심리음향모델과 SOLA 알고리즘을 이용한 코러스 칩 설계

The Design of Chorus DSP Chip Using Psychoacoustic Model and SOLA Algorithm

김 태 훈*, 박 주 성*
(Tae Hoon Kim*, Ju Sung Park*)

※ 이 논문은 반도체설계교육센터(IDEC)의 부분적인 지원에 의해 수행된 연구임

요 약

본 논문에서는 가요 반주기의 육성 코러스 기능을 구현하는데 핵심적인 기능을 하는 반도체 칩 설계에 관한 내용을 다룬다. 육성 데이터는 많은 저장 용량을 필요로 하고 있으므로 압축이 필요하고, 반주기의 키 및 템포 변화에 따라 육성 데이터의 키와 템포를 변화시키는 것이 필요하다. 본 연구에서는 압축을 위해서는 MPEG-1 오디오 계층1, 키 및 템포 변환을 위해서는 SOLA(Synchronized Overlap and Add) 알고리즘을 적절하게 변형하였다. 변형된 알고리즘을 구현할 수 있는 ASIC(Application Specific Integrated Circuit)을 설계하고 FPGA로 검증한 후 칩으로 제작하였다. 제작된 칩은 실제 시스템에 응용되어 정상적으로 동작하는 것을 확인하였다.
핵심용어: 오디오압축, 음성속도, 압축, 주문형반도체, 코러스, 심리음향모델

ABSTRACT

This research deals with the implementation procedures of a chorus processing DSP for karaoke system. It is necessary to compress the chorus data to store as many choruses as we can. We apply MPEG-1 audio algorithm to compress the chorus data. And the chorus system must be accompanied with the karaoke that can change the key and the tempo. So the chorus DSP must be able to change the key and tempo of the chorus data. We apply SOLA (Synchronized Overlap and Add) to do it. We designed the chorus DSP that can compress the chorus, change the key and tempo. And we verified the chorus DSP logic using FPGA. The used FPGA are two FLEX10K100s made by ALTERA. Finally we make the ASIC chip of chorus DSP and verify its operation.
Key words: MPEG, Audio, SOLA, Tempo, Compression, ASIC, Chorus
투고분야: 음성처리(2.2), 전기음향(3.2)

I. 서 론

노래 부르는 사람의 흥을 돋우기 위해서 합창단의 육성 코러스를 넣어주는 가요반주기들이 많이 나오고 있다. 대부분의 반주기는 곡의 템포나 키(음정)를 바꾸면 코러스 기능을 작동시키지 않고 있다. 그 이유는 저장된 육성 데이터의 키와 템포를 변환시킬 수 없기 때문이다. 본 연구에서는 압축되어 저장된 육성 데이터를 복원하고 키와 템포를 변환시키는 기능을 하는 DSP(Digital Signal Processor) 설계에 관한 내용을 다룬다.

육성 코러스 데이터를 저장하는데 많은 메모리 용량이 필요하므로 압축율이 높은 알고리즘을 사용해야한다. 압축 알고리즘을 선택할 때는 압축율과 복원하는데 얼마나 많은 비용(하드웨어, 계산량)을 고려해야 할 것이다. ADPCM 방식은 계산량은 적으나 압축율이 낮고, MPEG-1 계층 3(MP3) 방식은 압축율(12:1)은 높으나 구현하는데 많은 비용이 든다.[1-4] 이러한 측면을 생각하여 MPEG-1 오디오 계층1 방식을 채택하였다. 이 방식은 사람의 청각 구조를 이용한 효과적인 압축 방법이라고 할 수 있다. 사람의 청각 구조는 주파수 영역에서 필터 뱅크와 같이 시간 영역의 신호를 주파수 영역으로 바꾸어서 인지하며 이 때 각 주파수 대역에 따라서 민감도 또는 가청 한계가 다르다. 그리고 어떤 주파수 대역에 큰 에너지를 갖는 신호가 있을 때 주변 대역의 약한 신호를 듣지 못하는 마스킹 현상(Masking effect) 등이 생긴다. 마스킹 현상에 의해서 마스킹되어 인지할 수 없을 만큼의 양자화 잡음을 발생시키는 양자화 레벨을 결정한 후 그에 따라 비트 할당을 하여 데이터를 압축할 수 있다.

가요 반주기에서 반주의 키나 템포를 변할 시킬 경우 그에 따라서 코러스의 키나 템포도 같이 변하여야 한다. 이러한 기능을 수행하기 위하여 SOLA 알고리즘을 이용

* 부산대학교 전자공학과
접수일자 : 1999년 10월 20일

하여 키와 템포 변환을 수행한다.[5-8] SOLA 알고리즘은 시간 영역에서 템포를 변환시키는 대표적인 방법이며 계산량이 많기 때문에 계산량의 일부를 미리 처리하고 결과를 MPEG 오디오 헤더에 추가시킨다. 이러한 방법을 이용하면 키 및 템포 변환을 용이하게 구현할 수 있다.

먼저 Ⅱ장에서 MPEG에 대한 설명을 하고 Ⅲ장에서 SOLA에 대한 설명, Ⅳ장에서 설계 및 검증, Ⅴ장에서 칩 제작 및 테스트를 다루게 되고, 마지막으로 Ⅵ장에서 결론을 맺게 된다.

II. MPEG 오디오

2.1. 심리음향모델[1-4]

마스킹 현상을 이해하는 간단한 예로 그림 1에서 신호 A에 의해 마스킹 되는 양을 표시한 삼각형 모양의 마스킹 곡선을 주파수 영역에서 표시하였다. 이 마스킹 곡선의 아래에 있는 신호 B는 신호 A에 의해 모두 마스킹이 되어 우리 귀에 들리지 않지만 신호 C는 모두 마스킹이 되지 않아서 우리가 들을 수 있다. 위와 같은 경우에 우리가 들을 수 있는 소리는 신호 A와 신호 C이다. 이러한 마스킹 되는 양은 주파수마다 다르다. 저주파에서는 마스킹되는 양이 작고 고주파로 갈수록 많아진다. 그리고 주파수 각 대역마다 들을 수 있는 최소음의 크기인 절대 가청 임계치가 다르다. 이러한 성질을 이용하여 비트 할당시 발생하는 양자화 잡음이 마스킹 될 수 있다. 이러한 마스킹의 성질을 이용하여 압축에 응용할 수 있으며 특히 본 연구와 같이 코러스를 주로 하는 경우에는 마스킹값에 오프셋(offset)을 주어서 압축률을 조정하는 데 쓰일 수 있다.

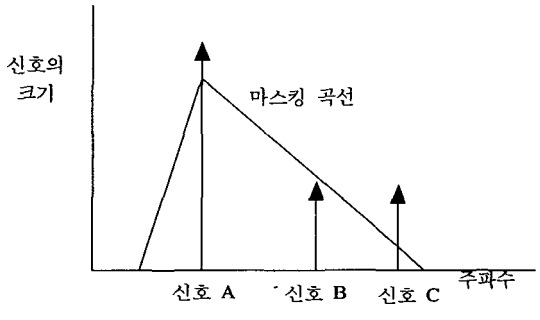


그림 1. 마스킹 곡선
Fig. 1. Masking curve.

2.2. MPEG 오디오 알고리즘 구현 방식 [11,13]

2.2.1. 부호화 과정

MPEG 오디오 계층 I, II의 부호화 과정을 그림 2에서 나타내고 있다.[3] 새로운 오디오 샘플이 입력으로 들어오면 주파수 영역에서 균일한 크기의 대역으로 분할된 32개의 서브밴드에서의 신호 크기를 구하기 위해서 서브밴드 분석 필터를 통과시켜 32개의 서브밴드 샘플을 구한다. 이와 같은 과정을 반복하여 각 서브밴드의 샘플이 12개씩 모이면 그 중 최대 값을 이용하여 scale factor를 구하며 그 값으로 나누면 서브밴드 샘플 값은 -1에서 1 사이로 정규화된다. 또한 오디오 샘플은 FFT를 이용하여 주파수 영역으로 변환된 후 심리음향모델을 이용하여 32개의 서브밴드에서의 마스킹 임계치를 구하게 된다. 여기서 구해진 마스킹 임계치로 각 서브밴드에서 필요한 비트수를 알 수 있다. 마스킹 값이 큰 서브밴드 샘플은 샘플 표현을 위한 비트를 적게 할당하여 양자화 잡음이 많이 발생하여도 그 양자화 잡음이 전부 마스킹될 수 있지만 마스킹 값이 작은 서브밴드 샘플은 많은 비트를 할당하여 양자화 잡음이 작아지도록 해야 한다.

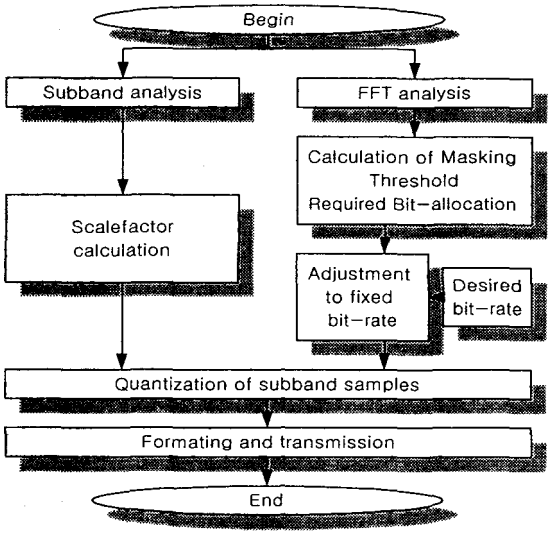


그림 2. 부호화 과정
Fig. 2. The encoding procedure.

심리음향모델에서 얻을 수 있는 것은 32개 서브밴드의 MNR (Masking to Noise Ratio)이다. 이것은 마스킹되는 양과 양자화 잡음의 비를 나타낸다. 즉 양자화 잡음이 얼마만큼 마스킹 되었는가를 나타낸다. 그러므로 MNR이 가장 작은 서브밴드부터 비트 할당을 시작한다. 전송률이 정해져 있는 경우에는 한 프레임에서 사용 가능한 총 비트수가 정해져 있으므로 할당 가능한 비트를 모두 사용할 때까지 비트를 할당한다. 비트 할당 정보를 이용하여 32개의 각 서브밴드 샘플을 마스킹값을 이용하여 구한 비트 수로써 표시하는 양자화 과정을 거친 후 일정한 포맷에 따라 한 프레임을 구성하게 된다. 이렇게 구해진 MPEG-1 계층 I의 한 프레임 포맷은 그림 3과 같다. Header에는 syncword, 계층, ID, 그리고 비트율 등의 정보가 있고, bit allocation에서는 32개의 각 서브밴드에 할당된 bit 수를 나타낸다. scale factor index는 각 서브밴드에서의 scale factor를 알 수 있는 index 값을 저장한다. 마지막으로 서브밴드 샘플이 들어 있다.

Header (32 bit)	Bit allocation (128 bit)	scale factor index (최대 192 bit)	Subband samples

그림 3. MPEG-1 계층 I의 한 프레임 포맷
Fig. 3. The format of a MPEG-1 layerI frame.

2.2.2. 복호화 과정

복호화 과정은 그림 4와 같은 과정을 따라 수행된다.[3] 헤더, 비트 할당 정보, scale factor index 정보를 읽은 다음 각 서브밴드의 비트 할당 정보를 이용하여 서브밴드 샘플을 읽어낸다. 서브밴드 샘플은 재 양자화를 거친 후 scale factor index를 이용하여 찾은 scale factor와 곱해진 후 서브밴드 합성 필터를 통과하여 32개의 새로운 PCM 샘플이 만들어진다. 새로운 32개의 서브밴드 샘플을 입력으로 받으면 matrixing 과정을 거친 후 윈도우 계수와 곱해진다. 이러한 과정을 거치면 복호화된 32개의 샘플을 얻을 수 있다.

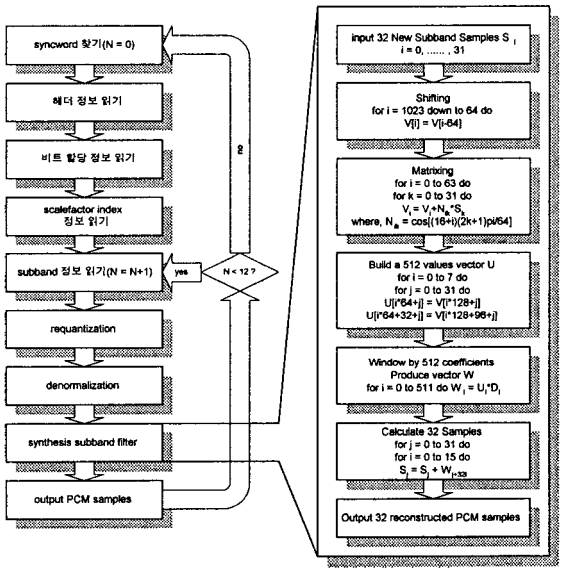


그림 4. 복호화 과정
Fig. 4. Decoding procedure.

2.3. 복호화 과정의 변형[9]

2.3.1. Syncword 제거

가요 반주기 시스템 내부에 있는 DRAM으로부터 데이터를 읽어올 경우 오류의 가능성은 거의 없으므로 별도로 syncword를 넣어서 각 프레임 별로 syncword를 찾아서 맞출 필요가 없다. 따라서 syncword와 그 외의 압축시 알고있는 정보를 담고 있는 헤더는 필요가 없다. 그러나 코러스의 끝이 어디까지인지는 syncword와 비슷하게 각 프레임의 처음에 sync 비트를 넣어서 복호화를 계속할지 여부를 결정한다.

2.3.2. 서브밴드 개수 가변

데이터의 종류에 따라서 신호가 주파수 모든 대역에 걸쳐서 나타나는 경우도 있고 저주파에서만 나타나는 경우도 있다. 따라서 기존의 경우 무조건 32개의 고정 서브밴드를 이용한다는 것은 비효율적인 면이 있다. 따라서 신호의 성질에 따라서 서브밴드 수를 다르게 하면 보다 효율적인 압축이 가능하다. 여기 구현된 복호화기는 프레임 헤더에 정보를 두어 32 서브밴드를 이용하는 경우, 16 서브밴드를 이용하는 경우, 서브밴드 데이터가 없는 경우로 나누어진다. 이것을 프레임마다 서브밴드 관련 정보로 넣어둔다. 그러면 압축하는 신호가 고주파에 신호가 없는 경우에는 16 서브밴드만 이용하고 묵음(silence)인 경우에는 모든 서브밴드 값이 0이 되도록 정보를 넣어 둔다. 그러면 비트 할당 정보에 드는 비트를 줄일 수 있고 더욱 압축율을 높일 수 있다.

2.3.3. MSB 반전 과정 제거

MSB 반전 과정은 전송시에 한 프레임에서 syncword (1111 1111 1111)와 같은 값이 프레임의 중간에 계속해서 나오지 않도록 하기 위하여 행하는 과정이라고 볼 수 있으므로 DRAM에서 부호화된 코러스 데이터를 읽어 오는 경우 저장된 어드레스만 알고 있으면 syncword가 필요 없다. 따라서 복호화시에 MSB 반전 과정은 생략될 수 있다.

2.3.4. 가변 비트율 적용

압축율을 결정하는 비트 할당은 심리음향모델에서 구해지는 마스킹값을 이용하여 양자화 잡음이 모두 마스킹 될 때까지만 할당하면 된다. 일반적으로 신호에 따라서 마스킹의 양이 변하게 되고 고정 비트율로 압축하면 마스킹이 많이 되는 구간에서는 양자화 잡음을 모두 마스킹하고도 비트가 남는 경우가 생길 수 있고 그 반대의 경우도 생길 수 있다. 따라서 마스킹의 양에 따라 비트율을 바꾸는 가변 비트율을 이용하여 압축율을 높일 수 있다.

III. SOLA 구현 및 알고리즘 연결

3.1. SOLA 알고리즘을 이용한 Time Scale방법[5-8]

3.1.1. SOLA 부호화 알고리즘 구현

SOLA 부호화 알고리즘은 다음의 그림 5와 같은 과정으로 수행된다. 여기서 시간축 인자(time scale factor)별로 구한 동기화 길이(synchronization lag)값 k는 SOLA 복호화시에 사용된다. k값을 구하기 위해 먼저 원 신호에 윈도우를 곱해 일정한 크기를 가지는 프레임으로 잘라낸다.

윈도우를 곱할 때 오버랩 시키면서 일정한 간격으로 이동시킨다. 이렇게 한 다음 연속된 두 개의 프레임간의 교차 상관(cross-correlation)을 구한다. 시간축 인자별로 구분하여 교차 상관이 최대가 되는 지점을 찾고 동기화 길이 값인 k를 구한다.

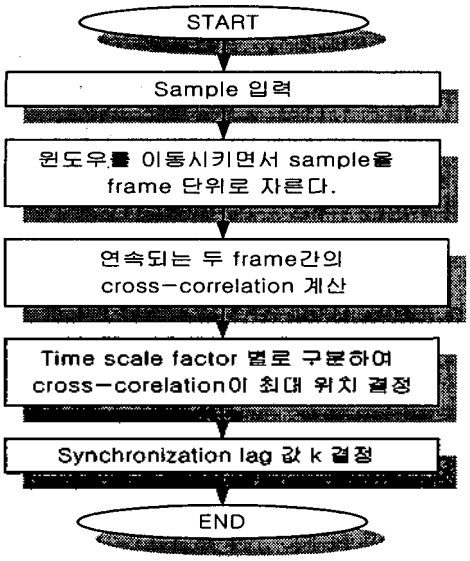


그림 5. SOLA 부호화 알고리즘
Fig. 5. SOLA encoding algorithm.

3.1.2. SOLA 복호화 알고리즘 구현

SOLA 복호화 알고리즘의 전체적인 흐름은 그림 6에 나타내었다. 윈도우를 씌워 구한 프레임들을 원하는 시간축 변환 비율인 시간축 인자 및 부호화시에 구한 동기화 길이 k를 고려하여 재배치시킨다. 이렇게 프레임들을 원하는 간격으로 재배치한 후 가중치를 주어서 더하면 원래와는 다른 길이의 신호 즉, 템포가 변환된 신호를 얻을 수 있다.

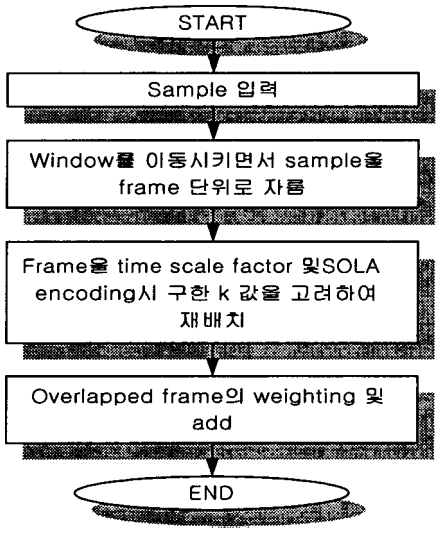


그림 6. SOLA 복호화 알고리즘 흐름도
Fig. 6. SOLA decoding algorithm flow.

3.1.3. SOLA 알고리즘을 이용한 음높이 가변 원리[10]

SOLA 알고리즘을 이용한 음높이 가변(pitch scale modification)과정은 그림 7과 같다. 먼저 SOLA 알고리즘을 이용하여 음높이 비를 고려하여 시간축 가변을 한다. 그림 7의 위쪽 그림은 600개의 원 샘플을 나타내고 있다. 가운데 그림은 시간 축 가변을 하여 800개의 샘플로 늘린 그림이다. 그런 후 일정한 샘플 간격으로 샘플을 제거하여 원 샘플 수와 같게 만들면 음높이 가변이 된 샘플이 된다. 그림 7은 음높이를 높이는 경우를 설명하고 있다. 반대로 음높이를 낮추는 경우는 먼저 원 샘플을 음높이 인자를 고려하여 일정한 간격으로 샘플)을 제거한 후 SOLA 알고리즘을 이용하여 원래의 샘플 길이만큼 시간축 가변을 시킨다. 이렇게 하면 원하는 음높이 가변이 된 샘플이 된다.

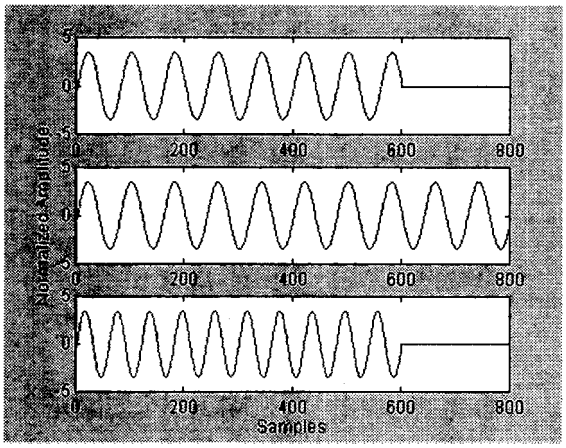


그림 7. Pitch Scale 과정
Fig. 7. Pitch scale procedure.

3.2. MPEG 오디오와 SOLA 알고리즘의 연결

PCM 코러스 데이터와 함께 원하는 압축율과 서브밴드 수가 엔코더 소프트웨어에 입력된다. 서브밴드 수는 입력이 어떤 종류냐에 따라서 두 가지 값을 선택할 수 있다. 음성이나 코러스 데이터인 경우 MPEG 오디오의 전체 32개의 서브밴드 중에서 16개만 이용하면 충분하고 그 외에 악기음이 추가된 경우에는 32개를 다 쓸 수 있도록 지정할 수 있다. 이렇게 압축률과 서브밴드 수가 지정되면 입력을 이용하여 SOLA 알고리즘을 수행한다. 여기서 얻어지는 것은 각 프레임의 k 값을 얻게 된다. 가요반주기에 필요한 15가지 시간 축 변화률에 대한 각각의 k 값을 다 구한 다음 MPEG 오디오 엔코더를 이용하여 입력 압축시에 k 값에 대한 정보를 헤더에 추가하게 된다. 이렇게 SOLA 알고리즘을 이용하여 미리 k 값을 찾아 놓는 이유는 실제로 교차 상관을 이용하여 k 값을 찾는 계산량이 무척 많기 때문에 이 계산량을 오프라인으로 처리하기 위함이다. 이와 같은 방법이 가능한 이유는 가요반주기의 코러스 데이터는 미리 정해져 있기 때문이다. 전체적인 흐름을 그림 8에 나타내었다.

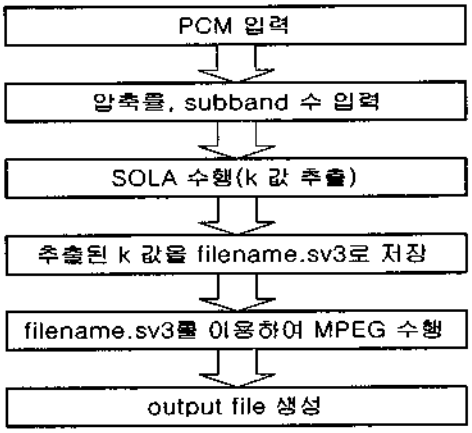


그림 8. 소프트웨어 흐름도
Fig. 8. Software flow.

IV. 설계 및 검증

4.1. 전체 블록 설계 및 동작

전체 코러스 DSP의 블록도를 그림 9에 나타내고 있다. 마이크로컨트롤러로부터 코러스 데이터를 입력받아 DRAM에 저장시킨다. 그런 다음 MPEG 오디오 복호화기가 코러스 데이터를 읽어와서 복호화를 수행한 후 다시 DRAM에 저장시키면 tempo control unit(SOLA) 블록에서 그 데이터를 읽어와서 타임 스케일링시켜 key control unit(interpolation/decimation)에 보낸다. 그러면 샘플링 주파수에 맞춰서 PCM 샘플이 DAC로 전달된다. 설계는 AVANTE사의 COMPASS tool(v8)을 이용하여 schematic 방식으로 진행하였다.

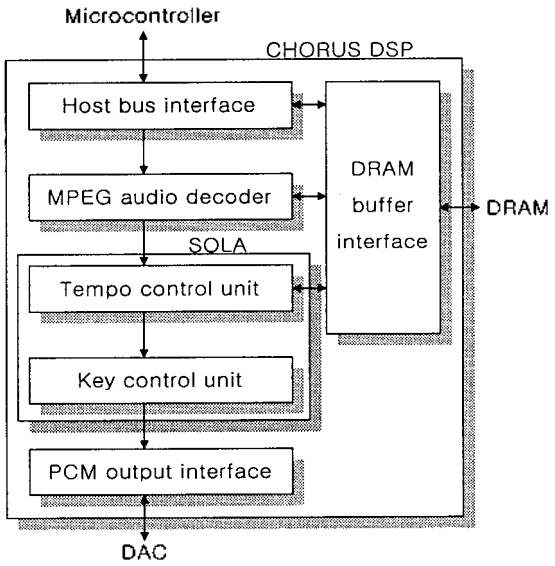


그림 9. 전체 블록도
Fig. 9. Top block diagram.

가라오케 시스템에 있는 마이크로컨트롤러가 코러스 DSP를 동작시키는 과정을 그림 10에 나타내고 있다. 먼저 전원을 킨 후 키, 템포 등 레지스터 값을 초기화시킨다. 그런 후 코러스 데이터를 다운로드 시킨다. 반주기에 맞춰서 코러스가 나와야될 시점에 play 레지스터를 set시키면 코러스가 나오게 되며 DRAM에 데이터가 다 차 있는 지 확인 하면서 계속해서 데이터를 다운로드 시킨다. 그리고 이것이 끝나면 다시 다른 코러스 데이터를 다운로드 시킬 준비를 한다. 데이터가 잘못되었거나 play가 끝나면 err_int가 1이 된다.

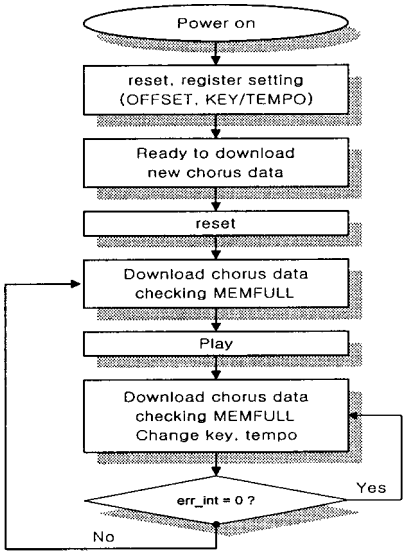


그림 10. 코러스 DSP 제어 과정
Fig. 10. Chorus DSP control flow.

4.2. MPEG 오디오 블록 설계 [9,12]

MPEG 오디오 복호화기의 전체 시스템 구성은 그림 11과 같이 인터페이스에 관련된 블록, date 읽기 블록, MAC(matrix) 블록, MAC(window) 블록의 4 가지 블록으로 크게 나누어진다. 입력은 CDROM으로부터 압축 데이터이며 이를 DRAM에 저장하며 출력은 PCM 샘플로써 이 역시 DRAM에 저장된다. 그러면, 이 샘플을 템포 변환 블록에서 가져간다. 빠른 처리 속도를 위해 복호화 과정을 3 단계의 파이프라인으로 구성하였다. 각 단계에서 수행해야할 일들이 균등하게 분배되게 하였다. 3 단계는 Data 읽기, MAC(matrix), MAC(window) 블록으로 이루어졌다. Data 읽기는 압축된 데이터를 DRAM으로부터 필요한 값을 읽어 내는 과정이고 MAC(matrix)은 필터 계수를 곱한 후 더하는 단계이며 MAC(window)는 윈도우 계수를 곱한 후 더하는 단계이다. 이 3 단계를 차례로 거친 후 원하는 PCM 샘플을 얻을 수 있다. 각 단계에서 실행이 끝난 후 각 단에서 다음 단으로 데이터를 넘겨주는 인터페이스 단계가 있다.

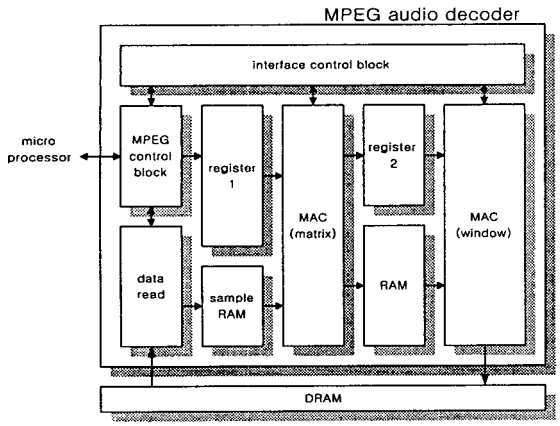


그림 11. MPEG 오디오 블록도
Fig. 11. MPEG audio block diagram.

4.3. SOLA 블록 설계

SOLA 블록은 그림 12와 같이 크게 9개의 실행 사이클을 가진다. MPEG 오디오 블록에서 압축된 데이터를 복원하여 DRAM에 저장하면 이 데이터와 윈도우 값을 가중치를 곱하고 더하는 과정을 반복하게 된다. SOLA 알고리즘에서 윈도우로 잘려진 4개의 프레임이 중첩되어야 하는 과정이 바로 여기서 수행된다. 여기서 add_condition은 각 윈도우와 곱해진 프레임이 새로 더해질지 여부를 나타내고, Acc(Accumulator)는 계산된 값을 계속 더해나가기 위한 임시 저장 장소로 이용된다.

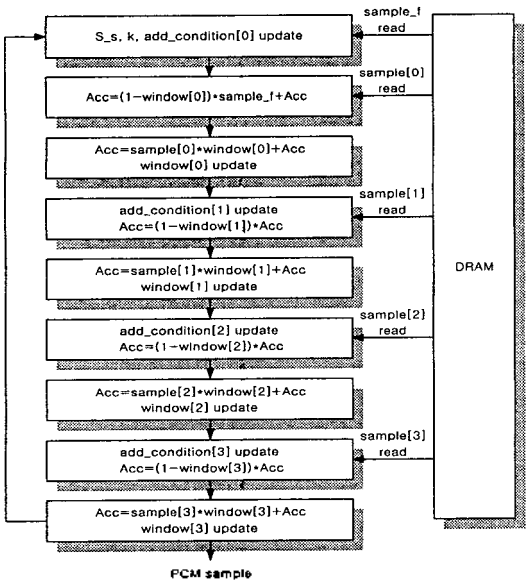


그림 12. SOLA Block의 사이클별 실행
Fig. 12. Cycle execution of SOLA block.

설계된 SOLA 블록을 그림 13에서 나타내고 있는데 실제로 교차 상관을 구하는데 많은 계산량이 필요하다. 따라서 상관관계로 소프트웨어로 미리 구한 다음 그 결과를 MPEG 헤더에 같이 넣게 되어 있다. 따라서 이러한 파라메터를 MPEG 블록으로부터 받아서 복호화 후에 DRAM에 저장된 데이터를 읽어와서 템포를 변환시킨다. Pre-encoding 과정에서 얻은 k와 s_s를 이용하여 템포가 변환된 출력 샘플을 만들어낸다. S_s와 k 값을 이용하여 각 프레임이 붙어야할 위치를 계산하고 윈도우 값을 변경시키면서 PCM 샘플을 DRAM으로부터 읽어와서 윈도우와 곱한 뒤 출력 샘플을 만든다.

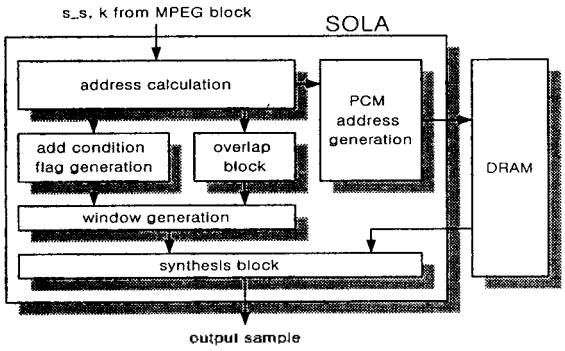


그림 13. SOLA 블록도
Fig. 13. SOLA block diagram.

4.4. 검증

구체적인 로직 설계 이전에 MPEG 오디오 디코딩 알고리즘, SOLA 알고리즘, 변형된 MPEG 포맷에 대하여 C 언어를 이용하여 고정소수점 시뮬레이션 하여 소리를 직접 들어봄으로써 전체적인 개념을 확인하였다. 이 과정에서 얻어진 중간 결과를 논리 시뮬레이션시의 참고자료로 활용하였다. 설계된 논리회로에 대하여 변형된 포맷으로 만들어진 실제 음성 데이터를 입력하여 음정변환 등이 되는지 확인하였다. 시뮬레이션 시간이 많이 소요되어 6 프레임의 샘플 데이터에 대해서만 시뮬레이션 하였다.

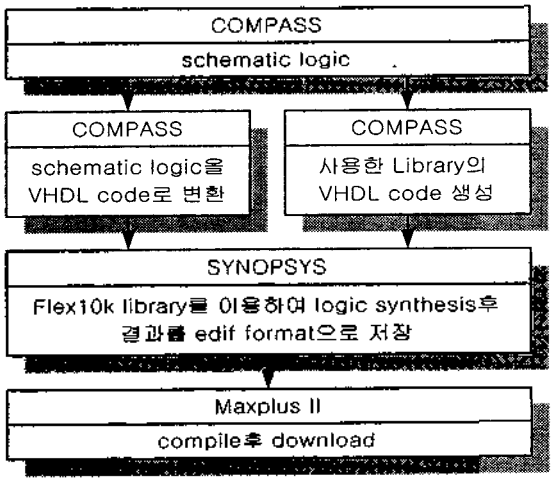


그림 14. 로직 변환 과정
Fig. 14. Logic translation flow.

논리 시뮬레이션을 통하여 5초간 재생되는 음성 데이터를 만들어 내기 위해서는 Sparc20에서 한달 정도가 소요되고 실제 시스템이 코러스 DSP를 제어하는 다양한 방식을 조사하는 테스트 벡터를 만들기 어렵기 때문에 FPGA를 이용하여 코러스 DSP를 만들어 검증하였다. 전체 로직은 COMPASS에서 schematic을 이용하여 설계되었다. 따라서 ALTERA FPGA를 이용하기 위해서 변환 과정이 필요하다. 그림 14에서 이러한 과정을 나타내고 있다.

FPGA 실험 보드 전체 시스템 구성 사진을 그림 15에서 나타내었으며 FPGA는 ALTERA사의 FLEX10K100GC-503-3 2개를 사용하여 구현하였다. 그림에는 2개의 FPGA 실험 보드를 나타내고 있다. FPGA를 이용하여 44.1KHz의 샘플링 주파수를 가지는 데이터를 실시간으로 복호화하여 키 및 템포 변환을 실제 소리로 청취할 수 있었다. C 언어를 이용하여 고정소수점 시뮬레이션한 결과와 FPGA 출력을 PC로 받은 값과의 SNR은 70dB이상으로 거의 같다고 볼 수 있다.

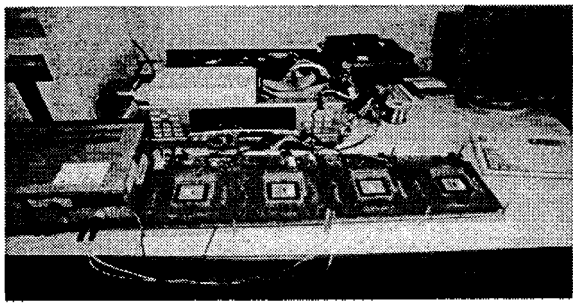


그림 15. FPGA를 이용한 전체 시스템 구성
Fig. 15. System configuration using FPGA.

V. 칩 제작 및 테스트

5.1. 칩 제작

Schematic으로 설계된 로직을 P&R(Placement and Routing) 수행 후 post-simulation을 수행한다. 여러 가지 delay factor에 대한 시뮬레이션 결과를 확인한 후 DRC check와 Net compare과정을 거치게 된다. 그리고 칩 동작 테스트 용이와 toggle rate를 높이기 위해서 2개의 test pin이 있다. 이 두 개의 pin으로 정상 동작, MPEG 블록, SOLA 블록, 카운터 증가의 네 가지 경우 선택하여 각 블록을 쉽게 테스트 할 수 있다.

여기서 카운터 증가 모드는 설계 블록내의 여러 가지 카운터 중에서 시뮬레이션이나 테스트 벡터를 이용해도 쉽게 증가하지 않는 카운터를 임의로 증가시켜서 그 값의 올바른 증가와 그에 따른 주변 로직의 동작을 쉽게 확인할 수 있도록 해준다. 이러한 과정을 거쳐서 제작된 코러스 DSP의 주요 특징을 요약하면 표 1과 같다.

표 1. ASIC 설계 사양
Table 1. ASIC design specification.

항 목	특 징
타임 스케일 변화율	0.6 ~ 1.6
음정 변화율	-1/2 octave ~ 1/2 octave
사용할 base array 또는 library 종류	vgc400278
Technology 종류	0.8 ㎛
Package type (production)	100-PQFP (100-PQFP)
동작 주파수	16.9344MHz
Gate 수	56000
Chip 크기	0.9735 X 0.9326 cm2

5.2. 테스트 및 응용

제작된 칩은 5V에 동작하며, 동작 주파수는 16.9344MHz, 소모전력은 420mW이다. 그리고 C 언어로 시뮬레이션한 결과와 칩에서 나온 결과가 칩화를 위하여 정수형 처리 과정에서 발생하는 오차처리 범위(70dB)이내에 들어가는 것을 확인하였다. 그림 16은 C 언어를 이용하여 시뮬레이션한 결과이며 그림 17은 설계된 칩의 출력 파형이다.

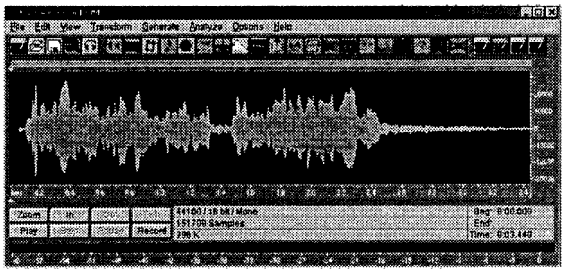


그림 16. C 언어로 시뮬레이션한 결과
Fig. 16. Simulation result using C language.

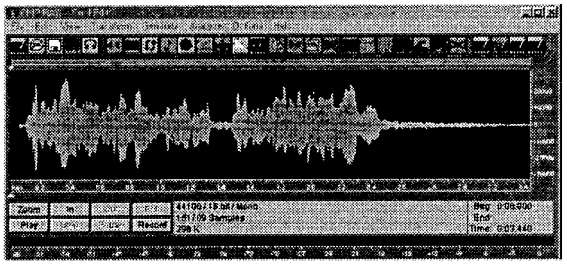


그림 17. 칩의 출력 파형
Fig. 17. Output waveform of chip.

그림 18과 그림 19에서 칩을 이용하여 타임 스케일한 결과를 디지털 오실로스코프로 잡은 파형을 나타내고 있다. 그림 18은 타임 스케일링 전의 코러스 데이터로 15초의 샘플 길이를 갖고 있다. 그림 19는 그림 20의 샘플 파형을 제작된 칩에 입력하여 -12% 타임 스케일하여 전체 샘플

길이를 13.3초로 변형시킨 결과를 나타내고 있다. 전체적인 파형의 윤곽은 그대로 유지하고 있으면서 시간축 변환이 일어난 것을 알 수 있다. 주관적인 평가에 의해서도 음질 손상 없이 타임 스케일링 된 것을 알 수 있었다. 응용보드 사진을 그림 20에 나타내고 있는데 우측 상단의 daughter 보드에 장착된 칩이 코러스 DSP이다.

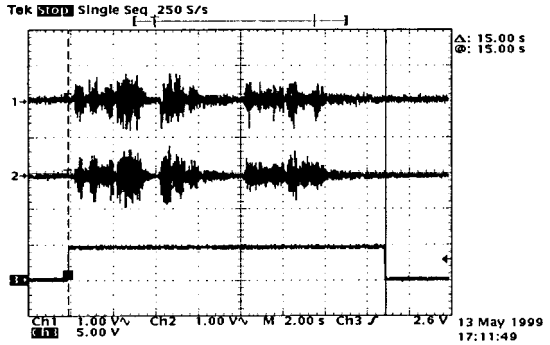


그림 18. 타임 스케일 전의 코러스 파형
Fig. 18. Chorus waveform before time-scaling.

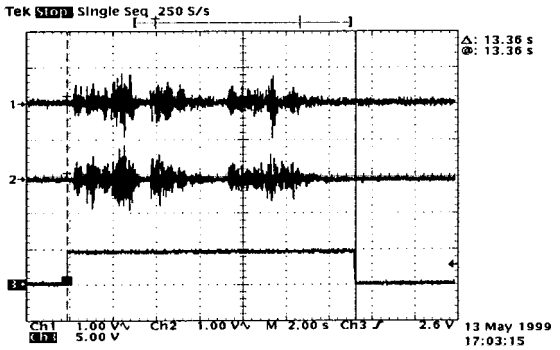


그림 19. -12% 타임 스케일 후 코러스 파형
Fig. 19 Time-scaled chorus waveform.

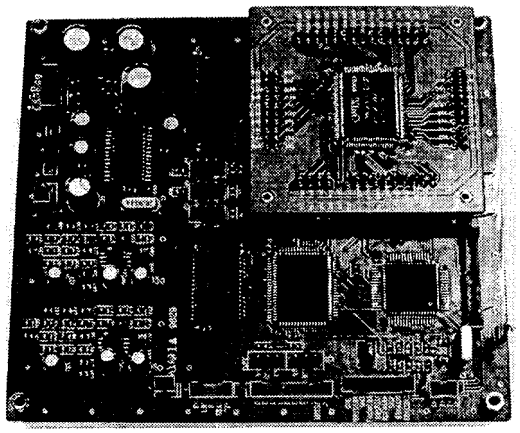


그림 20. 코러스 DSP를 적용한 오디오 보드
Fig. 20. Audio board using chorus DSP.

VI. 결 론

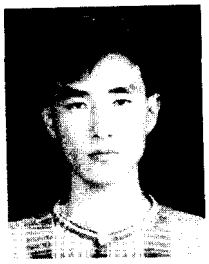
MIDI음악과 동기가 되어 있는 음성 데이터 즉 코러스 데이터를 처리하는 시스템에 있어서 음성 데이터의 압축이나 키, 템포 변환을 실시간으로 처리해 내는 기술은 매우 중요하다. 그러나 현재 발표된 MPEG 오디오나 MP3(MPEG 오디오 계층3)등의 규격에서는 데이터압축에 관한 부분은 고려되어 있지만 본 연구에서와 같이 데이터 압축과 키, 템포의 가변을 동시에 처리해 낼 수 있는 DSP 칩이나 연구내용은 찾아보기 힘들었다. 따라서 본 논문에서는 이러한 문제를 해결하기 위해 MPEG 오디오 계층1과 키, 템포 가변이 가능한 FPGA를 구현하고 이를 검증하였으며 반도체 칩으로 제작하였다. 또한 향후에는 보다 압축률이 높은 MP3와 키, 템포 가변이 가능한 알고리즘이나 단일 칩의 개발도 필요하다고 판단되며 경우에 따라서는 키, 템포 가변 개념이 고려된 새로운 압축 포맷도 연구할 필요가 있다고 판단된다. 개발 단계에서는 현재 템포 변환에 널리 사용하고 있는 시간 영역에서의 변환 방법인 SOLA 알고리즘을 분석하고 이를 C 언어를 이용하여 검증하였으며 C 프로그램을 바탕으로 로직 구현 및 FPGA 검증을 완료하였다. 여기에 사용된 FPGA로는 ALTERA 사의 FLEX10K100 두 개를 사용하였으며 최종 설계된 논리 게이트 수는 5만 6천 개 정도이고 동작 주파수는 16.9344MHz로 설계되었다. COMPASS tool을 이용하여 0.8㎛ gate-array library를 이용하여 제작되었으며 시스템에 적용시켜서 정상적으로 복원, 키, 템포 변환을 수행함을 확인할 수 있었다.

참 고 문 헌

1. Y. F. Dehery, et al. "A MUSICAM source codec for digital audio broadcasting and storage," Proc. ICASSP pp. 3605-3608, 1991.
2. Davis Pan, "A Tutorial on MPEG/Audio Compress," IEEE Multimedia, pp. 60-74, Aug. 1995.
3. Karlheinz brandenburg and Gerhard stoll, "ISO-MPEG-1 Audio: A generic Standard for Coding of High-Quality Digital Audio," J. Audio Eng. Soc. Vol. 42, No. 10, pp. 780-792, Oct. 1994.
4. Yves-francois dehery, "Musicam source coding," AES 10th international conference, pp. 71-79.
5. 이성주, "천이구간정보를 이용한 음성의 가변적인 시간축 변환," 부산대 전자공학과 석사논문, 1997.
6. Salim Roucos and Alexander M. Wilgus "High Quality Time-Scale Modification for Speech," IEEE, Aug. 1985.
7. M. R. Portnoff, "Time-scale modification of speech based on short-time Fourier analysis." IEEE Trans. Acoustic., Speech, Signal Processing, vol. ASSP-29, no.3, pp. 374-390, Jun. 1981.
8. Daniel W. Griffin and Jae S. Lim, senior member, IEEE, "Signal Estimation from Modified Short-Time Fourier Transform," IEEE Transaction on Acoustics, Speech, and Signal Processing, vol. ASSP-32, no. 2, April, 1984.

9. 김태훈, "MPEG Audio 복호기에 관한 연구," 부산대 전자공학과 석사논문, 1997.
10. Curtis Roads, "The Computer Music Tutorial".
11. 장호근, 김태훈, 곽종태, 박주성, "오디오 압축 방식을 적용한 사운드합성 시스템의 설계," 한국음향학회, 제17권 제3호,pp.27-36, Mar. 1998.
12. 한성철, 유선국, 정남훈, 박성옥, 한영태, 윤대희, "FPGA를 이용한 MPEG-2 오디오 복호화기의 구현," 한국음향학회 전기음향 워크샵, pp. 107-111, Oct. Apr. 1995.
13. 홍진우, 김성한, 장대영, "멀티채널 오디오 부호화 방식의 분석," 한국음향학회 학술발표대회논문집, 제16 권 1(s)호, pp. 177-182, Jul. 1997.

▲김 태 훈(Tae hoon Kim) 1972년 4월 25일생



1995년 2월 : 부산대학교 전자공학과 (공학사)
1997년 2월 : 부산대학교 전자공학과 (공학석사)
1997년 2월 ~ 현재 : 부산대학교 전자공학과 대학원 박사과정, 컴퓨터 및 정보통신연구소 연구원
※ 주관심분야 : 사운드 압축 및 복원, DSP 설계

▲박 주 성(Ju sung Park)
부산대학교 전자공학과 부교수
한국음향학회지 제 17 권 3 호 참조