Embedded ARM Processor를 이용한 MPEG-4 Simple Profile Decoder의 구현

論 文
52P-2-7

The Implementation of MPEG-4 Simple Profile Decoder using the Embedded ARM Processor

朴 成 煜*
(Sung-Wook Park)

Abstract - This paper has presented the efficient implementation of MPEG-4 simple profile video decoder, which is used as video compression standard in mobile video communication. We have used the ARM9 processor in implementing this MPEG-4 simple profile, which requires much processing power and low power implementation. At first we implemented with C-language under the PC environment with ADS(ARM Developer Suite) environment, and then we have tried to reduce a clock cycle for a power consumption optimization through conversion an assembly language for C-code partly. We have verified the processor is operated at 22.47MHz operation after optimization, but 148MHz before optimization.

Key Words : MPEG-4 Simple Profile, MPEG-4 Decoder, Embedded System

1. 서 론

최근 영상, 음성, 텍스트 등의 다양한 데이터를 종합적으로 취급하는 멀티미디어에 대한 연구 개발이 활발하게 이루어지고 있다. 그 중 영상 데이터는 시각적 효과라는 측면에서 중요한 정보원의 하나이다. 그러나 정보량이 방대하여 효율적인 저장 및 전송을 위해서는 압축부호화가 필요하다. 디지털 동영상 및 음향의 부호화와 다중화를 위한 국제표준을 정하고 있는 MPEG에서는 MPEG-1/2[1-2]에 이어서 차세대 멀티미디어의 국제표준으로 MPEG-4의 표준화를 추진하여 MPEG-4 version-2[3]를 발표하였다.

MPEG-4는 인터넷응용, 이동 멀티미디어 통신, 멀티미디어 컨텐츠 제작 등의 다양한 형태의 차세대 멀티미디어 서비스를 지원할 수 있도록, 단순한 AV 전달이 아닌 대화형 멀티미디어 서비스의 제공을 핵심으로 하고 있다. 또한 부호화의 관점에서 보면 지금까지의 동영상 압축기술은 영상의 내용과는 무관하게 프레임 단위로 부호화하는 방식인데 반하여, 수신 측에서의 다양한 조작이 가능하고 임의 형상의 정보를 고려한 영상 객체(VO:visual object) 단위의 부호화가 가능하여 그 사용이 널리 확대되고 있다[4].

MPEG-4의 기본 알고리즘은 영상의 공간적인 중복성을 제거하는 과정인 DCT(Discreate Cosine Transform) 변환과 시간적인 중복성을 제거하는 움직임 예측 및 보상과정을 조합한 방식을 채택하고 있으며, H.263[5]을 기본으로 하고 MPEG-1과 MPEG-2의 우수한 점을 채택하여 높은 압축률을 갖는다.

* 正 會 員 : (주)로직메카 멀티미디어통신 연구소
接受日字 : 2003年 1月 16日
最終完了 : 2003年 4月 17日

한편, ARM 프로세서는 비교적 적은 전력을 소모하고, 고성능이며, 코어의 크기가 작아 임베디드 시스템에서 널리 사용되고 있다[6]. 이러한 장점으로 인하여 고 성능을 요구하는 영상의 부호화 및 복호화 알고리즘을 이동 통신 환경에서 구현하기 위한 ARM 코어의 사용이 점점 증가되고 있다. 또한 휴대형 이동기기에서는 연산능력과 전력공급이 제한되어 있기 때문에 제한된 자원에서 효율적인 시스템을 구축하기 위하여 알고리즘과 프로세서 아키텍처에 최적화된 구현 방식이 요구되고 있다. 이를 위해, 본 논문에서는 ARM920T 코어를 이용한 MPEG-4 simple profile 영상 복호화기를 구현하고, 이를 위한 복호화기의 최적화 기법에 관하여 기술한다.

본 논문은 다음과 같이 구성된다. 2장에서는 MPEG-4 Simple Profile 영상 복호화 과정에 대하여 기술하고, 3장에서는 ARM920T 코어의 구조와 개발환경에 대하여 설명한다. 4장에서 최적화 기법에 대하여 기술하며, 5장에서 실험 방법과 결과를 검토하고 마지막 6장에서 결론을 맺는다.

2. MPEG-4 영상 복호화 과정

이동 통신 환경의 영상 압축 표준안인 MPEG-4의 가장 큰 특징은 영상을 객체(VO) 단위로 부호화 및 복호화가 가능하다는 것이다. 이러한 영상 객체는 자연 영상 데이터와 컴퓨터 그래픽스 영상 등의 합성 영상으로 나뉘며, 자연 영상은 다시 고정적인 형태를 가진 구형 형상과 임의 형상으로 나뉜다. 이러한 자연 영상을 부호화하기 위한 방법은 구형 형상은 프레임 기반 부호화를 행하며 임의 형상은 일련의 의미 있는 처리 단위로 모은 도구의 조합인 객체 또는 내용을 기반으로 한 부호화를 수행한다. 그림 1은 MPEG-4의 기본 개념도이다.

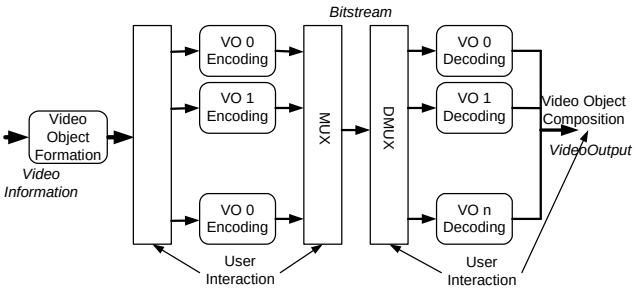

그림 1 MPEG-4의 블록도
Fig. 1 General block diagram of MPEG-4

그림 1의 각 객체로 나뉜 자연 영상의 부호화와 복호화를 위한 알고리즘은 그림 2와 같이 영상신호의 부호화 및 복호화를 나타내는 텍스처(texture) 부호화 및 복호화 부와 형상신호에 대한 정보를 알려주는 형상 부호화부 및 가변장 부호화 부로 구성되어 있다. 이 중 텍스처 부호화 부는 H.261, H.263, MPEG-1, MPEG-2등의 부호화 방식인 손실 부호화의 공간적 중복성을 이용한 압축 방식인 DCT 변환 및 양자화, 시간적 중복성을 이용한 압축 방식인 움직임 보상 및 예측을 조합한 방식을 기본으로 사용한다.

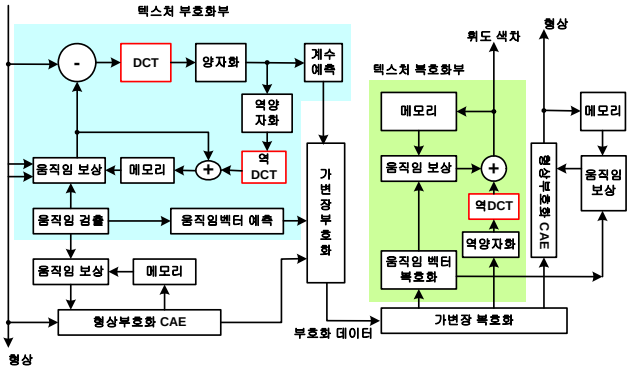

그림 2 객체의 부호화/복호화 알고리즘
Fig. 2 Basic algorithm for natural visual data

부호화기에서는 영상을 각 객체 단위로 분할한 후 부호화하여 전송하고, 복호화기에서 각 객체 단위로 복호화를 실시하여 각 객체를 결합한 후 영상을 복원한다. 여기서, 분할된 각 객체를 결합하는 단계에서 객체를 제거하거나 삽입하여 영상을 편집할 수 있다. 그림 3은 MPEG-4 영상의 객체 구조를 나타내었다.

그림 3과 같이 MPEG-4의 비트스트림 구조영상 입력으로써 영상 세션(VS)을 정의한다. VS은 영상 객체(VO)로 구성되고 VO는 영상 객체 층(VOL)으로 구성되며 각각의 VOL은 여러 개의 영상 객체 평면으로 이루어진 영상 객체 평면 그룹(GOV)으로 나누어지고 GOV은 각각 영상 객체 평면(VOP)으로 이루어져 있다.

MPEG-4 시스템의 영상부(VS)는 여러 개의 영상 객체로 이루어져 있고 각각의 영상 객체는 영상 객체 층(VOL:video object layer)을 포함한다.

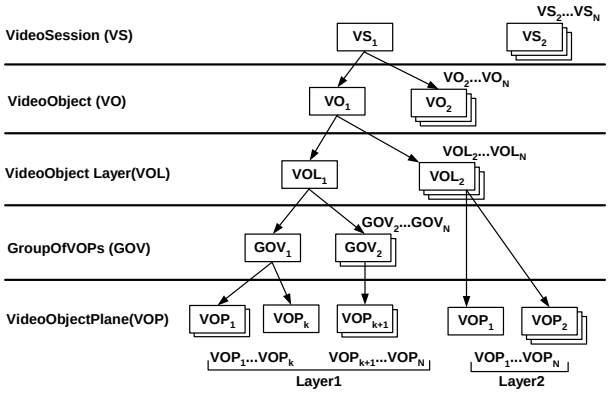

그림 3 MPEG-4 영상 객체의 구조
Fig. 3 An MPEG-4 video bitstream logical structure

영상 객체 층은 여러 개의 영상 객체 평면 집합(GOV:group of VOP)으로 구성되고 영상 객체 평면 집합은 각각의 영상 객체 평면(VOP:video object planes)를 포함한다. 그림 4는 MPEG-4 영상 복호화기의 블록도이다.

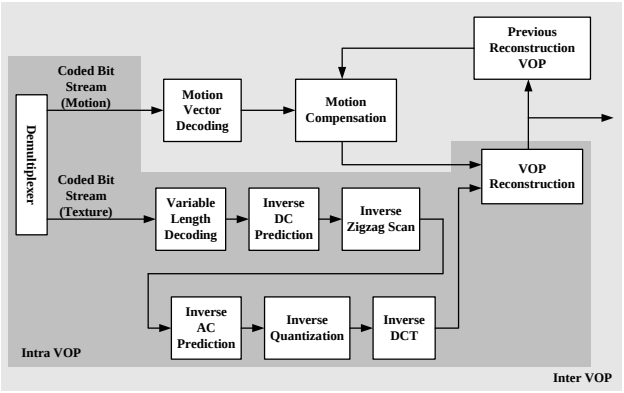

그림 4 MPEG-4 영상 복호화기의 블록도
Fig. 4 General block diagram of VOP based decoding in MPEG-4

MPEG-4 Simple Profile은 DCT 변환을 기반으로 부호화하는 인트라 VOP(I-VOP)와 움직임 예측과 보상을 기반으로 부호화하는 인터 VOP(P-VOP)의 두 가지 방식을 사용한다. 인트라 VOP는 부호화된 비트스트림을 분석하고 가변장 복호화(VLD:variable length decoder), DC 예측 복호화, 역 지그재그 스캔, AC 예측 복호화, 역 양자화, 역 DCT를 거쳐서 VOP를 복원하고, 인터 VOP는 비트스트림에서 움직임 벡터(MV:motion vector)를 추출하고 움직임 보상을 실시하여 VOP를 복원한다.

3. ARM920T 코어의 구조

3.1 코어의 성능 비교

임베디드 시스템 구현에 주로 사용되고 있는 ARM 코어와 DSP 코어, Hard-wired 방식은 적용되는 방식에 따라서

장단점이 존재한다. 표 1에 각 코어간의 성능을 비교하였다.

표 1 코어의 성능 비교
Table 1 Comparison of cores

core / classifier	ARM core	DSP Core	Hard-wired
Performance	Medium	High	High
Power	Medium	High	Low
Cost	Medium	High	Low
Time	Short	Short	Long

표 1에서와 같이 DSP 코어를 이용하는 방식은 높은 계산 능력과 짧은 개발 시간을 요구하나 소모비용이 높고 전력 소모가 많다는 단점이 있다. Hard-wired 방식은 높은 성능과 적은 전력 소모, 낮은 가격 면에서 다른 코어들보다 성능이 뛰어나지만 개발 기간이 오래 걸린다는 단점이 있다. ARM 코어를 사용하는 방식은 비교적 높은 성능과 적은 전력 소모, 낮은 가격과 빠른 개발시간을 갖는다. 본 논문에서는 Time-to-Market의 개념에서 우수성을 갖는 ARM 코어를 선택하여 구현하였다.

3.2 ARM 코어의 비교

ARM 코어 중 최근에 주로 사용되고 있는 코어는 ARM7TDMI와 ARM9TDMI이다. 표 2에 ARM7TDMI와 ARM9TDMI의 성능을 비교하였다.

표 2 ARM 코어의 성능 비교
Table 2 Comparison of ARM cores

core / classifier	ARM7TDMI	ARM9TDMI
Architecture	ARMv4T	ARMv4T
Pipeline	3 Stage	5 Stage
Power	87mW	150mW
Clock	66MHz	200MHz
MIPS	60	220
MIPS/W	690	1500

표 2에서와 같이 ARM7TDMI와 ARM9TDMI는 같은 ARMv4T 아키텍처를 사용하지만 파이프라인 개수가 각각 3개와 5개로 다르다. ARM9TDMI의 경우 명령어 파이프라인 개수가 증가함으로써 ARM7TDMI 보다 좀 더 효율적인 명령어 수행이 가능하다. 전력 소모량의 경우 ARM7TDMI가 87mW, ARM9TDMI 150mW로 ARM9TDMI가 더 많이 소모하나 동작 속도가 각각 66MHz, 200MHz로 ARM9TDMI의 동작 속도가 더 빠름을 알 수 있다. 그리고 전력의 효율성을 나타내는 파라메터인 MIPS/W를 비교해 보면 ARM7TDMI는 690MIPS/W이나 ARM9TDMI는 1500MIPS/W로 같은 전력에서 두 배 이상의 성능을 나타내는 것을 알 수 있다. 본 논문에서는 휴대형 이동 통신 환경에서 중요한 요건인, 효율적인 전력 사용을 위하여 ARM9TDMI를 선택하여 구현하였다.

3.3 ARM 코어의 비교

ARM9 코어의 종류에는 ARM920T와 ARM940T가 있다. ARM920T 코어는 ARM9TDMI 코어를 프로세서로 사용하고 16KB의 명령어/데이터 캐시와 Memory Management Unit(MMU)를 포함하는 구조이고 ARM940T 코어는 MMU 대신에 캐시 보호 블록(Cache Protection Unit)이 포함된 구조이다. 그림 5는 ARM920T 코어의 구조이다.

ARM920T 코어는 내부 명령어 캐시 및 데이터 캐시 사이의 명령어와 데이터 교환을 레지스터(R13)를 통하여 제어하고, 내장되어 있는 보조 프로세서 CP15는 캐쉬와 MMU를 제어 및 설정하는 기능으로 사용한다. 또한 코어 외부에 다른 코프로세서를 사용하여 동작시킬 수 있는 외부 인터페이스와 Multi-Trace를 이용한 실시간 디버깅이 가능하도록 Trace Interface 포트가 존재하며 효율적인 메모리 관리를 위하여 내부에 MMU를 적용하였다. 코어 내부의 각 블록의 연결은 ARM에서 제안한 AMBA(Advanced Microcontroller Bus Architecture) 버스를 사용하여 구성한다.

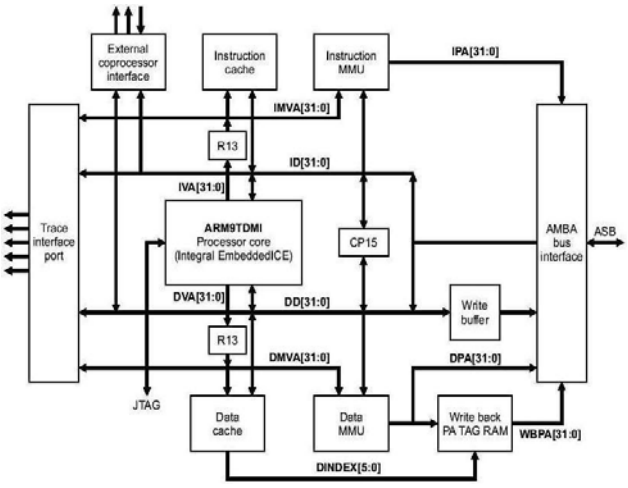

그림 5 ARM920T 코어의 블록도
Fig. 5 ARM920T core block diagram

3.4 ARM 코어를 이용한 개발 환경

ARM 코어를 이용하여 알고리즘을 구현하는 경우 구현하고자 하는 알고리즘의 PC상의 검증, ARM 코어의 특성을 고려한 구현, 디버깅, 성능 테스트 등의 과정을 거쳐야 한다. 그림 6은 ARM 코어를 이용한 알고리즘의 구현 절차를 나타내었다.

먼저 구현하고자 하는 알고리즘을 PC상에서 구현하여 알고리즘을 검증하고 ARM 코어를 이용한 구현 시 동작 검증을 위한 기준으로 사용한다. ARM 컴파일러 환경에서 알고리즘을 구현한 후 ARM 에뮬레이터를 사용하여 PC상에서 구현된 알고리즘과 비교하고 구현된 알고리즘의 정확성을 검증한다.

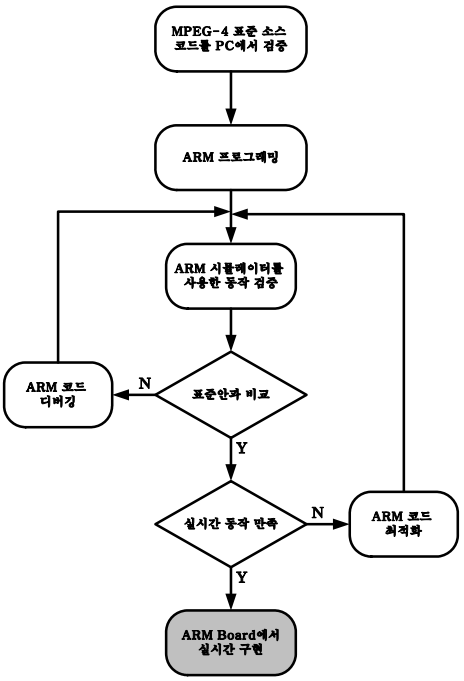

그림 6 ARM 코어를 이용한 구현 절차
Fig. 6 Implementation flow chart using ARM core

구현상의 오류가 발생하는 경우 디버거를 이용하여 오류를 정정하고, 오류가 없는 경우 성능 테스트를 실시한다. 만일, 성능 테스트 결과가 목표 성능에 미치지 못한다면 ARM 코드의 최적화를 실시하여 목표한 성능에 도달하도록 한다. 그리고 결과물을 ARM 보드에 적용하여 알고리즘의 실시간 구현을 완료한다.

4. 알고리즘 최적화 기법

최적화 기법에는 MPEG-4 구현 단계에서의 알고리즘 선택에 의한 방식과 ARM의 아키텍처의 특성을 이용한 방식이 있다.

알고리즘 레벨에서의 최적화 에서는 MPEG-4의 구성요소 중 역 DCT 알고리즘의 코사인 테이블 변형에 의한 방식과 Coded Pattern Code(CBP) 파라메터를 이용한 방식 두 가지가 있다. 역 DCT의 코사인 테이블 변형 방법은 역 DCT 연산에 사용되는 코사인 값은 -1~1 사이의 값을 갖는 실수로써 연산의 수행 과정에 실수 곱셈이 필요하다. 일차원 역 DCT를 위한 수식은 식 1과 같다.

$$Z_k = \frac{c(k)}{2} \sum_{i=0}^{7} \cos\left(\frac{(2i+1)k\pi}{16}\right) \quad (1)$$

여기서, k=0,...,7이다. 그러나 ARM920T 코어 내부에는 실수 곱셈기가 없기 때문에 이를 소프트웨어적으로 변환하여 연산을 수행한다. 이 과정에서 많은 클럭을 소모하며 역 DCT 블록의 동작속도를 저하시킨다. 따라서 역 DCT 연산에서 사용되는 실수의 코사인 테이블을 정수화 함으로써 수행시간을 줄이는 방법이다. 표 3에 역 DCT 연산에 사용되는 실수 코사인 테이블과 수정된 정수 코사인 테이블을 나타내었다.

표 3 실수형 코사인 테이블과 수정된 코사인 테이블
Table 3 Comparison of real-number cosine table and modified cosine table

Real number cosine table	Positive number cosine table
0.7071068	2896
0.4903926	2009
0.4619398	1892
0.4157348	1703
0.3535534	1448
0.2777851	1138
0.1913417	784
0.0975452	400

표 3에의 정수 코사인 테이블은 실수 코사인 테이블에 4096(2^{12})을 곱해줌으로써 얻어진 것이며, 역 DCT 연산시 요구되는 실수 연산을 정수 연산으로 대신하여 수행한다. 역 DCT 연산을 수행한 후 곱해주었던 4096(2^{12})을 나누어줌으로써 정확한 값을 구한다. 실수 연산을 정수 연산으로 변형하는 경우 라운딩 오차가 발생하게 되나 이는 +1~-1 사이의 값을 갖게되고 규약에 정의된 오차범위를 만족한다.

또한 MPEG-4의 내부 파라메터 중 CBP 파라메터는 휘도 성분을 나타내는 CBPY와 색차 성분을 나타내는 CBPC로 구성된다. 이 파라메터는 매크로 블록 내부에 데이터의 존재 유무를 나타내는 파라메터로서 하드웨어 구현 관점에서는 참조 하지 않고 처리하는 것이 구현의 편의성 증대와 면적을 줄일 수 있는 방안이다. 그러나 소프트웨어 구현 관점에서는 이 파라메터를 참조하여 데이터가 없는 경우, 이후의 역 지그재그 스캔 블록과 AC/DC 예측 복호화 블록, 역 양자화 블록, 역 DCT 블록을 수행하지 않음으로써 연산량을 감소시킬 수 있다. 본 논문에서는 CBP 파라메터를 참조하여 데이터가 0 값을 가지는 경우 역 지그재그 스캔 블록에서 0 값을 채워주고 AC/DC 예측 복호화 블록과 역 양자화 블록, 역 DCT 블록을 수행하지 않음으로써 연산량을 줄였다.

아키텍처 레벨의 최적화에서는 ARM 아키텍처의 특성을 이용하여 함수의 인자 개수를 4개 이하로 유지하여 외부 메모리로의 접근 횟수를 줄이는 방식과 여러 개의 블록을 단일 프로세스 내에서 처리하는 방식을 적용하여 동작 시간을 감소 시킬 수 있다.

ARM 프로세서에서는 R0~R3의 네 개의 레지스터를 함수의 인자 값 전달에 사용한다. 네 개를 초과하는 인자 값은 스택을 통하여 함수에 전달되는데 스택은 외부 메모리로써 접근하는데 많은 클럭이 소모된다. 따라서 함수의 인자 값을 네 개 이하로 유지함으로써 인자 값을 내부 메모리에서 바로 사용 할 수 있도록 하였다. 임베디드 프로그램은 외부 메모리로 접근 하는데 많은 클럭을 필요로 한다. 같은 데이터를 사용하는 블록의 경우 하나의 프로세스 내에 위치시키면 메모리 접근 횟수를 줄일 수 있다. 역 지그재그 스

캔 하여 새로 구성된 데이터를 같은 프로세스 내에서 역 양자화 하고, 역 DCT 연산을 수행하며 움직임 보상 블록과 VOP 재구성 블록을 같은 프로세스에서 처리함으로써 연산량을 줄일수 있었다. 또한 연산에 사용되는 고정된 길이의 반복문을 펼쳐서 사용함으로써 연산량을 감소 시켰다. 또한 MPEG-4의 블록 중 역 DCT는 인트라 VOP와 인터 VOP가 같은 수식을 사용하고 데이터의 움직임 보다는 직접 연산량이 많다. 역 DCT 블록의 경우 C로 작성된 코드를 어셈블리로 변환하여 수행 시간을 줄일 수 있었다. MPEG-4 Simple Profile 부호화기의 실시간 구현 시 목표한 성능에 도달하지 못한다면 이러한 최적화를 반복적으로 수행한다. 이러한 최적화 기법을 사용함으로서 연산량에 의한 전력 소모를 줄일 수 있고, 동작 속도의 향상을 기대할 수 있다.

5. 실험 및 결과

본 논문에서 구현한 MPEG-4 Simple Profile 복호화기의 정확성을 입증하기 위하여 MPEG 워크 그룹 중 ACTS-098 MoMuSys에서 제공하는 복호화기와 비교를 실시하였다. MoMuSys 부호화기에서 비트스트림을 생성하고 PC 상에서 MPEG-4 Simple Profile 복호화기를 구현하여 출력 영상을 저장한 후 MoMuSys 복호화기의 출력 영상과 ARM920T 테스트 보드의 출력 영상과 비교하여 정확성을 검증하였다. 즉, PC 상에서 구현된 알고리즘을 ARM에서 제공하는 ADS를 이용하여 ARM 코어에 적합한 코드로 변환시키고 컴파일러를 이용하여 바이너리 파일을 생성하였다. 생성된 바이너리 코드를 ARM 에뮬레이터인 ARMulator를 사용하여 원하는 성능을 만족시켰는지를 검토하고, 목표 성능을 이루지 못한 경우 최적화를 수행한 후 실제 보드에 적용하였다. 또한 하드웨어 디버거인 Multi-ICE를 이용하여 바이너리 코드가 실제 보드에 적용되었을 경우에 발생하는 문제점들을 해결하였고, 최종적으로 바이너리 코드를 ARM 코어를 사용한 보드에 적용하여 알고리즘을 수행하였다. 그림 7은 프로그램 개발 및 실험 환경을 나타낸다.

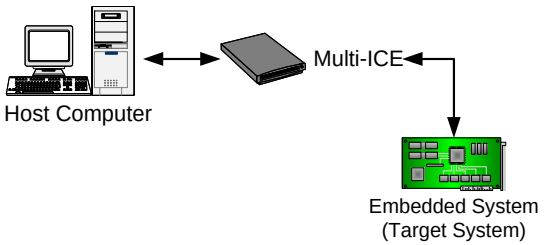

그림 7 실험 환경
Fig. 7 Test environment

테스트에 사용된 영상은 MPEG-4에서 제공되는 영상 중 QCIF(176x144) 크기의 SUZIE 영상을 사용하였고 15프레임을 추출하여 테스트하였다. 표 4는 최적화 이전의 복호화기 실행 정보를 나타내었다.

연산과정이 없고 상황에 따른 분기만 실시하는 비트 스트림 분석 블록과 가변 길이 복호화 블록, 역 지그재그 블록은 적은 클럭을 소모하였으나 연산 수식이 복잡하고 메모리의 이동이 많은 역 DCT, 움직임 보상, 역 양자화, VOP 복원 블록은 많은 클럭을 소모하여 전체 수행시간의 90% 이상을 차지하였다. 복호화 과정에서 가장 많이 차지하는 블록은 역 DCT 과정으로써 실수의 곱셈 연산을 하드웨어 곱셈기 없이 소프트웨어로 변환하여 처리함으로 많은 클럭이 요구되었다. 표 5는 최적화 이후의 복호화기 실행 정보를 나타내었다.

표 4 최적화 이전의 복호화기 실행 정보
Table 4 Execution cycle of the decoding processes before optimization

Execution block	Speed
Bitstream Parsing & VLD	4.2 MHz
I-Zigzag Scan & I-AC/DC Prediction	6.22 MHz
Inverse Quantizer	3.7 MHz
Inverse DCT	49.56 MHz
Motion Compensation	8.4 MHz
VOP Reconstruction	11.76 Mhz

표 5 최적화 이후의 복호화기 실행 정보
Table 5 Decoder process execution cycle after optimization

Execution block	Speed
Bitstream Parsing & VLD	1.86MHz
I-Zigzag Scan & I-AC/DC Prediction	1.04MHz
Inverse Quantizer	0.76MHz
Inverse DCT	4.22MHz
Motion Compensation	12.49MHz
VOP Reconstruction	2.11Mhz

비트 스트림 분석 블록과 가변 길이 복호화 블록은 같은 프로세스 내에 위치시키고, 발생 빈도수가 많은 구문을 상위에 위치시키며, 고정된 수의 반복 구문은 풀어서 연산을 수행함으로써 동작 속도를 감소시킬 수 있었다.

역 지그재그 스캔 블록과 AC/DC 예측 복호화, 역 양자화, 역 DCT 블록을 하나의 프로세스에 통합하여 내부 메모리에서의 연산을 가능하게 하였고 각 블록의 코드 최적화를 통하여 표 5의 동작 속도로 구동시킴으로써 성능을 향상시킬 수 있었다. 또한 가장 많은 클럭이 요구되던 역 DCT 블록은 49.56MHz에서 4.22MHz로 약 12배의 성능 향상이 있음을 알 수 있다. 표 6에 최적화 이전의 실행 시간과 본 논문의 최적화 과정을 수행한 후 테스트 영상 전체의 실행 시간을 비교하였다.

표 6 최적화 전과 후의 복호화기 실행 속도
Table 6 Comparison of decoder processing time

optimization / stream	Before optimization	After optimization
Suzie (Qcif)	83.84MHz	22.47 MHz

표 7은 본 논문에서 제안한 기법의 전력 소모량을 나타내었다. 성능의 비교를 위해 MPEG-4 Simple Profile 복호화기를 ARM7TDMI를 사용하여 구현한 논문을 참조하였다[7].

표 7 전력 소모량의 비교
Table 7 Comparison of power consumption

	[7]에서의 방법	제안된 방법
Power	19.77mW	8.02mW

표 5에서와 같이 기존의 방법에서는 테스트 영상의 복호시 19.77mW의 전력을 소모하였다. 제안된 논문에서는 최적화 과정을 수행한 결과로 8.02mW의 전력을 소모하여 전력면에서 약 60%의 성능 향상을 이루어 이동 기기에 적용이 가능하도록 전력 사용량을 줄일 수 있었다.

그림 8은 실제 보드에 적용된 MPEG-4 Simple Profile 복호화기를 나타내었다.

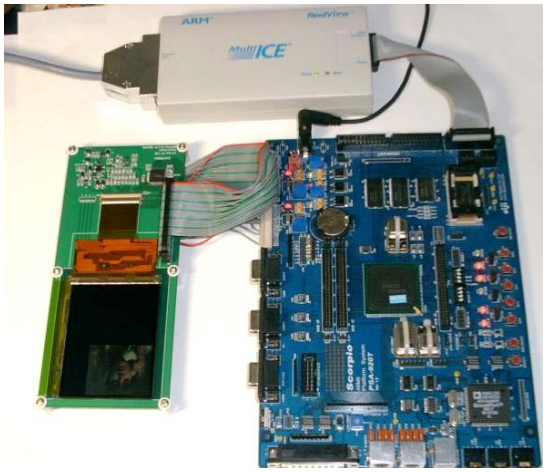

그림 8 MPEG-4 simple profile 복호화기
Fig. 8 MPEG-4 simple profile decoder

5. 결 론

본 논문에서는 임베디드 ARM 프로세서를 이용한 MPEG-4 Simple Profile 복호화기의 실시간 구현에 관해 연구하였다. 현재 많이 사용되는 코어인 ARM7TDMI 코어와 ARM9TDMI 코어 중 전력 사용량이 적고 동작속도가 빠른 ARM9TDMI 코어를 선택하였으며, 구현 과정에서 보다 적은 전력으로 동작시키기 위하여 MPEG-4의 알고리즘 적용의 최적화를 이용한 방식과 ARM 코어의 특성을 이용한 최적화 과정을 수행하였다. 그 결과 최적화 이전의 MPEG-4 Simple Profile 복호화기의 동작 속도는 83.84MHz이었으나 최적화 이후에는 22.47MHz로 동작하여 약 3.7배의 수행 성능을 향상시킬 수 있었고, 전력 소비면에서 기본의 방법에 비해 약 60%의 성능 향상을 이룰 수 있었다.

향후 연구 과제로는 MPEG-4 Simple Profile 부호화기를 ARM 코어로 구현하여 MPEG-4 부호화기/복호화기를 통합 구현하고, 더 나아가 MPEG-4의 다른 Profile을 지원하는 부호화기/복호화기로 확장하는 것이다.

참 고 문 헌

[1] ISO/IEC JTC1 CD 11172, Coding of moving pictures and associated audio for digital storage media up to 1.5 Mbits/s, ISO, 1992.
[2] ISO/IEC JTC1 CD 13818, Generic coding of moving pictures and associated audio, ISO, 1994.
[3] ISO/IEC 14496-2:1999/Amd.1:2000(E), Information technology - Coding of audio-visual objects, ISO/IEC 2000.
[4] K. R. Rao and Zoran S. Bojkovic, Packet Video Communications over ATM Networks, Prentice-Hall Inc., 2000.
[5] ITU-T Recommendation H.263, Video coding for low bitrate communication, ITU-T, May 1996.
[6] ARM920T Technical Reference Manual , ARM DDI 0151C, April 2001.
[7] K. Ramkishor, Real Time Implementation of MPEG-4 Video Decoder on ARM7TDMI , Digital Multimedia Group, Bangalore Indea.

저 자 소 개

박 성 욱 (朴 成 煜)
1999년 인천대 대학원 전자공학과 졸업(석사). 2003년 인천대 대학원 전자공학과 졸업(박사). 2002～현재 (주)로직메카 멀티미디어 통신연구소 선임연구원
Tel: 032-770-8441
Fax: 032-764-2371
E-mail : psw@incheon.ac.kr