디지탈/아날로그 겸용 이동통신 단말기를 위한 오디오/데이타 프로세서의 실시간 구현에 관한 연구

A Study of Real-Time Implementation of Audio/Data Processor for Digital/Analog Dual mode Mobile Phone

변 경 진*, 김 종 재*, 한 기 천*, 유 하 영*, 차 진 종*, 김 경 수*
(Kyung-Jin Byun*, Jong-Jae Kim*, Ki-Chun Han*, Hah-Young Yoo*, Jin-Jong Cha*, Kyung-Su Kim*)

요 약

본 연구는 현재 디지탈 방식의 이동통신에서 사용되는 디지탈/아날로그 겸용 단말기에서 아날로그 방식을 지원하기 위한 오디오/데이타 프로세서를 ETRI DSP를 이용하여 실시간 구현하는 것에 대한 것이다. 오디오/데이타 프로세서는 단말기가 아날로그 방식으로 동작할 때 광대역 데이타 처리, 오디오신호 처리 및 demodulation, data rate conversion 기능을 수행한다. 이와같은 기능은 어셈블리 언어로 프로그램되어 디지탈 방식에서 사용되는 보코더 프로그램과 함께 ETRI DSP에 탑재되었다. 즉 하나의 하드웨어를 이용하여 디지탈 방식의 보코더와 아날로그 방식의 오디오/데이타 프로세서를 함께 구현 함으로써 하드웨어의 효율성을 극대화 하여 기존의 아날로그 전용의 단말기와의 경쟁력을 가질 수 있도록 하였다.

ABSTRACT

In this paper, the implementation of audio/data processor using ETRI DSP to support analog mode in digital/analog dual mode mobile phone is presented. Audio/data processor performs the wideband data processing, audio signal processing, demodulation function, and data rate conversion when it is operated in analog mode. These functions are programmed in assembly language, and then loaded to ETRI DSP together with vocoder program for the digital mode operation. This is a very efficient implementation of the dual mode cellular phone ASIC since the vocoder for the digital mode and audio/data processor for the analog mode are programmed together in the same hardware.

I. 서 론

현재 상용화 되어 있는 디지탈 방식의 이동통신에서 사용하는 단말기는 이중모드를 지원하도록 되어 있으므로 디지탈 방식인 CDMA(Code Division Multiple Access) 방식의 시스템에 접속할 수도 있고, 기존의 아날로그 방식인 AMPS(Advanced Mobile Phone Service) 방식의 시스템에도 접속할 수 있도록 되어 있다.[1] AMPS 모드에서 베이스밴드 신호처리를 하는 핵심 부품인 오디오/데이타 프로세서는 기존의 아날로그 방식의 단말기에서는 상용화되어 있는 외국의 전용 칩을 사용하고 있다.[4][5] 그러나 국내의 단말기 제조업체에서도 외국의 단말기에 대한 경쟁력을 갖기 위해 상용화되어 있는 부품들 중에서 핵심 부품이라 할 수 있는 오디오/데이타 프로세서를 국산화 하기 위하여 많은 노력을 기울여 왔다.

디지탈 방식의 이동통신에서 사용되는 디지탈/아날로그 겸용의 단말기에서는 CDMA모드에서 사용하는 하드웨어를 AMPS모드에서도 공유할 수 있어야 전체 소자의 갯수를 줄이고, 단말기의 부피를 적게하고, 전력소모도 줄일 수 있게되어 기존의 아날로그 전용 단말기와의 경쟁력을 가질 수 있다.

그러므로 본 연구에서는 AMPS 모드에서 사용되는 오디오/데이타 프로세서를 CDMA모드에서 보코더로 사용되는 DSP에 프로그램으로 구현하여 하나의 하드웨어를 이용하여 두가지 모드를 지원할 수 있도록 함으로써 하드웨어의 사용 효율성을 극대화 하였다. 여기서 사용된

*한국전자통신연구원 반도체연구단
접수일자 : 1996년 12월 9일

DSP는 ETRI에서 개발한 DSP로써 CDMA시스템에서 사용되는 음성부호화기인 QCELP 보코더를 개발하기 위해 개발된 16 bit 고정 소수점형 DSP이다.[6][7]

본 연구에서 구현한 오디오/데이타 프로세서는 독립된 상품이 아닌 다른 상품 칩과 통합되어 디지탈/아날로그 겸용 단말기에 사용하기 위하여 ETRI에서 개발하고 있는 베이스밴드 신호처리 IC인 IMM2 (Integrated Mobile Station Modem 2)라는 하나의 ASIC으로 개발하는 것을 목표로 하였다. IMM2 IC는 이미 상용화 되어 있는 디지탈/아날로그 겸용 단말기인 Qualcomm사의 QCP-800에 사용되는 MSM2 (Mobile Station Modem 2) IC와 호환성을 갖도록 설계 되어 있다. 그러므로 오디오/데이타 프로세서는 실시간 구현에 있어서 IMM2라는 전체 시스템의 사양에 맞추기 위하여 동작속도 및 입출력 인터페이스 등에 대하여 많은 제약을 받았다.

제 II절에서는 오디오/데이타 프로세서의 전체적인 개요 및 주요한 블럭 들에 대한 기능설명을 하였고, 제III절에서는 ETRI DSP를 이용하여 오디오/데이타 프로세서를 실시간 구현하고 실시간 시험을 위하여 실험환경을 구성한 것에 대하여 논하고, 마지막으로 IV절에서 결론을 맺는다.

II. 오디오/데이타 프로세서의 기능

2.1 개 요

오디오/데이타 프로세서는 그림 1의 오디오 프로세서 기능과 그림 2의 데이타 프로세서 기능으로 구분할 수 있다.

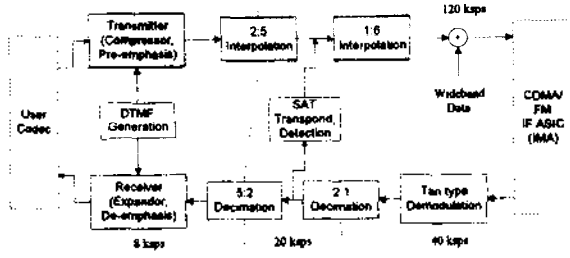


그림 1. 오디오 프로세서 블럭도

우선 오디오 프로세서의 구성은 크게 세부분으로 나누어 볼 수 있다. 첫째, 오디오 프로세서의 수신단 입력이 demodulation된 신호가 아니라 단지 down conversion된 신호를 샘플링한 신호이기 때문에 이신호를 가지고 디지탈 신호처리 방법으로 demodulation과정을 거쳐 원래의 음성 신호로 만들어야 한다.

둘째는 오디오 프로세서와 IMA2 ASIC (Integrated Mobile Station Analog ASIC)에서 처리하는 data rate이 다르기 때문에 데시메이션 혹은 인터폴레이션 과정을 거쳐야 한다. 즉 오디오 프로세서의 수신단으로 입력되는 40K SPS(Symbol per Sec.)의 데이타는 데시메이션 과정을 거쳐 8K SPS의 데이타로 만든 후에 de-emphasis 및 expansion 과정을 수행하게 되고, 송신단에서 출력되는 8K SPS의 데이타는 인터폴레이션 과정을 거쳐 120K SPS의 신호로 만들어 IMA2 ASIC으로 보내어 진다.

셋째는 오디오 프로세서의 중심 기능블럭으로써 송신부에는 pre-emphasis, compressor가 있고, 수신부에는 de-emphasis, expandor가 있으며, 그 외에도 DTMF 생성블럭, SAT 신호 검출블럭이 있다.

반면, 데이타 프로세서의 기능flow는 그림2와 같다. 즉 이동국에서 기지국으로 data 전송시 전송할 데이타를 CPU에서 BCH encoding을 수행한 후 각 채널에 따른 data format에 일치되도록 data formatting을 한 다음 manchester encoding을 수행하여 modulator단으로 입력시킨다.

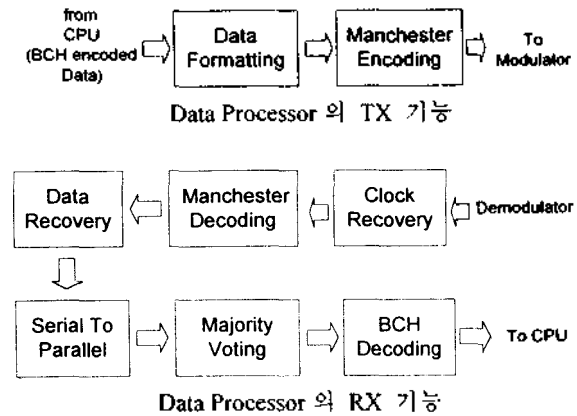


그림2. 데이타 프로세서의 기능블럭도

이동국에서 기지국으로 부터의 데이타를 수신할 경우에는 demodulator단으로 부터 data를 전송받아 PLL을 이용하여 clock을 복구하고 복구된 clock을 이용하여 manchester decoding을 수행하면 10 Kbps의 data를 추출해낼 수 있다. 추출된 data를 parallel data로 전환한 다음 majority voting을 수행한다. Majority voting을 수행한 40 bit의 command data를 BCH decoding하여 최종 필요한 28 Bit의 data를 추출한다.

2.2 Demodulation

오디오 프로세서에서는 IF 처리용 ASIC인 IMA2로부터 modulation되어있는 디지탈 샘플을 받아서 arc tangent계산을 하여 demodulation을 하게 된다. IMA2에서는 RF단에서 오는 신호를 down conversion한 후, 아날로그 신호를 ADC를 거쳐 8bit의 binary형태의 디지탈 신호로 만들어 출력한다. IMA2에서는 I 신호와 Q 신호 두개를 보내 주는데 I 신호와 Q 신호는 $\pi/2$의 위상차를 갖는다. 그림 3에 arc tangent를 이용하여 demodulation하는 과정의 블럭도를 나타 내었다. 그림 3에서의 X(k)와 Y(k)

는 각각 IMA2에서 오는 I신호, Q신호로 생각할 수 있다. 그림 3에서의 arc tangen 블럭의 계산은 3단계로 나눌 수 있다. 첫째 Q/I의 계산을 한 후 둘째로 atan(Q/I)를 계산한 다음에 마지막으로 미분항 계산을 수행하면 된다.

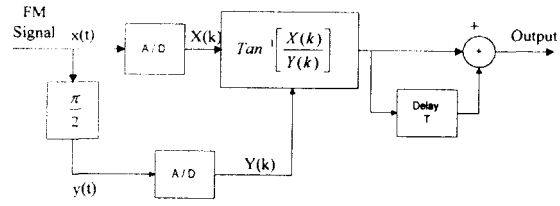


그림 3. Arc tangent를 이용한 demodulation

위와같이 arc tangent를 이용하여 demodulation이 되는 과정은 다음과 같다. FM modulation된 신호는 식 (1)과 같이 표현된다.

$$I = A_c \cos 2\pi \left[f_c t + k_f \int_0^t s(\tau) d\tau \right]. \tag{1}$$

여기서 I신호에 $\cos 2\pi f_c t$ 항을 곱하여 다음과 같이 down conversion 하게 된다.

$$\begin{aligned} & I \times A_c \cos 2\pi f_c t \\ & = A_c \left[\cos 2\pi (f_c t + k_f \int_0^t s(\tau) d\tau) \times \cos 2\pi f_c t \right] \\ & = \frac{A_c}{2} \left[\cos 2\pi (2 f_c t + k_f \int_0^t s(\tau) d\tau) + \cos 2\pi (k_f \int_0^t s(\tau) d\tau) \right]. \end{aligned} \tag{2}$$

위의 신호를 저역통과 필터링 하면 앞의 항은 없어지고 뒤의 낮은 주파수 성분의 신호만 남게되어 down conversion 된 최종적인 I 신호는 다음과 같이 된다.

$$I = \frac{A_c}{2} \cos 2\pi (k_f \int_0^t s(\tau) d\tau) \tag{3}$$

위의 I 신호와 $\pi/2$ 만큼의 위상차를 갖는 Q 신호는 다음과 같다.

$$Q = \frac{A_c}{2} \sin 2\pi (k_f \int_0^t s(\tau) d\tau) \tag{4}$$

위와같은 I, Q 신호는 IMA2에서 Analog to Digital변환하여 오디오 프로세서의 입력으로 넘겨주게 된다. 그러면 오디오 프로세서에서는 다음과 같이 I, Q 신호를 가지고 arc tangent 방법을 이용하여 modulation하기 전의 신호인 s(t)를 복원하게 된다.

$$\frac{Q}{I} = \frac{\sin 2\pi (k_f \int s(\tau) d\tau)}{\cos 2\pi (k_f \int s(\tau) d\tau)} = \tan 2\pi (k_f \int s(\tau) d\tau) \tag{5}$$

그러므로

$$\tan^{-1} \left(\frac{Q}{I} \right) = 2\pi k_f \int s(\tau) d\tau \tag{6}$$

위의 식에서 양변을 미분하면 원래의 신호인 s(t)를 얻을 수 있다.

$$\frac{d}{dt} \left(\tan^{-1} \left(\frac{Q}{I} \right) \right) = s(t) \tag{7}$$

위의 식에서 얻어진 s(t)는 $-\pi \sim +\pi$ 사이의 값을 갖게 된다.

이와같이 tangent형식을 이용하여 디지탈 신호처리 방식으로 FM demodulation을 하게되면 선형성이 좋아 부품의 왜곡 복조가 가능하고, AM 억제 능력이 있으므로 입력단의 AGC가 필요없으며, 협대역 FM에서부터 광대역 FM 까지 넓게 사용할 수 있는 등의 장점이 있다.

2.3 Data Rate Conversion

그림 1의 오디오 프로세서에서는 data rate를 조정하기 위해 두번의 인터폴레이션과 두번의 데시메이션 과정을 수행하게 되는데 인터폴레이션 과정에서는 오디오 프로세서의 코아 블럭에서 처리된 8KSPS의 음성신호를 인터폴레이션을 취하여 120 KSPS의 신호로 만들어 IMA2로 보내게 된다. 그림 4에 인터폴레이션 블럭을 자세히 나타내었다.

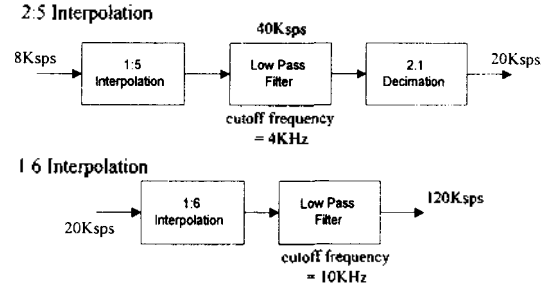


그림 4. 인터폴레이션 블럭

인터폴레이션의 수행 방법에는 여러가지가 있으나 주로 사용되는 방법은 영 삽입 방법이다. 영을 삽입하여 인터폴레이션을 수행하고 나면 주파수 영역에서 원래의 이미지가 반복 되기 때문에 안티-이미지 용도의 저역통과 필터를 사용하여 필요없는 부분은 제거한다. 그리고 그림 1의 두번의 인터폴레이션 중에서 2:5 인터폴레이션은 비정수 인터폴레이션이므로 1:5로 인터폴레이션한 후에 다시 2:1로 데시메이션하여 구현하는 방법을 사용한다.

데시메이션 과정에서는 IMA2에서 40KSPS의 I, Q데이타를 받아서 arc tangent 블럭을 수행한 후 2:1 데시메이션과 5:2 데시메이션을 거쳐서 8KSPS의 데이타로 만든 후 오디오 프로세서의 코아 블럭을 처리하게 된다. 데시메이션을 할 시에는 알리아싱을 피하기 위하여 안티-알리아싱 용의 저역통과 필터링을 수행하여야 한다. 5:2

데시메이션은 비정수 데시메이션이므로 그림 5과 같이 우선 1:2 인터폴레이션을 거친 후 5:1 데시메이션을 수행한다.

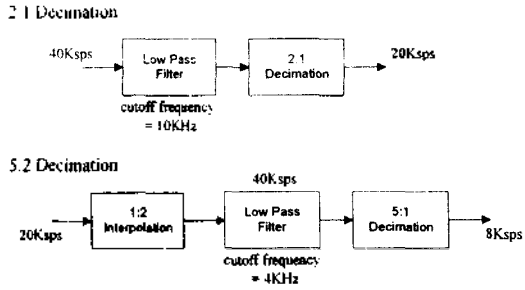


그림 5. 데시메이션 블럭

2.4 SAT 신호의 검출

그림 1의 오디오 프로세서 블럭 중에서 중요한 기능 중의 하나가 SAT(Supervisory Audio Tone)신호 검출 기능이다. AMPS 방식에서는 이동국과 기지국 간의 접속상태를 감지하기 위하여 기지국에서 이동국으로 음성 채널에 음성신호와 함께 SAT 신호를 보내게 되어 있는데 이 SAT 신호는 5970, 6000, 6030 Hz 중 하나를 사용하게 되어 있다. SAT신호 감시 기능에는 SAT transponding 기능과 SAT detection 기능이 있다. SAT transponding은 이동국에서는 기지국에서 보낸 SAT 신호를 그대로 기지국으로 다시 보내어 기지국에서 자신이 보낸 SAT 신호와 되돌아온 SAT 신호의 주파수가 일치하는가를 감시하여 접속상태를 확인하는 기능이다.

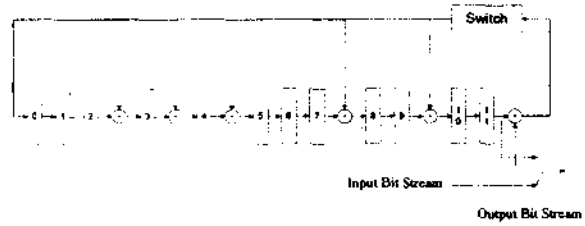
SAT detection 기능은 이동국에서, 기지국에서 보내온 SAT 신호의 주파수를 검출하여 이미 기지국에서 광대역 데이타 전송시 보내온 SAT color code와 비교하여 일치하는가를 확인하는 기능이다. SAT detection의 수행은 연속적으로 수행할 필요는 없지만 최소한 250 msec 마다 한 번씩은 이루어져야 한다. 표 1에 SAT detection시의 주파수 범위를 나타내었다.

표 1. SAT detection시의 주파수 범위

입력신호의 측정된 주파수	측정된 SAT신호의 결정	
$f \le f_1$	불확실한 SAT	$f_1 = 5955 \pm 5Hz$
$f_1 \le f < f_2$	SAT = 5970	$f_2 = 5985 \pm 5Hz$
$f_2 \le f < f_3$	SAT = 6000	$f_3 = 6015 \pm 5Hz$
$f_3 \le f < f_4$	SAT = 6030	$f_4 = 6045 \pm 5Hz$
$f_4 \le f$	불확실한 SAT	
SAT 수신불능	불확실한 SAT	

2.5 데이타 프로세서

데이타 프로세서는 그림 2와 같이 TX 부분과 RX 부분으로 나눌 수 있다. TX 부분에서는 우선 BCH encoding을 하게되는데 여기서는 총 계산량의 제한 때문에 DSP에 구현하지 못하고, CPU에서 수행을 하였고, RX 부분의 BCH decoding만을 DSP를 사용하여 구현하였다. BCH encoder는 총 bit수가 63 bit이고 information bit가 51 bit, error check bit가 12 bit인 (63, 51) double error correction BCH code를 사용한다. AMPS방식은 forward channel의 command size가 28 bit이므로 실제는 상위 23 bit는 "0"인 것으로 간주한다. BCH encoder의 구성 및 generator polynomial은 그림 6과 같다.



Generator Polynomial :

$$X^{12} + X^{10} + X^8 + X^5 + X^4 + X^3 + 1$$

그림 6. BCH encoding block diagram

BCH encoder의 동작 순서는 다음과 같다.

(1)매 clock마다 shifter register를 right shift시키고 동시에 information 28 bit [0-27]를 입력시킨다.

(2)Switch 를 open한다.

(3)12개의 register에 들어있는 parity bit들을 right shift하여 information bit의 후미에 추가하여 40 bit를 만든다. 위와같이 10Kbps BCH encoding된 40 bit data를 manchester encoding하여 20 Kbps data로 변환한다.

데이타 프로세서의 RX부분에서는 AMPS 단말기의 RX part에서입력된 20 Kbps의 manchester coding된 data에서 20KHz clock을 추출하는 clock recovery를 수행한후 manchester encoding 의 역과정을 수행하여 data recovery를 하게 된다. Majority voting 과정에서는 forward channel

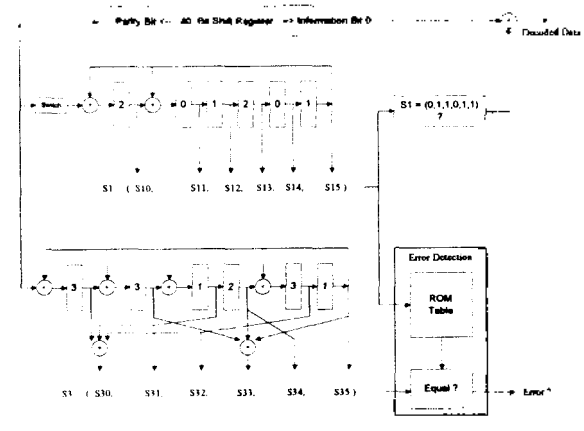


그림 7. BCH decoding block diagram

의 data format에 따라 연속적으로 입력되는 40 bit data를 5 번 반복적으로 받아들여 각 bit 마다 3번 이상인 data를 취한다. 마지막으로 majority voting된 40 bit data를 이용하여 그림 7과 같이 BCH decoding을 수행한다.

BCH decoder의 동작순서는 다음과 같다.

- 전송되어 온 40 bit 입력 data를 information bit의 LSB부터 차례로 S1측 shift register와 S3측 shift register에 통과시켜 S1과 S3를 얻는다.
- S1의 3제곱값을 ROM table을 읽어서 얻는다.
- Error의 발생갯수는 다음과 같다.

  $S3=(S1)^3$인 경우 $S1=0$ 이면 No error

  $S1\neq 0$ 이면 Single error

  $S3\neq(S1)^3$인 경우 두개 이상의 Error
- Error가 1개이상일 경우 switch를 open한다.
- Data를 40 bit shift register에 통과시키고 S1 = (0, 1, 1, 0, 1, 1)일 경우 해당 bit가 error발생 bit이다.
- S15와 XOR를 수행하여 error를 정정한다.

Ⅲ. 오디오/데이타 프로세서의 구현

3.1 디지탈 신호처리 프로세서

본 연구에서 오디오/데이타 프로세서의 실시간 구현을 위하여 사용된 디지탈 신호처리 프로세서는 상용화된 범용 DSP가 아니고 ETRI에서 개발된 ES-C2340(ETRI DSP)라는 DSP이다. ES-C2340는 CDMA 음성 부호화기 개발을 위하여 개발된 16bit 고정 소수점형 DSP로서 다음과 같은 특징을 갖는다.[2][3]

- 40 MIPS (80 MHz 외부 클럭 사용)의 처리속도
- 16 * 16 bit 곱셈 및 더하기(MAC) 및 32 bit data load를 한 사이클에 수행
- 2개의 36 bit accumulator
- 12K * 24bit 프로그램 ROM 내장
- 최대 64K * 24 bit 외부 프로그램 메모리 사용가능
- 2K * 16 bit 데이타 RAM 및 3K * 16 bit 데이타 ROM 내장
- 1개의 serial port와 1개의 parallel port
- 100 pin TQFP (Thin Quadrature Flat Package)

3.2 전체 프로그램의 구성

ETRI DSP의 어셈블리 언어로 구현된 오디오/데이타 프로세서 프로그램의 전체 흐름도는 그림 8과 같다.

흐름도의 프로그램은 SO(Serial Output) interrupt를 기준으로 수행되는데 SO sample(8K) 2개(250 usec)마다 한번씩 수행된다. SO interrupt가 2번 걸리면 RX I, Q data (40K) 10개가 arc tangent블럭에서 처리되어 demodulation한 결과를 출력하게 된다.

Demodulation된 신호를 가지고 우선은 데이타 프로세서에서 처리를 하여 데이타의 여부를 확인하여 광대역 데이타이면 데이타 프로세서의 계속적인 동작이 진행되고, 광대역 데이타가 아니면 오디오 프로세서의 수신단이 수행되어 최종 출력인 음성신호가 codec으로 나가게 된다. 오디오 프로세서의 수신단이 수행된 후 송신부가 동작되는데 송신부에서는 codec에서 입력된 음성신호를 처리하여 일정한 버퍼에 저장을 하고 실제로 외부로 출력하는 것은 TX interrupt(10K)에 의해 이루어진다. 그리고 데이타 프로세서의 송신단은 CPU의 TX interrrupt (10K * 16bit)에의해 동작되므로 interrupt service routine에서 call 하게 되어있다. 이러한 프로그램의 실시간 구현시의 가장 큰 문제점은 오디오/데이타 프로세서를 동시에 수행하여야 하므로 많은 계산량으로 인하여 목표한 시간 내에 프로그램을 수행시키기가 어렵다는 것이다.

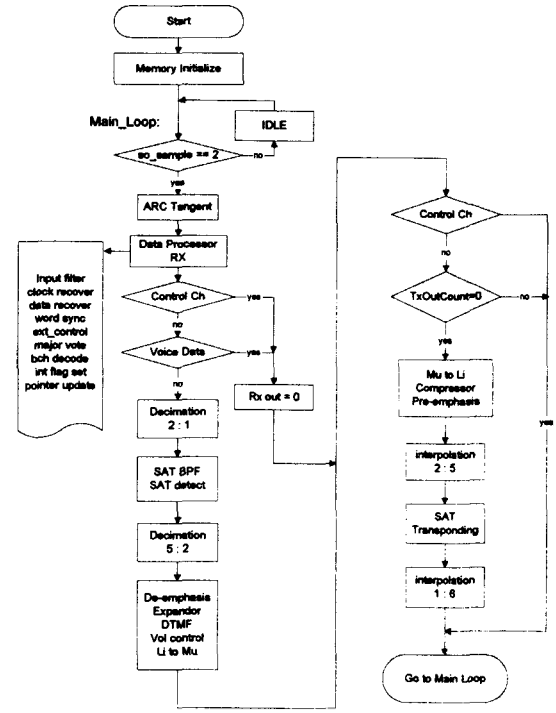


그림 8. 오디오/데이타 프로세서 전체 흐름도

3.3 계산량 감소를 위한 구현

프로그램의 수행에서 오디오 프로세서가 동작 중일때에도 demodulation된 신호가 광대역 데이타인가를 계속 감시하여야 하므로 이때가 가장 계산량을 많이 소비하게 된다. 오디오 프로세서에서는 demodulation, data rate conversion 블럭이 계산량이 많고, 데이타 프로세서에서는 오디오 프로세서와 동시에 동작하는 블럭만이 전체 계산량에 영향을 미치므로 이 부분에 대한 계산량을 최소화 하였다.

3.3.1 Demodulation 블럭의 구현

Arc tangen 블럭의 계산은 3단계로 나눌 수 있다. 첫째 Q/I의 나눗셈 계산을 한 후 둘째로 atan(Q/I)를 계산한

다음에 마지막으로 미분항 계산을 수행하면 된다. 우선 Q/I의 나눗셈 계산은 어셈블리 언어로 구현할 시 많은 계산량이 요구 되므로 128개로 구성된1/I의 ROM 테이블을 이용하여 1/I 값을 구한 후 (1/I)•Q를 계산하여 Q/I 값을 구한다. Arc tangent의 계산도 테이블 방식을 사용하여 계산량을 최소화 하는데 중점을 두었다.

Arc tangent 테이블은 0~$\pi/2$ 사이의 arc tangent 값을 128개의 데이타로 구성하였고 Q/I 값의 각 영역에 따른 테이블 할당은 Q/I의 조합에서 나타날 수 있는 확률에 따랐다. 이렇게 구성된 테이블을 포인팅하기 위해서는 Q/I의 계산된 값을 적당히 정규화할 필요가 있다. 이러한 정규화 과정도 역시 Q/I 값에 따른 arc tangent 테이블의 할당된 스텝 수에 따라 표 2와 같이 수행하였다.

표 2. Arc tangent 테이블을 포인팅하기 위한 정규화 값

Q/I의 정수 값	Q/I의 binary 표현		Step 수	정규화 값
	정수	소수		
0	0000	xxxx xx	64	*64
1	0001	xxxx xx	32	*32
2~3	0010	xxxx	16	*8
	0011	xxxx		
4~7	0100	xxxx	8	*2
	0101	xxxx		
	0110	xxxx		
	0111	xxxx		
8~15	1000	xxxx	4	÷2
	...			
	1111	xxxx		
16~	00010000	xxxx	4	÷32
	...			
	01111111	xxxx		

그리고 테이블 값은 0~$\pi/2$ 사이의 값만 이용하지만 실제의 I, Q 데이타의 값은 ±I, ±Q에 따라서 $-\pi$~$+\pi$ 사이의 값을 갖게 된다. 그러므로 계산은 모두 양수로 변환하여 한 후 표 3에 따라 최종 값을 구하게 된다. 표 3의 맨 우측 열에는 $-\pi$~$+\pi$ 사이의 값을 다시 $-\pi/2$~$\pi/2$ 사이의 값으로 변환한 것을 표시 하였다.

표 3. I, Q 데이타에 따른 θ의 영역

Q	I	구간	결과	최종결과
−	−	180°~270°	$-\pi+\theta$	$(-\pi+\theta)+\pi=\theta$
−	+	270°~360°	$-\theta$	$-\theta$
+	+	0°~90°	θ	θ
+	−	90°~180°	$\pi-\theta$	$(\pi-\theta)-\pi=-\theta$

3.3.2 Data rate conversion의 구현

오디오 프로세서에서 인터폴레이션과 데시메이션 블럭의 구현은 많은 계산량이 요구된다. 이중에서 대부분의 계산량은 LPF를 수행하는데 소비 되는데, 앞단에서 영을 삽입한 신호를 그대로 LPF하게 되면 많은 계산량을 필요로 하게된다. 그러므로 실제 구현에서는 삽입된 영의 신호는 LPF계산에서 제외할 수 있도록 구현하였다. 이러한 과정을 편의상 1:3 Interpolation을 예로 들어 설명하면 다음과 같다. 이때 FIR LPF의 tap수는 9로 가정하였다.

표 4. Interpolation과 LPF

DI	0	0	**D0**	0	0	**D−1**
			b0	b1	b2	**b3**
		b0	**b1**	b2	b3	**b4**
	b0	b1	**b2**	b3	b4	**b5**
	0	0	**D-2**	0	0	D-3
	b4	b5	**b6**	b7	b8	
	b5	b6	**b7**	b8		
	b6	b7	**b8**			

위의 표4에서 D0, D-1 ... 은 입력 신호로써 filter memory에 저장되고, b0, b1 ... 은 filter 계수로써 역시 memory에 저장되어 있다. 그리고 D0와 D-1사이의 0은 1:3 interpolation을 위한 것이다. 이렇게 영을 삽입한 후 FIR LPF를 수행하여 1:3 interpolation을 취하면 아래와 같이 1개의 입력 D0에 대하여 3개의 출력이 생성된다. 위의 표4를 참조하면 쉽게 알 수 있다.

out1 = D0 * b0 + 0 * b1 + 0 * b2 + D − 1 * b3 + ... 0 * b8
out2 = 0 * b0 + D0 * b1 + 0 * b2 + 0 * b3 + D − 1 * b4 + ... 0 * b8
out3 = 0 * b0 + 0 * b1 + D0 * b2 + 0 * b3 + 0 * b4 + D − 1 * b5 + ... D − 2 * b8

결과적으로 위의 3개의 출력을 얻기위한 총계산량은 다음과 같다.

- MAC : 9 * 3 = 27회
- filter memory shift : 2 * 8 * 3 = 48회
- 총 계산량 : 75회

앞에서 설명한대로 interpolation을 위하여 삽입된 영의 신호를 그대로 LPF하게되면 많은 계산량을 필요로하게 된다. 그러나 앞의 표 4에서 보면 영의 신호와 필터 계수를 곱하는 동작은 생략할 수 있고, 또한 filter memory를 shift 하는 동작도 생략할 수 있음을 알 수 있다. 결국 표 4에서 bold체로 표시된 것에 대해서만 곱셈을 하면 아래와

같이 출력을 얻을 수 있다.

$$out1 = D0 * b0 + D-1 * b3 + D-2 * b6$$
$$out2 = D0 * b1 + D-1 * b4 + D-2 * b7$$
$$out3 = D0 * b2 + D-1 * b5 + D-2 * b8$$

이와같은 방법으로 구현하면 계산량은 MAC동작 3*3=9회 만이 필요하게 되어 앞의 방법에 비해 많은 개선이 이루어짐을 알 수 있다. 이 방법을 사용하면 interpolation의 비가 클수록, filter의 tap 수가 클수록 많은 양의 계산량 감소 효과를 얻을 수 있다. 그리고 이 방법을 사용했을 때의 또다른 장점은 표 4에서 보면 입력 신호가 저장되는 filter memory를 적게 사용할 수 있다는 점이다. 앞에서 구현된 방법으로는 filter의 tap수 만큼의 filter memory가 필요하지만 여기서는

filter memory = filter의 tap 수/interpolation rate

만으로 구현이 가능하다. 단 이 방법을 사용하기 위해서는 필터계수의 배치순서를 바꾸어 주어야한다. 즉 원래 b0, b1, b2,...b8의 순서로 배치된 것을 b0, b3, b6, b1, b4, b7, b2, b5, b8로 배치 해야 구현이 간단해진다. 또한 filter의 tap 수를 interpolation rate의 정수배로 해야만이 구현이 간단해진다.

3.3.3 프로그램 사양 및 계산량

오디오/데이타 프로세서의 어셈블리 프로그램의 전체 계산량은 표 5에 나타 내었고 프로그램의 사양을 표 6에 나타 내었다.

표 5. 오디오/데이타 프로세서의 계산량

	Function	Computation
RX	arc tangent	2.80 MIPS
	decimation 2:1	1.58 MIPS
	SAT detect	1.78 MIPS
	decimation 5:2	1.14 MIPS
	de-emphasis, expandor, DTMF	2.25 MIPS
	Data Processor	1.60 MIPS
sub total		*11.15 MIPS*
TX	pre-emphasis, compressor	1.28 MIPS
	interpolation 2:5	0.72 MIPS
	interpolation 1:6	2.24 MIPS
	Data scaling	1.69 MIPS
	Data Processor	1.87 MIPS
sub total		*7. 80 MIPS*
	Interrupt	8.00 MIPS
Total		**26.95 MIPS**

오디오/데이타 프로세서의 전체 계산량은 프로그램이 최대의 계산량으로 수행될 때를 기준으로 시뮬레이션 상에서 측정한 것이다. 최대의 계산량이 요구될 때는 오디오 프로세서가 동작 하면서 데이타 프로세서는 voice channel 상에서의 광대역 데이타를 검출하고 있을 때이다. 그러므로 표 5에 나타나 있는 데이타 프로세서의 계산량은 전체 데이타 프로세서의 계산량이 아니고, 데이타 프로세서가 수신된 데이타가 광대역 데이타인가를 판정하는 부분에 대한 계산량만을 표시한 것이다. 표 6은 오디오/데이타 프로세서 프로그램의 전체 사양이다. 프로그램 ROM은 12K 중에서 오디오/데이타 프로세서 프로그램이 3.3K를 차지하고, 나머지 부분에는 CDMA 모드의 보코더 프로그램이 탑재되어 있다. 데이타 ROM은 전체 3K 중에서 0.8K 정도만을 오디오/데이타 프로세서 프로그램이 사용한다. 데이타 RAM은 두개의 프로그램이 공유할 수 있는 부분이므로 오디오/데이타 프로세서 프로그램이 2K전부를 사용할 수 있으나, 실제로 사용하는 영역은 1.2K 정도이다. 그리고 오디오/데이타 프로세서 프로그램은 DSP의 최대 계산속도인 40MIPS보다 훨씬적은 27MIPS의 계산량으로 실시간 수행을 할 수 있도록 설계되었다.

표 6. 오디오/데이타 프로세서 프로그램 사양

	오디오/데이타 프로세서	DSP전체
Program ROM	2,184/1,185 (3,369)	12K * 24 bit
Data RAM	755/366 (1,121)	2K * 16 bit
Data ROM	820/64 (884)	3K * 16 bit
MIPS	27 MIPS	40 MIPS

3.4 시험환경 구성 및 시험결과

DSP를 이용하여 구현된 오디오/데이타 프로세서를 실시간 시험하기 위하여 그림 9와 같이 시험 환경을 구성하였다. 오디오/데이타 프로세서의 입출력은 하나의 serial port와 하나의 parallel port로 구성되어 있다. Serial port는 음성 코덱과 연결되어 음성신호의 입출력을 담당한다. Parallel port는 세가지의 용도로 사용되는데 첫째, IMA2에서 오는 RX I, Q 데이타의 수신, 둘째 DSP에서 만들어진 TX 데이타의 송신, 셋째 CPU와의 통신에 사용된다. 이와같이 오디오/데이타 프로세서와 주변의 회로들을 연결해 주기위하여 FPGA를 이용하여 인터페이스 블럭을 설계하였다. 오디오/데이타 프로세서를 실시간 시험하기 위해 FPGA에 설계된 인터페이스 블럭은 최종적으로 오디오/데이타 프로세서와 함께 IMM2 ASIC에 집적화되었다. 설계된 인터페이스 블럭은 FM_RX block, FM_TX block, CPU_IO block, PIO_interface block으로 구성되어 있다.

FM_RX block에서는 FM_RX block은 40 KHz로 입력되는 8 bit serial I data와 Q data를 오디오/데이타 프로세

서인 DSP에 전달하는 기능을 한다. 오디오/데이타 프로세서에서는 8 KHz 단위로 RX data를 처리하기 때문에 FM RX block에서는 RX I/Q data를 buffer에 저장한 후 8KHz마다 한 번씩 DSP에 전송한다.

FM_TX block에서는 오디오/데이타 프로세서에서 생성한 TX data를 IMA2 ASIC 측으로 전송하는 블럭이다. IMA2 ASIC의 TX의 part는 120 KHz로 8 bit parallel data를 받아들이게 되어 있다. 그러나 120 KHz의 속도로 DSP에서 data가 송신될 경우 DSP에 많은 load가 걸려 오디오/데이타 프로세서의 프로그램을 제시간 안에 수행할 수 없게 된다. 그러므로 TX interface logic을 부가하여 DSP로부터는 10 KHz마다 한 번씩 TX data를 받아 buffer에 저장하고 이를 다시 120 KHz의 TX_CLK에 맞게 IMA2 측으로 전송한다.

CPU_IO block에서는 오디오/데이타 프로세서와 전체 system을 control 하는 CPU사이의 각종 command와 data의 상호이동을 조정하는 기능을 한다. 즉 CPU측에서 PIO_BUS 사용 요구가 비동기적으로 발생하는데 이를 처리하기 위한 block이다. PIO_interface block에서는 RX, TX, CPU part에서 요구하는 PIO bus의 사용을 서로 충돌이 일어나지 않게 조정하는 기능을 가진 block이다.

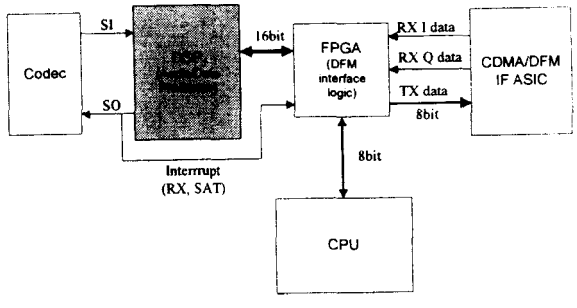


그림 9. 오디오/데이타 프로세서의 주변인터페이스

본 논문에서 구현한 오디오/데이타 프로세서는 실제로 실시간 테스트를 위하여 테스트 보드를 제작하였다. 구현된 오디오/데이타 프로세서 프로그램은 테스트 보드상의 프로그램 RAM으로 다운로드하여 DSP에서 마이크로프로세서 모드로 실행시켜 실시간 테스트를 하였다. 테스트를 위하여 사용된 장비는 HP 8920A로서 AMPS 기능을 에뮬레이션 해주는 RF 무선통신 장비이다. 실시간 테스트는 프로그램을 다운로드한 후 HP장비에서 call을 하여 데이타 프로세서 프로그램에서 데이타가 처리되고 오디오 프로세서 프로그램을 이용하여 voice channel상에서 음성통화가 잘 이루어 짐을 확인하였다. 또한 시험 보드가 40MIPS로 동작하도록 설계되어 있으므로 오디오/데이타 프로세서가 우리가 목표로하는 27MIPS의 계산량으로 실시간 동작이 이루어 지는가를 확인하기 위하여 프로그램에 필요없는 동작을 13MIPS만큼 추가하여 실시간 시험을 하여 정상동작 함을 확인하였다.

Ⅳ. 결 론

본 연구에서는 ES-C2340이라는 16 bit 고정소수점 DSP를 이용하여 오디오/데이타 프로세서를 실시간 구현하였으며 실장환경을 구성하여 AMPS기능을 실시간 시험하므로써 디지탈/아날로그 겸용 단말기에의 적용여부를 검토하였다. 오디오/데이타 프로세서는 IMM2라는 ASIC의 일부로 개발 되었기 때문에 IMM2 전체 시스템의 동작속도를 고려하여 27MIPS 이내에 동작하는 것을 목표로 구현하였다. 또한 오디오/데이타 프로세서 전체의 기능을 전부 프로그램으로 구현 하므로써 개발 기간을 크게 단축하였고, 실시간 시험을 통한 수정작업도 빠른 시간 내에 용이하게 할 수 있었다. 뿐만아니라 이 프로그램은 CDMA 모드에서 보코더로 사용되는 DSP를 함께 사용하게 되므로 IMM2를 설계하는데 있어 하드웨어의 사용 효율성을 높였다.

이렇게 테스트 보드 상에서 DSP를 이용하여 실시간 시험을 거친 오디오/데이타 프로세서는 IMM2 ASIC의 일부로 집적화되어 제작을 완료하고, 최종적으로 이동단말기를 이용하여 실장 시험을 한 결과 DSP상에서 실시간 시험한 결과와 똑같이 IMM2에서도 정상적으로 동작함을 확인하였다.

참 고 문 헌

1. EIA/TIA Standard, "Mobile Station-Land Station Compatibility Specification", 1989.
2. 한국전자통신연구소 반도체연구단, "ES-C2340 DSP2 Digital Signal Processor User's Guide", 1995.
3. 한국전자통신연구소 반도체연구단, "ES-C2340 DSP2 Digital Signal Processor Technical Manual", 1995.
4. Philips Semiconductors, "RF Communications Products Application note", pp. 646-659, May. 29, 1991.
5. SONY, "Semiconductor IC Data Book Mobile Communication", pp. 145-184, 1993.
6. 현진일, 한기천, 변경진, 김종재, 유하영, 김경수, 배명진, "보코더용 DSP의 설계", 대한전자공학회, DSP 설계 및 응용 워크숍 논문집, pp. 201-209, Mar. 25, 1995.
7. H. Y. Yoo, J. J. Kim, K. J. Byun, K. C. Han, D. K. Kim, J. S. Kim, H. B. Lee, M. J. Bae, "Efficient DSP Design for Vocoder Application", CICC '95, pp. 189-192, May 2, 1995.
8. Hewlett Packard, "HP 8920A User's Guide", September 1993.

▲변 경 진(Kyung Jin Byun) 1962년 2월 11일생

1987년 2월 : 국민대학교 전자공학과 졸업
1987년 3월~현재 : 한국전자통신연구원 통신회로연구실 선임연구원
※주관심분야 : DSP 설계, 음성신호 코딩

▲김 종 재(Jong Jae Kim) 1962년 4월 8일생
1988년 2월 : 광운대학교 재료공학과 졸업
1990년 2월~현재 : 한국전자통신연구원 VLSI 구조연구실 선임연구원
※주관심분야 : DSP 설계, 디지털 통신용 VLSI 설계

▲한 기 천(Ki Chun Han) 1964년 1월 12일생
1986년 2월 : 광운대학교 전자계산학과 졸업(학사)
1988년 2월 : 광운대학원 전자계산학과 졸업(석사)
1988년 2월~현재 : 한국전자통신연구원 통신회로연구실 선임연구원
※주관심분야 : DSP 설계, 디지털 통신용 시스템 프로그래밍

▲유 하 영(Hah Young Yoo) 1958년 10월 31일생
1982년 2월 : 서울대학교 전자공학과 졸업
1985년 2월 : Iowa State University 전기공학과(석사)
1988년 2월 : Iowa State University 전기공학과(박사)
1989년 2월~현재 : 한국전자통신연구원 VLSI 구조연구실 책임연구원
※주관심분야 : 영상통신용 VLSI 설계, DSP 설계, 음성신호 코딩

▲차 진 종(Jin Jong Cha) 1956년 9월 17일생
1980년 2월 : 한양대학교 전자공학과 졸업(공학사)
1982년 2월 : 한국과학원 전기 및 전자공학과 졸업(석사)
1982년 3월~1985년 4월 : 한국전자기술연구소 선임연구원
1985년 3월~현재 : 한국전자통신연구원 책임연구원 VLSI 구조연구실장
※주관심분야 : 디지털 통신용 VLSI 설계, 영상통신용 VLSI 설계, DSP 설계

▲김 경 수(Kyung Soo Kim) 1951년 12월 21일생
1977년 2월 : 서강대학교 전자공학과 졸업
1977년 2월~1985년 : 한국전자기술연구소
1986년~현재 : 한국전자통신연구원 책임연구원