 하여 멀리서 들려오는 듯한 효과를 내기 위해서 추가로 band pass filtering과 Dolby NR 부호화기를 통과하게 된다. 따라서 복호기 측에서 이를 복호하기 위해서는 역과정을 수행해야 한다.

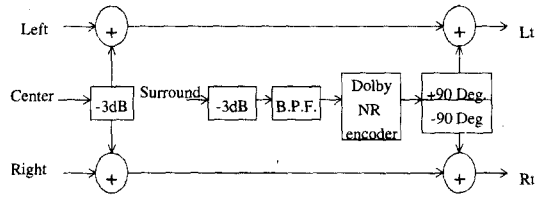


그림 1. Dolby Pro Logic 부호화기.

2.2 Dolby Pro Logic 복호기

Dolby Pro Logic 복호기의 주요 기능은 Lt, Rt 두 입력을 바탕으로 left, center, right, surround의 4개 채널을 모두 만들어 내야 한다. 그림 2에 Dolby Pro Logic 복호기의 블록도를 도시하였다. Dolby Pro Logic 복호기는 기본적으로 Pro Logic 부호화기의 역과정을 수행한다. 즉, Lt, Rt의 두 채널 입력을 바탕으로 left, center, right, surround의 4 채널을 만들어 내는데 각 부분의 동작 원리를 살펴보면 다음과 같다.

먼저 surround 채널을 만들기 위한 기본적인 가정은 음향 신호의 주된 부분은 앞 채널에 존재한다고 가정하는 것이다. Surround 채널은 앞 채널의 음향을 보조해주는 역할을 하게 되는데 앞 채널에 비해서 멀리서 들려오게 된다고 느껴질수록 음의 입체감을 높이고 앞 채널로부터의 음의 leakage를 감소하는 효과를 얻을 수 있다. 이를 위해서 그림 2의 audio time delay 부는 surround 채널의 음이

front 채널의 음보다 청취자의 귀에 늦게 도달하게 함으로써 음의 입체감을 높인다. 또한 7 kHz low-pass filter는 surround 음의 high frequency 성분을 제거하는데 이는 음의 high frequency 성분이 없어지면 음이 보다 멀리서 들리는 듯한 효과를 내게 된다. 또한 modified Dolby B-type NR 복호기는 Pro Logic 부호화기의 B-type NR 부호화기의 역과정을 수행하는데 신호의 leakage 성분과 잡음을 효과적으로 상쇄한다.

다음에 Dolby Pro Logic 적응 행렬 (adaptive matrix) 부는 Dolby Pro Logic 복호기의 중심이 되는 부분이다. 부호화된 2 채널의 음향 신호를 계속 측정해서 음향의 주된 방향 성분을 찾아 4 채널 복호화시 이를 강조해서 음의 입체감을 극대화 시킨다. 이때 음의 입체감을 높이기 위해서는 음의 분리도와 음의 방향성을 극대화시키는 것이 중요하다. 음의 분리도와 방향성을 극대화시키기 위해서 사용되는 것이 적응 행렬 부인데 음의 분리도를 높이기 위해서는 마주보는 speaker의 분리도를 최대한으로 하는 것이 중요하다. 즉, left 채널과 right 채널을 하나의 쌍으로 생각하고 center 채널과 surround 채널을 다른 쌍으로 생각하여 각 쌍의 분리도가 최대로 되도록 적응 행렬을 설계한다. 각 쌍에 해당하는 적응 행렬의 proto-type을 설계하고 음의 방향성을 계속 측정하여 방향성 계수를 적절한 proto-type 행렬 쌍에 곱하고 더함으로써 행렬의 값을 얻을 수 있다. 구해진 4×2 적응 행렬을 입력되는 Lt, Rt 채널에 곱함으로써 4 채널의 출력을 얻게 되는 것이다.

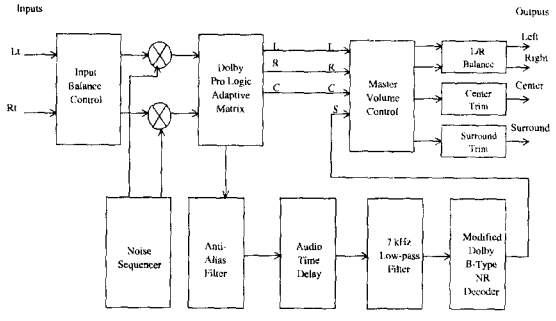


그림 2. Dolby Pro Logic 복호기.

그림 3에는 채널 separation map을 도시하였다. 그림에서 알 수 있듯이 Dolby Pro Logic 복호기는 특히 마주보는 채널의 음의 분리도를 최대로 하여 음의 입체감과 방향성을 높이도록 설계되어 있다.

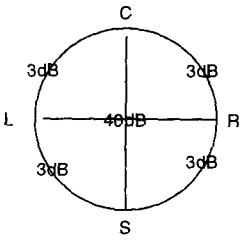


그림 3. Dolby Pro Logic 복호기의 채널 분리도.

Ⅲ. Dolby Pro Logic 복호기 프로그램 최적화

3.1 Dolby Pro Logic 복호기 프로그램 알고리듬

본 장에서는 Dolby 사에서 제공하는 프로그램을 바탕으로 Dolby Pro Logic 복호기를 ARM-7을 사용하여 구현하기 위한 프로그램 최적화에 대해 논한다.

Pro Logic decoding을 위해서는 Lt, Rt 2 채널로부터 center 채널과 surround 채널 신호를 생성해야 하는데 center 채널 신호는 Lt 채널 신호와 Rt 채널 신호의 합으로 surround 채널은 Lt 채널 신호와 Rt 채널 신호의 차로부터 구할 수 있다. 그림 2에 도시한 것과 같이 surround 채널의 처리를 위해서는 lowpass filter와 B-type filter를 거쳐야 하는데 이는 surround 채널에서 들리는 소리가 front 채널에서 들리는 소리와 구분이 되도록 하기 위한 것이다.

Surround 채널을 위한 lowpass filtering과 B-type filtering의 주파수 응답 특성을 그림 4와 그림 5에 도시하였다. B-type filter의 계수는 prol_side routine에서 계산되는데 입력되는 음향 신호의 변화되는 정도에 따라서 그림 5와 같은 lowpass filter와 all pass filter 사이의 특성을 보인다.

다음에 적응 행렬 값을 계산하기 위해서는 마주보는 채널인 left와 right 채널 신호와 center와 surround 채널 신호의 상대적인 power 비를 계산하여 만일 left 채널 신호의 power보다 right 채널 신호의 power가 더 크다면 적응 행렬을 계산할 때 표 1에 나타낸 ***mat_l*** 값이 사용되고 그렇지 않으면 ***mat_r*** 값이 사용된다. Center 채널과 surround 채널에 대해서도 마찬가지로 ***mat_c***이나 ***mat_s*** 값이 선택된다. 이렇게 선택된 행렬 값에 각 채널의 power로부터 계산된 계수가 곱해지고 고정된 행렬인 ***mat_fix***에 더해져서 새로운 행렬 값이 계산된다. 만

일 left 채널과 center 채널이 선택되었다면 적응 행렬 mat_{adap}은 다음 식으로 계산할 수 있다.

$$mat_{adap} = mat\_fix + fastlr \times mat\_l + fastcs \times mat\_c \quad (1)$$

이때 fastlr, fastcs 값은 각각 left 채널, right 채널과 center 채널, surround 채널의 power로부터 계산되어지는 값이다. 표 1에서 알 수 있듯이 mat_l은 mat_r에 비해서 left 채널 신호를 크게 하고 mat_c는 mat_s에 비해서 center 채널의 신호를 강조한다. 즉, 계산되는 행렬 값은 채널의 power에 따라서 power가 강한 채널을 더욱 강조하는 방향으로 설정된다.

3.2 Pro Logic 복호기 프로그램의 최적화

Dolby Pro Logic 복호기를 ARM-7에서 구현하기 위해서 3.1 절에서 설명한 Dolby 사에서 제공하는 Pro Logic 복호기 프로그램을 기반으로 하여 ARM-7 simulator를 사용하여 성능을 측정하였다. 이때 ARM-7은 54 MHz로 동작한다고 가정하였는데 시간 최적화를 하지 않는 경우에는 뒤에서 밝히는 것처럼 실시간 복호가 불가능하다. 따라서 실시간 처리를 위한 프로그램 최적화를 하였는데 그 기법을 뒤에 자세히 설명하였다.

또한 Pro Logic 프로그램은 부동 소수점 (floating point) 연산을 위해서 작성되었기 때문에 이를 고정 소수점 (fixed point) 연산으로 고쳐야 한다. 고정 소수점의 덧셈과 곱셈 연산에 대한 설명과 이의 정밀도에 대해서는 4장에 자세히 설명하였다. 그런데 Pro Logic 복호기 프로그램에서 dominance power를 계산하기 위해서는 한번의 나눗셈이 필요하다.

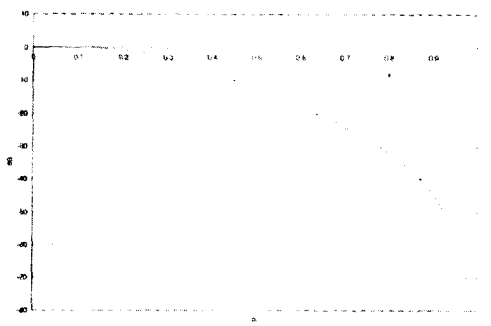


그림 4. Surround 채널 처리를 위한 7 kHz lowpass filter의 주파수 응답 (48 kHz sampling의 경우).

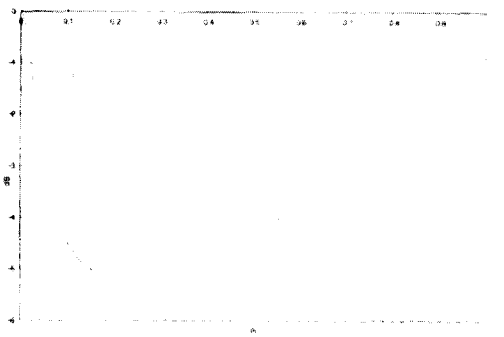


그림 5. Surround 채널 처리를 위한 B-type shelf filter의 주파수 응답 (48 kHz sampling의 경우).

표 1. Dolby Pro Logic 적응 행렬을 계산하기 위한 행렬.

mat_fix		*mat_l*	
0.707106781	0	0.292893218	0
0.5	0.5	-0.5	0
0	0.707106781	0	0
0.707106781	-0.707106781	-0.707106781	0.207106781
mat_r		*mat_c*	
0	0	-0.35355339	-0.35355339
0	0.5	0.207106781	0.207106781
0	-0.292893218	-0.35355339	-0.35355339
0.207106781	-0.707106781	-0.207106781	0.207106781
mat_s			
0.35355339	-0.35355339		
0	0		
-0.35355339	0.35355339		
0	0		

나눗셈을 부동 소수점 연산으로 수행하게 되면 정밀도는 보장되지만 시간이 매우 오래 걸리고 부가적인 library가 많이 필요하게 된다. 만일 정수의 나누기 연산을 이용하여 나누기 연산을 시행하게 되면 정밀도에서 문제가 발생한다. 이를 방지하기 위해서 본 연구에서는 연산시 30 bits의 정밀도를 갖는 div_fix routine을 작성하여 문제를 해결하였다. 고정 소수점 연산을 위한 프로그램의 정확성을 측정하기 위해서 부동 소수점 연산 결과와 본 연구에서 구현한 복호기를 사용한 결과를 SNR (signal to noise ratio) 척도를 사용하여 비교하였는데 그 결과는 4장에서 자세히 다룬다.

수행 시간을 측정하기 위해서는 test vector를 먼저 선정해야 한다. Test vector는 sine wave oscillator인 osc command를 사용해서 만들었는데 test vector의 특성은 표 2에 도시하였다.

Test vector를 사용하여 수행 시간을 측정하여 보았는데 부동 소수점 (floating point) 연산을 사용하였을 때와 부동 소수점 연산을 고정 소수점 연산으

로 고친 경우, 고정 소수점 연산에 대해 C source 최적화와 assembler 최적화를 각각 수행하였을 때의 수행 성능을 측정하였다. C source 최적화와 assembler 최적화에는 다음과 같은 기법들을 사용하였다.

첫번째로 C source code를 ARM-7 코어의 수행 속도를 향상시키도록 최적화하였는데 이때 사용된 주요 기법은 다음과 같다.

- ARM-7 코어는 32 bits 코어이기 때문에 integer 연산이 short 연산에 비해서 수행 시간이 적게 걸린다. Short 형태로 선언이 된 변수를 integer 형태로 고침으로써 약 10%의 수행 속도의 향상을 가져올 수 있었다.

표 2. Test vector의 특성 (0.1 초의 2 채널 data, 48kHz sampling).

dnrng1.pcm	0~100 dB attenuation, 3kHz sine wave, Steer to Left (Odd-numbered channels disabled)
dnrng2.pcm	3~103 dB attenuation, 3kHz sine wave, Steer to Center (All channels in phase)
dnrng3.pcm	0~100 dB attenuation, 3kHz sine wave, Steer to Right (Even-numbered channels disabled)
dnrng4.pcm	3~103 dB attenuation, 3kHz sine wave, Steer to Surround (Even numbered channels out of phase)

- C source 프로그램에서 global variable로 선언된 변수의 addressing에는 index 계산에 불필요한 clock이 소요된다. 크기가 크지 않은 별도의 local variable을 새로 만들고 반복적인 loop 계산 이전에 global variable을 local variable로 바꾸어 줌으로써 수행 시간을 많이 줄일 수 있었다.

- Assembler를 사용하는 경우 for loop의 반복 계산을 위해서는 branch 명령이 많이 사용된다. Branch 명령에는 최소한 3 clock이 소요된다. Loop 내의 routine을 몇 번 반복 사용하면 branch 명령을 줄일 수 있고 loop 내에서 최적화할 수 있는 여지도 많아진다. 이를 loop unrolling 기법이라 한다.

다음에는 ARM-7 assembler를 다음과 같은 방법을 사용하여 최적화하였다.

- ARM 7의 user mode에서 사용할 수 있는 register는 15 개이다. 그런데 stack pointer 등으로 사용되는 register를 제외하면 실제 사용 가능한 register는 12 개 정도이다. 그런데 memory에 저장된 data를 직접 연산에 사용하기 위해서는 data access에 3 clock 정도가 소요되기 때문에 자주 사용되는 변수를 memory로부터 register에 load하여 향후 다시 memory를 access하지 않고 register로부터 바로 연산을 하는 것이 효율적이다. 이것은 특히 for 문이나 while 문에서와 같이 반복적인 계산을 하는 부분의 수행 시간을 줄이기 위해서 유용한데 ARM-7 컴파일러의 선택 사항에서 시간 최적화를 선택하여도 완전히 최적화되지 않는다. 따라서 일단 최적화하려는 부분을 ARM-7 code로 compile한 후에 ARM-7 code에서 register 할당 관계를 직접 고려하여 code를 최적화시키면 수행 시간을 많이 단축시킬 수 있다.

- C source 최적화를 위해서 사용한 loop unrolling 기법을 이용하고 load multiple 또는 store multiple 명령을 사용하여 수행 속도를 줄일 수 있다.

특히 loop unrolling 기법과 assembler에서 register의 최적 할당 기법을 이용하여 수행 시간을 많이 줄일 수 있었다.

표 3에 0.1초의 dnrng1.pcm~dnrng4.pcm data에 대해서 L,C,R,S 4 개의 출력 채널을 생성하는 경우의 수행 시간을 정리하였다.

표 3. Dnrng*.pcm data에 대한 decoding time (μ s) (0.1초 입력 data, ARM: 54MHz clock, L,C,R,S 출력).

Test vector	Floating Point	Fixed Point	C source optimization	Assembler optimization (file operation 제외)
dnrng1.pcm	2,061,953	229,830	162,831	117,321(91,510)
dnrng2.pcm	2,097,429	229,348	162,846	117,591(91,720)
dnrng3.pcm	2,060,334	230,004	162,982	117,473(91,629)
dnrng4.pcm	2,138,270	231,204	164,563	119,384(93,120)

표 3에서 알 수 있듯이 부동 소수점 연산에 비해서 고정 소수점 연산을 사용하면 수행 시간을 약 1/10로 줄일 수 있다. 고정 소수점 연산을 사용해도 decoding 시간은 data가 차지하는 시간의 2 배 이상이 되는데 C source 최적화와 assembler 최적화를 통해서 수행 시간을 반 이하로 줄일 수 있었다. Test vector 중에서 수행 시간이 가장 오래 걸리는 dnrng4.pcm data의 경우 수행 시간은 약 0.093 초로 원래 data가 차지하는 시간인 0.1 초에 비해서 decoding 시간 면에서 약 7%의 여유를 갖고 decoding할 수 있어서 속도면에서는 큰 문제가 없을 것으로 판단된다.

Ⅳ. 고정 소수점 연산을 위한 ARM-7 code 구현과 에러 해석

4.1 고정 소수점 연산을 위한 ARM-7 code 구현

Dolby Pro Logic 복호기에서는 2 채널로 부호화한 정보를 4 채널로 복호하기 위해서 실수 연산을 많이 해야 하는데 복호기를 실제 구현할 때 실수 연산을 부동 소수점 연산으로 구현하는 것은 하드웨어나 소프트웨어의 크기나 속도 면에서 매우 비효율적이 되기 때문에 보통 고정 소수점 연산으로 구현하게 된다.

실수 연산을 구현할 때 만일 부동 소수점 연산을 사용한다면 $m \times 2^{-e}$와 같이 mantissa m과 exponent e를 사용해서 실수 값을 나타내게 된다. 고정 소수점 연산의 경우에는 exponent 부가 일정한 수로 고정이 된다. 먼저 고정 소수점 연산에서 덧셈의 경우에 대해서 살펴보면 다음과 같다. Exponent를 나타내는 변수가 e로 고정된 경우 더해지는 두 수가 각각 $m1 \times 2^{-e}$과 $m2 \times 2^{-e}$라면 덧셈의 결과는 $(m1+m2) \times 2^{-e}$이 되어서 고정 소수점에 영향을 미치지 않는다. 그러나 곱셈의 경우에는 결과가 $(m1 \cdot m2) \times 2^{-2e}$가 되어서 결과의 exponent 값이 원래의 고정 소수점 형식에서 벗어나게 된다. 따라서 결과를 $m \times 2^{-e}$로 나타내기 위해서는 한번의 shift 연산이 필요하게 된다.

그런데 Dolby Pro Logic에서 사용되는 신호의 크기는 1 이하의 값을 갖고 연산의 정밀도도 최소 16 bits을 요구한다. 만일 sign bit로 1 bit를 사용한다면 최소 16 bits의 곱하기 연산 결과는 34 bits에 저장되어야 한다. ARM-7의 경우 정수 곱하기의 결과는 32 bits로 저장되기 때문에 정밀도에서 문제가 생긴다. 이 문제를 해결하기 위해서 long multiplication 명령어를 사용하였다.

Long multiplication 명령어를 사용해서 고정 소수점 연산을 구현하면 두 개의 32 bits 정수형 변수의 곱셈 결과가 64 bits의 정수형 변수로 저장이 된다. 본 연구에서는 고정 소수점을 그림 6에서 볼 수 있는 것과 같이 MSB에서 4 번째로 고정을 하였다. 이때 $m1 \times 2^{-28} \times m2 \times 2^{-28}$의 결과는 $(m1 \times m2) \times 2^{-56}$이 되어서 결과를 $m \times 2^{-28}$ 형태로 나타내려면 상위 32 bits를 취해서 왼쪽으로 4 bits 만큼 shift 하면 된다.

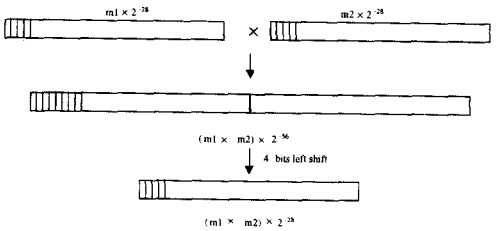
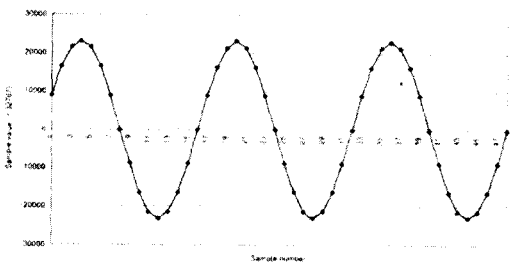


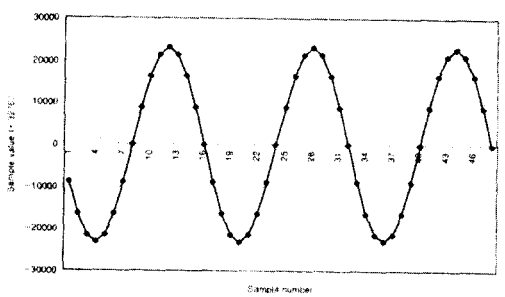
그림 6. ARM-7에서 고정 소수점 곱셈.

4.2 고정 소수점 연산의 정밀도 해석

Dolby Pro Logic 복호기의 동작 상태와 정밀도를 측정하기 위해서 4.1 절의 수행 시간 측정을 위해서 사용한 test vector인 dnrng1.pcm~dnrng4.pcm을 그대로 사용하여 decoding 결과 파형과 SNR을 측정하였다. 먼저 dnrng4.pcm의 파형과 decoding 후의 파형을 부동 소수점 연산과 고정 소수점 연산의 경우에 대해서 그림 7~그림 8에 각각 제시하였다. 그림 7에서 알 수 있듯이 dnrng4.pcm은 left 채널과 right 채널의 위상이 반대인 3kHz sine wave 이다. 또한 attenuation이 3~103 dB를 갖는 신호로서 크기가 점점 감소하는 특성을 갖는다.



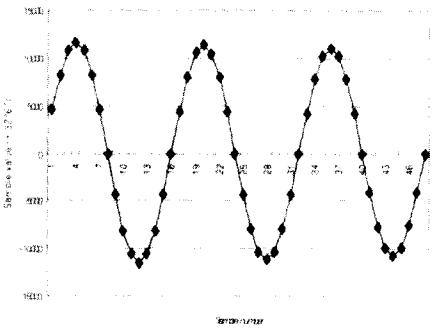
(a) Left 채널



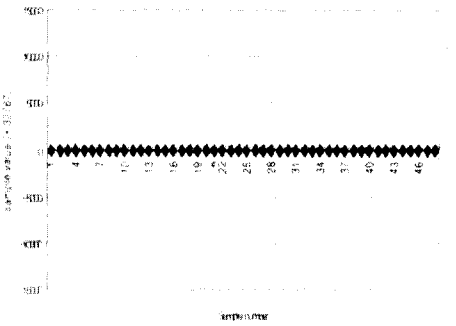
(b) Right 채널

그림 7. Dnrng4.pcm 파형

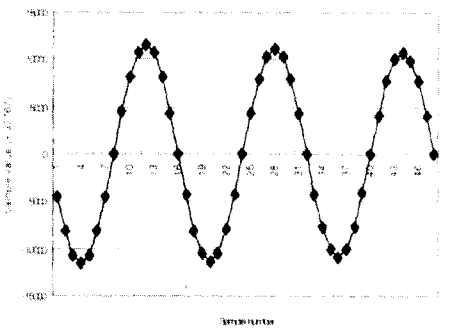
이러한 신호를 decoding한 결과를 그림 8에 도시하였는데 left 채널과 right 채널은 decoding 전과



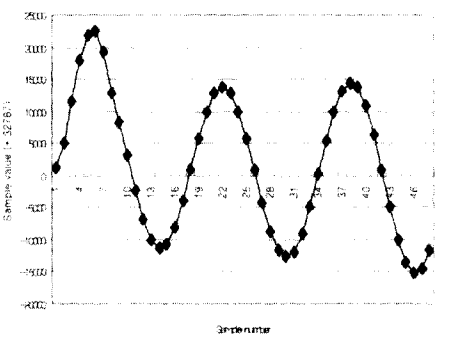
(a) Left 채널



(b) Center 채널



(c) Right 채널

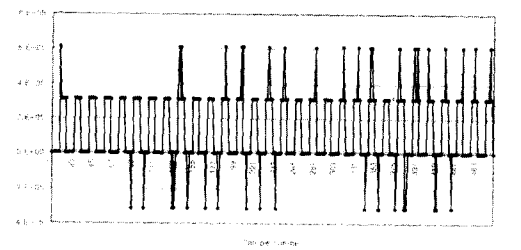


(d) Surround 채널

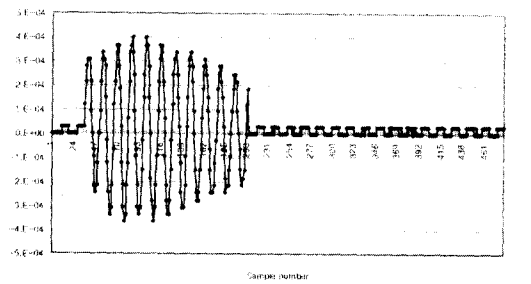
그림 8. Dnrng4.pcm의 복호 후의 파형 (부동 소수점 연산)

파형의 모양이 크게 바뀌지 않으나 center 채널은 left 채널과 right 채널의 합으로부터 구하고 둘의 위상이 반대이기 때문에 크기가 0으로 된다. 부동 소수점 연산과 고정 소수점 연산 결과의 파형을 비교하여 보면 파형에서 느낄 수 있는 차이점은 거의 없다.

고정 소수점 연산으로 인한 오차를 알아보기 위해서 부동 소수점 연산의 결과의 차 신호를 구해서 그림 9에 도시하였다. 차 신호를 구해보면 left, center, right 채널 신호는 오차가 매우 작은 반면에 surround 채널 신호는 오차가 다른 채널에 비해서 크게 나타난다. 이는 surround 채널 연산에는 low-pass filtering과 B-type noise reduction filtering 등의 연산이 추가로 필요하기 때문이다.



(a) Left 채널



(b) Surround 채널

그림 9. 부동 소수점 연산과 고정 소수점 연산의 차이

고정 소수점 연산으로 인한 오차를 측정하기 위해서 test vector인 dnrng1.pcm~dnrng4.pcm에 대해서 부동 소수점 결과와의 SNR을 측정하였다. SNR은 부동 소수점 연산과 고정 소수점 연산의 결과 신호의 차이의 에너지를 부동 소수점 연산 결과 에너지로 나눈 값으로서 다음 식으로 정의된다.

$$SNR\ (dB) = 10 \cdot \log_{10} \frac{E_s^2}{E_e^2} \qquad (2)$$

식 (2)에서 E_s^2 는 신호의 에너지로서 부동 소수

점 연산 결과로 생기는 신호의 에너지를 사용하였고 E_e^2는 에러 신호의 에너지로서 부동 소수점 연산과 고정 소수점 연산 결과 차이인 에러 신호의 에너지를 의미한다. 계산한 SNR 값을 표 4에 제시하였다. 표 4에서 알 수 있듯이 고정 소수점 연산에서 발생하는 오차를 나타내는 SNR은 약 65 dB 이상의 값을 가져서 대체로 정확한 결과라고 판단할 수 있다.

다음에 Pro Logic 복호기를 구현하였을 때 필요한 memory를 계산하였다. Pro Logic 복호기에서 필요로하는 memory는 크게 프로그램을 수행하기 위한 memory와 data를 저장하기 위한 RAM으로 나눌 수 있다. 본 논문에서는 프로그램 영역으로 RAM을 사용하여 필요한 프로그램을 외부에서 download 받을 수 있도록 설계하였기 때문에 Dolby AC-3 복호기와 Pro Logic 복호기를 하나의 chip에 구현하는 경우라도 프로그램 memory의 크기는 두 프로그램의 합으로 결정되어지는 것이 아니라 두 프로그램 중에서 크기가 큰 쪽으로 결정되어 진다. 만일 복호기에 MPEG 오디오 복호기 등의 기능을 추가하고 싶은 경우라도 역시 MPEG 오디오 복호기 프로그램 크기가 Dolby AC-3 프로그램 크기를 넘지 않는다면 프로그램 영역을 구현하기 위한 추가 memory는 필요가 없다. 프로그램 영역을 ROM으로 바꾸는 경우, 같은 용량이라면 ROM이 차지하는 크기가 RAM이 차지하는 크기에 비해서 작은 장점이 있지만 모든 프로그램 크기의 합으로 ROM의 용량이 결정되어야 한다. 따라서 두 경우의 memory 크기를 정확히 비교하는 것이 필요하다. Pro Logic 복호기에서 필요로 하는 memory는 프로그램과 data를 위한 RAM이 각각 9.42 Kbytes와 12.752 Kbytes가 각각 필요하다.

표 4. Test vector에 대한 SNR.

Test vector	SNR (dB)
dnrng1.pcm	65.30
dnrng2.pcm	67.02
dnrng3.pcm	65.25
dnrng4.pcm	65.14

V. 결 론

본 논문에서는 ARM-7 RISC 코어를 사용하여 Dolby Pro Logic 복호기를 구현하였다. Dolby AC-3 복호기와 Dolby Pro Logic 복호기를 구현하기 위한 여러 가지 방법 중에서 ARM-7 RISC 코어를 이용하였는데 이러한 방법은 전체 구조를 hard-wired logic으로 고정시킨 구조에 비해서 기능을 변형시키거나 새로운 기능을 추가시키고자 할 때 소프트웨어로 처리할 수 있는 유연성이 있다.

Dolby Pro Logic 복호기는 모두 24 bits의 정밀도를 가지도록 고정 소수점 연산을 이용하여 구현하였다. 고정 소수점 연산 결과 원 신호와의 차이를 나타내는 SNR (signal to noise ratio) 이 모두 65 dB 이상으로서 정밀도에 문제가 없는 것으로 판단된다. 또한 sample by sample 비교에서 surround 채널의 경우에 약간의 오차가 발생하였는데 이는 surround 채널이 여러 filtering 과정을 거칠 때 발생하는 오차인 것으로 생각된다. Surround 채널의 경우에도 오차는 크지 않아서 정밀도에서는 문제가 없는 것으로 판단된다. 또한 Dolby Pro Logic decoding을 위해서는 54MHz clock을 사용하면 주어진 시간에 복호가 가능함을 알 수 있었다. 그러나 3-D surround 효과 등의 부가 기능을 추가하려면 54 MHz에서 동작하는 ARM-7으로는 실시간 구현이 힘들다. 따라서 음향 복호기 뿐 아니라 향후 멀티미디어 기기에 활용될 수 있는 고성능의 DSP 코어의 개발이 필수적이다.

앞으로 Dolby AC-3나 Dolby Pro Logic 복호기 뿐 만 아니라 MPEG-2 오디오 복호기도 하나의 chip에 집적화하는 연구가 있어야 하겠고 궁극적으로는 audio/video 복호기를 하나의 chip에 집적화하는 연구가 수행되어야 할 것으로 생각된다.

참 고 문 헌

[1] *Grand alliance HDTV system specification*, Dec. 1994.

[2] C. C. Todd, G. A. Davidson, M. F. Davis, L. D. Fielder, B. D. Link and S. Vernon, "AC-3: Flexible perceptual coding for audio transmission and storage," *96th Convention of Audio Engineering Society*, Feb., 1994.

[3] *Digital Audio Compression (AC-3) Standard*, May 1995.

[4] *International organization for standardisation: Coding of moving pictures and associated audio*, Nov. 1994.

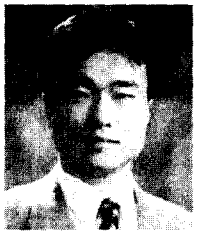
[5] R. Dressler, "Dolby Pro Logic surround

decoder, principles of operation," *URL: http://www.dolby.com*, March 1996.

[6] *ARM software development toolkit v. 2.0: Windows Toolkit Guide*, Advanced RISC machines, June 1995.

[7] Chang Woo Lee, Sang Keun Lee and Jae Moon Jo, "Dolby AC-3 audio decoder IC with ARM-7 core," *Proc. of the 1998 International Technical Conference on Circuit/Systems, Computer and Communications*, pp. 1371-1374, July 1998.

이 창 우(Chang Woo Lee) 정회원



1988년 2월 : 서울대학교 제어계측공학과 (공학사)
1990년 2월 : 서울대학교 대학원 제어계측공학과 (공학석사)
1996년 2월 : 서울대학교 대학원 제어계측공학과 (공학박사)
1996년 3월~1997년 8월 : 삼성전자 신호처리 연구소 선임연구원
1997년 9월~현재 : 가톨릭대학교 컴퓨터·전자공학부 조교수
<주관심 분야> 영상압축, 영상전송 및 영상신호처리, 통신 및 음향 신호처리

이 상 근(Sang Keun Lee) 정회원

1991년 2월 : 부산대학교 전자공학과 (공학사)
1993년 2월 : 부산대학교 전자공학과 (공학석사)
1993년 3월~현재 : 삼성전자 중앙연구소 정보미디어 연구소 선임연구원

조 재 문(Jae Moon Jo) 정회원

1984년 2월 : 서울대학교 전기공학과 (공학사)
1986년 : 한국과학기술원 전기 및 전자 공학과 (공학석사)
1991년 : 한국과학기술원 전기 및 전자공학과 (공학박사)
1989년 6월~현재 : 삼성전자 중앙연구소 정보미디어 연구소 수석연구원