DOI: 10.3745/KIPSTA.2008.15-A.5.269

# 스크래치 패드 메모리의 압축을 통한 저전력 임베디드 시스템의 구현

서 효 중†


## 요 약

최근 임베디드 시스템의 고성능화에 따라 고해상도의 디스플레이를 채용하고 대용량 멀티미디어 데이터응용 등 다기능을 갖춘 임베디드 프로세서가 다수 발표되고 있다. 이러한 응용 중 실시간 오디오 스트리밍 같은 시간 제한적 응용을 다루어야 하는 모바일 시스템의 경우, 전력, 메모리 용량등 여러 자원이 부족한 상황에 놓이게 된다. 본 논문은 스크래치 패드 메모리에 대하여 활용도를 높이고 저전력을 구현하기 위하여 압축 기법을 스크래치 패드 메모리의 데이터 영역에 구현하였다. 무선 통신과 실시간 오디오 스트리밍에 사용하는 GDM1202 프로세서에 제안한 방법을 구현하여 측정한 결과, 압축으로 얻어진 스크래치 패드의 영역에 코드와 데이터를 추가하여 할당함으로써 13.3% 에너지 절감을 얻을 수 있었으며, 기존의 방법과 동등한 프로그램 수행 성능을 나타냈다.


**키워드 : 임베디드 프로세서, 메모리압축, 스크래치패드 메모리**

# Implementation of A Low-Power Embedded System via Scratch-pad Memory Compression

Hyo-Joong Suh†


## ABSTRACT

Recently, lots of embedded processors which can run streaming multimedia with high resolution display are introduced. Among the applications running on these embedded processors, real-time audio streaming is one of the applications that suffer from the lack of energy and memory space. In this paper, we propose a novel data compression method on scratch-pad memory, which saves both useful space on the scratch-pad memory and energy. We have implemented the data compression scheme on the GDM1202 real-time audio streaming processor, and the performance results show that we obtained 13.3% energy saving while maintaining comparable application performance to that of the non-compression case.


**Keywords : Embedded Processor, Memory Compression, Scratch-Pad Memory**

## 1. 서 론

최근 임베디드 시스템(embedded system)에 대한 관심이 높아지고 해당 시스템의 성능이 증가함에 따라, 임베디드 시스템에서도 메모리를 관리하는 것이 중요한 이슈로 떠오르고 있다. 반도체 미세공정 기술의 개선에 따라 프로세서의 속도가 높아지고, 메모리의 크기도 확대되었으며, 응용 프로그램의 복잡도 및 필요로 하는 통신 대역폭도 높아지고 있는 추세이다. 필연적으로 에너지 소모량은 지속적으로 증가하고 있으며, 용량의 확대와 달리 접근 시간의 개선은 상대적으로 미진한 편이다. 또한, 프로세서와 메모리간의 성능 격차로 인해서, 프로세서가 메모리에서 데이터가 처리되는 것을 기다리는 경우가 종종 발생하게 되어 시스템의 성능이 저하되는 현상이 빈번하게 발생하고 있다[1]. 이러하게 디지털 시스템에서 메모리는 성능과 전력 소모의 주요한 병목 지점이 되고 있다. 고성능 컴퓨터 시스템에서는 이러한 문제를 해결하기 위해서 기본적으로 프로세서 내부에 고속의 캐시메모리 크기를 확대하여 내장함으로써 외부 메모리로의 접근을 최소화 하여 성능향상 및 전력 소모의 절감을 달성하였다.

이와 같은 맥락에서 메모리 압축기법 또한 다양하게 연구되었다. 디지털 시스템의 메모리는 크게 프로그램과 데이터 영역으로 볼 수 있는데, 이러한 영역 중 많은 부분이 0의 값을 갖고 있으며[2], 이러한 의미는 메모리의 압축을 통하여 물리적 메모리 용량에 비하여 보다 큰 논리적 공간을 활용할 수 있음을 의미하고, 압축된 데이터를 전송함으로써 데이터의 전송을 위한 신호의 변화량을 줄일 수 있어 이로 인한 전력 소모의 절감을 얻을 수 있음을 뜻한다[3]. 반면에

※ 본 연구는 2007년도 가톨릭대학교 교비연구비의 지원으로 이루어졌음.
† 종신회원 : 가톨릭대학교 컴퓨터정보공학부 조교수
논문접수 : 2007년 8월 21일
수 정 일 : 1차 2008년 6월 3일, 2차 2008년 7월 14일
심사완료 : 2008년 7월 25일

메모리의 압축은 압축하기 이전의 데이터의 물리적 위치와 압축 후의 데이터의 물리적 위치간의 연계성을 직접적으로 유지하기 힘들며, 압축과 압축 해제로 인한 부가적 시간 지연이 발생하게 됨으로써 시스템 전체의 성능 저하를 나타낼 수 있다[4].

일반적인 메모리의 압축은 고성능 시스템에서 다양하게 연구 및 구현된 바 있으며, 압축 대상에 따라서 캐시 메모리를 압축하는 경우[5], 메인 메모리를 압축하는 경우[1], 가상메모리 상에서 페이지를 압축하는 기법[6] 등 다양하게 소개되고 있다.

모바일 환경에서 사용하는 임베디드 시스템의 경우, 일반적 고성능 시스템과 달리 일차 또는 이차전지를 사용하는 경우가 일반적이며, 이로 인한 저전력 요구 및 저용량의 메모리에서 원활하게 수행할 수 있도록 구현되어야 한다. 한편, 임베디드 시스템에서는 시스템이 기능하는 응용의 시간 요구를 만족시킬 수 있는 경우, 보다 고속의 처리능력을 요구하지 않아도 시스템 요구사항을 만족시킬 수 있으므로, 전력 절감 및 메모리 용량 절감을 달성할 수 있는 메모리 압축 시스템을 적용하기 적합하다.

본 논문은 이러한 모바일 임베디드 프로세서에서 프로세서 내부에 스크래치패드 메모리에 대한 데이터 압축을 활용하여 시스템의 전력 소모량을 줄이고 압축 해제로 인한 시간지연이 발생하기는 하나, 모바일 임베디드 시스템의 외부 메모리 접근 시간이 상대적으로 느림에 착안하여 성능 감소를 최소화 하는 개선 방법을 제안하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 메모리 압축 기법과 관련된 연구를 살펴보며, 3장에서는 제안하는 시스템의 전체구조와 동작방식, 그리고 제안하는 기법의 특성 등에 대해서 기술한다. 4장에서는 실험환경 및 방법, 그리고 제안하는 시스템과 비교대상 시스템의 성능을 측정해서 평가한다. 끝으로 5장에서는 결론으로 본 논문을 마무리한다.

## 2. 관련 연구

초기에 연구된 메모리 압축 기법은 교체되어 나가는 페이지 단위로 메모리 압축을 수행한 것으로써 캐시나 메모리에서 데이터의 압축 및 복원이 이루어지게 되고, 보통 메모리의 크기를 늘려서 페이지 폴트율을 감소시키기 위해서 압축 기법을 사용한다. 데이터의 압축 및 복원은 운영체제에 의해 소프트웨어적으로 구현되었으며, 메모리의 일부 영역에 압축된 페이지를 저장하는 소프트웨어 압축 기법이 연구되었다[6][7][8][9]. 그러나 압축 및 복원에 소요되는 시간으로 인해 처리속도가 늦어지기 때문에 페이지 시스템에서 상당한 부가 지연을 유발시켰으며, 실제 응용 프로그램에 대해서 성능 향상이 미미하였다.

반면에 하드웨어에 기반한 압축 및 해제를 이용한 방법으로 IBM의 MXT(Memory Expansion Technology)는 하드웨어적으로 구현한 메인 메모리 압축 시스템으로써 LZ에 기반한 알고리즘[10]을 구현하여, 외부의 3차캐시와 메인 메모리 사이에 압축기를 두어 3차 캐시로부터 프로세서 간에는 압축되지 않은 데이터를 관리하고, 3차 캐시와 주 메모리로 전송할 때는 페이지 단위의 압축을 하여 저장하며, 압축된 메인 메모리의 데이터를 읽어들일 때, 압축해제를 수행하여 읽어들임으로써 두 배의 메인 메모리의 확대효과를 꾀한 것이다[11]. 그러나 이 방법에서도 메모리 접근은 메모리상에 상주하는 테이블을 먼저 액세스 한 후에 이 테이블로부터 참조하는 구조이므로 메모리 액세스 지연이 두 배가 되며, 캐시에 비해 워킹 셋이 커질 경우 성능 저하가 일어난다. 또한 이러한 메모리 접근을 구현하기 위하여 운영체제 수준에서 메모리 크기와 주소를 대응 시켜줄 수 있어야 한다.

메인 메모리가 아닌 캐시에 압축된 데이터가 저장되는 형태로 선택적 압축 메모리 시스템(SCMS :Selective Compressed Memory System)[12]과 FVCS(Frequent Value Cache Scheme)[13], 그리고 FPCS(Frequent Pattern Compression Scheme)[14]이 있다. SCMS는 선택적인 데이터 압축과 압축 데이터들에 대한 고정위치 할당 방법, 복원시간을 효과적으로 줄이기 위한 기법을 적용한 것이 특징이다.

선택적 압축이란 압축의 대상이 되는 각 데이터 블록의 압축률이 미리 정해진 임계 압축률보다 작을 경우에만 압축이 이루어지는 것을 의미한다. 고정위치 할당 방법은 공간적인 손실을 감수하고 시간과 압축 효율성 면에서 이득을 얻기 위한 방법으로 압축 블록을 L2 캐시와 주 메모리에 일정한 크기의 공간을 할당하여 저장한다. SCMS는 압축 알고리즘으로 연속적인 '0'의 값을 포함한 블록이 압축된 경우 복원 시간을 줄일 수 있는 X-RL 알고리즘[15]을 사용하였다. 또한 일정 크기의 복원 버퍼를 두어 반복적으로 값이 참조될 경우, 저장되어 있는 블록을 제공함으로써 데이터가 복원되는 시간을 효과적으로 줄였다.

FVCS는 작은 개수의 값들이 메모리에 접근할 때 빈번하게 사용되는 것과, 바이트 크기로 많은 양을 차지하는 '0'값을 압축하였다. 데이터 캐시는 직접 사상되는 캐시와 자주 접근되는 값을 저장하는 캐시로 구분하여 적어도 처리되는 데이터의 50% 이상을 압축된 데이터가 차지하게 만들었다.

FPCS는 FVC(Frequent Value Compression) 기법에 중요성 기반 압축 기법의 특징을 적용하여 제안한 기법이다. 1차 캐시에는 압축하지 않은 데이터를 저장하며, 2차 캐시에 저장할 때 압축된 값을 저장하는 구조이다. 압축의 형태는 3비트의 프리픽스(prefix)값을 이용하여 총 8가지의 패턴을 데이터의 크기에 따라 대응되도록 하여 압축 하였다.

스크래치 패드 메모리는 캐시와 달리 메모리 주소를 가지는 영역으로써 캐시 이상의 고속 접근을 구현할 수 있으나, 빈번히 접근되는 데이터의 크기에 비하여 상대적으로 작은 크기로 구현되는 경우가 일반적이다. 그러나 정적인 메모리 주소를 할당받아야 하므로 컴파일러 및 운영체계가 스크래치 패드 메모리 영역을 직접 관리해 주어야만 한다. 이러한 스크래치 패드 메모리에 빈번히 접근되는 데이터를 적절하게 배치시키기 위한 연구도 다양하게 이루어졌다.

관리의 형태로 성능향상을 꾀하는 방법으로는 정적[16] 또는 동적[17]으로 관리하는 방법이 제시되었고, 관리의 대

상도 데이터에 대한 부분[18]과 명령과 데이터를 모두 대상으로 하는 방법[19] 또한 제시되었다. 이러한 연구는 스크래치 패드 메모리에 대한 소프트웨어적인 관리 방법에 초점을 맞추고 있다. 마찬가지로 컴파일러에 기반한 휴리스틱을 이용하여 프로그램이 접근하는 데이터 벡터를 접근 패턴에 따라 구별하여 데이터의 접근 흐름에 따라 작은 블록으로 구별하여 블록의 접근 순서에 따라 메인 메모리와 스크래치패드 메모리의 압축 및 비압축 데이터 영역으로 스케줄링하는 기법도 제안된 바 있다[20].

## 3. 압축 데이터의 스크래치 패드 메모리 적용

이 장에서는 무선 실시간 응용 스트리밍 오디오를 구현한 응용에 기반하여, 이러한 시스템에 적용할 수 있도록 개발된 스크래치 패드 메모리 구현 방법을 서술한다.

### 3.1 목적 시스템

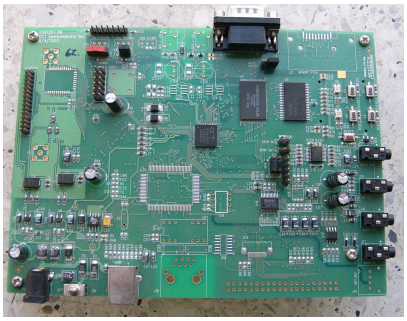
제안하는 방법이 적용될 대상은 블루투스 무선 채널을 통하여 실시간 오디오 스트리밍을 구현할 수 있는 GCT Semiconductor의 GDM1202 프로세서[21]로 한다. 목적 프로세서의 사양은 <표 1>과 같다.

내부의 스크래치 패드 메모리는 다시 6개의 블록으로 분할되어 블루투스 베이스밴드 프로그램 중 성능에 대해 민감한 서브루틴과 데이터를 사용하게 되며, USB, 오디오 코덱, NAND 스토리지 등 각종 입출력장치의 고속 데이터 전송을 위한 DMA 전송용 버퍼 메모리로도 활용된다.

(그림 1)은 본 논문에서 사용할 대상 프로세서의 개발 보드로써 블루투스 기본 프로파일과 함께 제안하는 스크래치 패드 메모리의 시험 환경을 적용할 개발 플랫폼이다.

〈표 1〉 GDM1202 임베디드 프로세서 사양

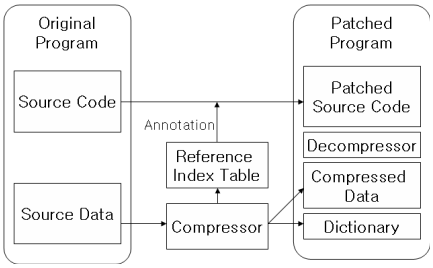
| | |
|---|---|
| Processing core | Vincent Architecture |
| | 32bit RISC/24bit DSP |
| Operation clock | 128MHz |
| Internal cache | 16KB |
| Scratch-pad size | 128KB |
| RF interface | Bluetooth spec 1.2 |
| Applications | Real-time Audio Streaming |
| Implemented codec | MP3, SBC |
| Operating System | VPOS |
| Chain tool | GNU-based toolkit |
| Prod. Technology | 0.13um CMOS |



(그림 1) GDM1202 개발 플랫폼

### 3.2 압축된 데이터의 생성과 프로그램 수정

(그림 2)는 본 연구에서 제안하는 프로그램의 수정 과정을 도시한 것이다. 우선 목적 시스템이 임베디드 시스템이기에 일정한 응용만을 수행하는 형태이므로, 목적 시스템에서 필요로 하는 부분은 압축에 대한 해제기와 압축을 해제할 자료가 될 사전이다. 소스 프로그램의 수정단계는 일단 원 프로그램에서 데이터 부분을 압축기를 통하여 압축하며, 이 과정에서 압축된 데이터와 사전 그리고 각 데이터의 참조 테이블을 생성해 낸다. 참조 테이블은 관리되는 단위 데이터의 참조 지점으로써, 참조가 일어나는 각 내용에 대하여 압축 이후에 대한 데이터의 참조 위치를 대응하여 유지한 것이다. 이 참조 테이블은 원 프로그램에서 데이터를 참조하는 부분에 압축되어 있는 위치를 대응하여 생성하며, 소스코드를 수정하기 위하여 사용되고 목적 바이너리에는 포함되지 않는다. 소스코드 중 데이터를 참조하는 부분은 압축 해제기를 호출하는 것으로 대체되며, 이때 참조 데이터의 크기와 압축된 위치에 대한 값이 파라미터로 전달된다. 압축해제기는 전달받은 값에 따라 압축되어 있는 데이터에서 필요로 하는 부분을 압축 해제하여 반환하여 전달하게 된다.




(그림 2) 프로그램의 수정 과정

### 3.3 스크래치 패드 메모리 할당

본 연구에서 사용한 프로세서는 메모리관리장치가 없고 가상메모리를 사용하지 않는 실시간 운영체제인 VPOS[21] 상에서 구현된 경우이기에, 스크래치 패드 메모리의 할당에 대하여, 우선 각 프로그램의 컴파일을 수행하고, 각 모듈에 대하여 필요로 하는 성능을 개발 툴 상의 프로파일러를 통하여 도출하며, 이 결과에 따라서 필요로 하는 서브루틴에 대하여 링크 스크립트를 생성하여 스크래치 패드 메모리 및 각 서브루틴에 대한 할당을 수행하게 된다.

시험할 대상이 되는 응용 프로그램은 블루투스 오디오 스트리밍 응용을 이용하였다. 이 응용에서 수행하는 프로그램 모듈 부분은 크게 무선 통신을 유지하기 위한 베이스밴드 소프트웨어와 프로토콜의 상위 계층에 해당되는 관리자 프로그램인 프로파일을 구현하여야 하며, 전송하는 스트리밍 데이터가 압축된 형태이기에, 이를 압축 및 압축해제 하기 위한 SBC 코덱 [22] 및 MP3 디코더가 있다. 목적 시스템에

배치할 수 있는 메모리는 스크래치 패드 메모리와 메인 메모리가 있으며, 전체 프로그램을 컴파일 한 결과를 개발 체인 툴에 포함된 프로세서 명령어 시뮬레이터를 통하여 해당 프로그램에서 각 모듈 및 데이터가 접근되는 빈도와 필요로 하는 명령어 사이클 수를 도출하게 된다. 이와 같이 도출된 빈도를 기반으로 스크래치 패드 메모리 및 외부의 메모리에 대응시켜 링커 스크립트를 생성하며, 다시 대응된 메모리 접근 시간을 적용하여 응용 프로그램의 시간 요구 조건 만족을 시험하고, 실제 하드웨어에 탑재하여 동작성을 검증하는 순서로 진행한다.

응용 프로그램의 바이너리를 생성하기 위하여 위와 같은 시뮬레이션 프로파일링을 통한 성능 분석 및 실시간 조건을 만족시키는가의 시험 단계 및 검증이 이루어져야 하기에, 기존의 컴파일러 상에서 메모리 접근 패턴에 대한 휴리스틱을 이용한 스크래치 패드 할당 방법[20]을 적용하기 힘들다. 그러나 본 논문에서 제안하는 방법은 이와 같은 실시간 조건을 만족시켜야 하는 경우에, 압축 해제기를 포함한 시뮬레이터 진행과 프로그램의 관리 단위와 일치하는 펑션 및 서브루틴별 메모리 할당을 용이하게 이룰 수 있는 장점이 있다.

### 3.4 데이터의 압축

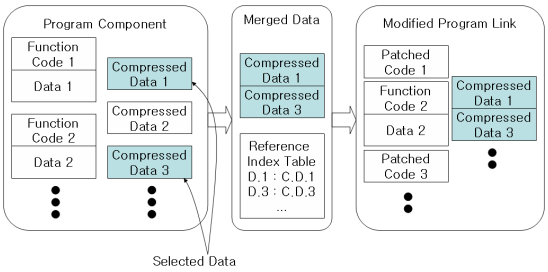
제안하는 시스템에서 메모리 압축을 효과적으로 수행하기 위해서는 압축 알고리즘의 압축률이 높고 압축속도가 빠르며, 압축 및 복원시 발생하는 부하가 적어야 한다. 따라서 본 논문에서는 제안하는 메모리 압축 기법의 기본 압축 알고리즘으로 LZ 알고리즘의 변형인 LZW(Lempel-Ziv-Welch) 알고리즘을 사용하였다. LZW 알고리즘[23]은 Lempel과 Ziv의 아이디어를 Welch가 효과적으로 구현한 것으로써, 여러 심볼을 묶은 가변길이 심볼열을 가변길이 부호로 표현하는 방법이다.

본 논문에서 제안한 방법의 경우 압축 프로그램은 실행 바이너리에 필요하지 않기에, 개발 환경에서 수행하는 압축 프로그램을 구현하였으며, 소스 프로그램에서 사용하는 여러 데이터를 동일한 사전을 이용하도록 압축한다. 응용에서 사용하는 데이터 블록은 베이스밴드가 사용하는 것과 프로파일이 사용하는 것, 코덱이 사용하는 것 등 모듈 별로 분리되어 있으나 모든 데이터를 하나의 사전을 생성하도록 한다. 이 결과 원 프로그램은 각 펑션과 압축되지 않은 데이터, 그리고 압축된 데이터, 사전이 생성되며 이 단계에서 목적 파일을 생성하여 시뮬레이터를 통한 프로파일링을 수행한다.

이제 프로파일링의 결과로 얻은 각 펑션의 수행 빈도와 수행 명령어를 이용하여 응용의 시한성을 만족시키는 메모리배치를 수행하는 단계가 된다. 각 모듈에 대한 메모리 지연값을 스크래치 패드 메모리와 외부 메모리에 대응한 결과 및 압축된 데이터를 참조하는 경우에 대한 지연값으로 대응시켜 검증하게 되는데, 압축 데이터를 액세스 하는 경우, 스크래치 패드 메모리 상에 할당한 압축 해제 프로그램을 수행하여야 하므로, 생성된 사전과 압축되어 있는 데이터를 참조하여 메모리 시간 지연값을 산출하게 된다.

이러한 결과로 스크래치 패드 메모리에 할당할 데이터를 선별하게 되며, 데이터가 선별되면, 해당되는 압축 데이터를 선택하여 하나로 합치는 과정을 수행하며, 데이터 참조 테이블을 생성하게 된다. 데이터 참조 테이블은 스크래치 패드 메모리 상에 탑재할 데이터 위치를 참조할 수 있도록 원 펑션 코드를 수정하는 데 사용된다. 메모리 참조 부분은 압축 해제 프로그램을 호출하는 것으로 대체되며, 파라미터로 해당되는 메모리 위치 및 메모리 크기를 전달하게 된다.

다음 (그림 3)은 데이터의 압축과 스크래치 패드에 올릴 데이터의 선택 및 합쳐진 바이너리를 생성하는 과정을 제시한 것이다.




(그림 3) 압축 데이터의 선택 및 바이너리 생성과정

## 4. 성능 평가

### 4.1 실험 환경

성능 평가를 위하여 사용한 응용 프로그램은 앞서 제시한 개발 플랫폼 상에서 구현된 실시간 오디오 스트리밍 프로그램으로 하였으며, 다음 표 2 는 이러한 응용 프로그램에서 사용하는 데이터를 연속된 크기에 따라 정리한 것이다. 표에서 나타나는 바와 같이, 총 연속된 데이터의 개수는 917개이나, 전체 데이터의 총량에 비하여 차지하는 비율은 큰 데이터가 상대적으로 높음을 알 수 있다. 또한 베이스밴드 통신과 코덱에서 사용하는 정적 데이터의 크기가 상대적으로 크고 이러한 데이터가 시간 제한성을 갖고 접근되어야 하는 경우가 발생하는 경우를 우선 선택하여 스크래치 패드 메모리로 할당하도록 하는 방법을 취하였다.

각 펑션은 앞서 설명한 방법을 통하여 데이터의 압축 및

〈표 2〉 응용에 사용된 데이터 특성

| 데이터 크기 종류 | 데이터 총합 | 개수 | 비율(%) |
|---|---|---|---|
| 100 바이트 미만 | 12,587 | 343 | 3.5 |
| 100이상, 200미만 | 27,353 | 203 | 7.7 |
| 200이상, 400미만 | 62,305 | 180 | 17 |
| 400이상, 1K미만 | 98,205 | 130 | 27 |
| 1K 이상 | 155,193 | 61 | 43 |
| 합계 | 355,643 | 917 | 100 |

〈표 3〉 스크래치 패드 메모리 할당

| 종류 | 코드 영역 크기 | 데이터 영역 크기 |
|---|---|---|
| 기존 방법 | 59KB | 37KB |
| 데이터 압축 | 75KB | 21KB |

프로파일링을 통한 데이터의 선별 과정을 거쳤으며, 마찬가지로 시뮬레이터에 의하여 베이스밴드 프로토콜 처리 및 코덱을 위한 펑션의 필요에 따라 스크래치 패드 및 외부 메모리에 대한 배치를 수행하였다. 위에서 제시한 데이터 특성에 의하여 압축 및 압축을 하지 않은 경우에 대하여 데이터를 배치한 결과의 비교는 다음 표 3과 같다.

128KB의 스크래치 패드 메모리 영역은 일단 응용을 위한 입출력 장치 데이터의 DMA 전송을 위한 공간 32KB를 제외하고 사용하게 되며, 나머지 96KB의 공간을 코드와 데이터가 차지하게 된다. 이 메모리 영역에서는 기존 방법에 비하여 데이터 압축을 통하여 44% 정도의 데이터 압축이 이루어 졌으며, 압축 해제를 위한 프로그램으로 2KB 가량의 코드가 부가된다. 따라서 실제 여분의 스크래치 패드 메모리 영역은 18KB 가 확보되었으며, 이 영역에 추가로 자주 호출되는 프로그램 코드와 데이터를 부가하였다.

기존 방법의 경우 데이터의 액세스에 있어서 스크래치 패드 메모리는 지연 없이 즉시 접근되나 외부 메모리는 상대적으로 많은 사이클 지연을 발생시킨다. 반면에 본 논문에서 제안한 방법의 경우 압축 해제를 위한 함수를 호출하게 되므로, 압축 해제 함수의 수행 시간 및 압축된 데이터를 접근하는 지연이 부가되게 된다. 이와 같은 접근 시간의 차이와 외부 메모리에 대한 접근이 발생하는 정도에 따라서 성능 차이 및 소모전력의 차이가 나타나게 될 것이며, 보다 자세한 내용은 다음 절에서 서술한다.

### 4.2 실험 결과 및 성능 분석

논문에서 제안하는 방법의 적용으로 얻을 수 있는 가장 큰 이득은 두 가지로 볼 수 있다. 첫 번째는 외부 메모리를 접근해서 수행하여야 하는 횟수를 감소시키고 반대로 스크래치 패스 메모리를 접근하는 횟수를 증가시킴으로써 외부 메모리를 구동하는 데 필요한 소모 전력을 줄이는 것이다. 두 번째는 상대적으로 느린 외부 메모리에 대비하여, 고속의 스크래치 패드 메모리의 접근으로 대치됨으로써 성능 효과를 얻을 수 있으나, 이는 줄어든 외부 메모리의 지연 시간과 내부의 압축 해제 펑션의 수행으로 인한 부가적 지연 발생이 비교되어야 할 부분이다. 이와 같은 두 가지 이득은 모두 압축을 통하여 스크래치 패드 메모리의 공간을 확대하는 효과를 이용하여 더 많은 코드와 데이터를 스크래치 패드 메모리에 배치시킴으로써 얻을 수 있는 것이다.

다음 <표 4>는 기존 방법과 데이터 압축을 이용한 방법을 사용하였을 때 얻을 수 있는 전력 소모를 대비한 것이다. 우선 기존 방법으로 시스템 전체의 전력을 디지털 부분과 RF 부분을 구분하여 측정하였으며, RF 부분의 전력소모는 제외하고 디지털 부분만의 전력 소모로 상대적 비교를 수행하였다.

압축만 사용한 경우는 기존 방법과 동일한 단계를 거치고, 압축할 데이터를 선택한 후 프로그램의 수정을 거쳐 압축 해제 프로그램을 호출하여 수행하도록 수정한 후 여분으로 생기는 스크래치 패드 메모리를 사용하지 않고 비워 둔 것이다. 두 번째로 압축 및 추가 할당을 수행한 경우는 위와 같이 비워지는 영역에 대하여 코드 및 데이터의 호출 빈도에 따라서 추가적으로 코드와 압축 데이터를 스크래치 패드 메모리에 배치한 경우이다.

표에서 제시되는 바와 같이, 왼쪽 압축만을 사용한 경우 외부 메모리에 대한 접근 패턴은 기존 방법과 차이가 나타나지 않게 되며, 따라서 압축 해제를 위한 부가적 지연과 이로 인한 전력 소모의 증가가 10.9% 가량 손실이 있음이 나타난다. 반면에, 압축을 통하여 추가적인 코드와 데이터를 스크래치 패드에 할당하는 경우 외부 메모리 구동이 줄어든 효과에 의하여 13.3%의 전력 소모가 줄어들었다.

다음 <표 5>는 프로그램의 수행 성능을 비교한 것이다. 수행 성능의 비교는 본 성능 측정에서 대상으로 한 응용이 베이스밴드 무선 통신 프로토콜 처리와 코덱 처리 및 DMA 입출력을 포함하고 있기에, 이를 통합한 프로그램에 대하여 메모리 내 배치를 완성한 후에 해당 프로그램 바이너리를 완성하고, 이 바이너리를 실제 수행하고 수행 중 각 프로세스가 수행하고 남는 프로세서의 유휴 시간을 기준으로 측정하였다. 즉, 통신 프로토콜 및 실시간 전송되는 데이터의 처리는 통신 패킷의 주기적 전송과 이에 부연된 DMA 데이터 전송 및 주기적인 코덱 호출을 수반하기에, 각 프로세스를 수행하는 시간의 합이 해당 시간 주기 내에 처리를 만족할 수 없게 되면 응용의 수행이 불가능하게 된다. 따라서 응용을 수행에서 프로세서가 유휴 상태로 유지되는 사이클의 비율을 측정하면 역으로 프로그램의 수행을 위하여 필요로 하는 시간이 측정된다. 실험 결과는 스트리밍을 구현할 수 있는 몇 가지의 프로파일을 이용하여 통신을 초기화 하였으며, 접속이 완결되고 안정되게 스트리밍이 지속되는 동안의 유휴 상태의 비율을 측정한 것이다.

기존의 방법에서 동일한 펑션 및 데이터만을 스크래치 패드 메모리에 배치시킨다고 할 때, 해당되는 데이터 접근을 위해 압축 해제 호출을 수행하는 지연으로 인하여 프로그램의 수행 성능은 19% 저하되었으며, 앞서의 표 4와 같이 볼 때, 기존 프로그램에서 압축만을 사용하고 발생하는 여분의 공간을 이용하지 않는 경우 전력 소모 및 수행 성능 모두에서 큰 손실이 나타남을 알 수 있다. 반면에, 압축 및 추가할당을 통하여 외부 메모리에 대한 접근 횟수를 줄인 결과 압축 해제로 인하여 추가로 발생된 시간 지연과 상대적으로 훨씬 큰 외부 메모리의 시간 지연간의 상쇄 효과로 거의 유사한 수행 사이클 비를 나타내게 되었다. 물론 이와 같은 결과는 해당 응용 프로그램의 수행 패턴과 내부의 지역적 참조 특성과 상당한 연관관계를 갖게 될 것이다.

〈표 4〉 디지털 부분의 전력 소모 비율

| 기존방법 대비 | 압축만 사용 | 압축 및 추가 할당 |
|---|---|---|
| 전력소모율 | 110.9% | 86.7% |

〈표 5〉 프로그램 수행 성능 비교

| 기존 방법 대비 | 압축만 사용 | 압축 및 추가할당 |
|---|---|---|
| 수행 사이클 비 | 119% | 99% |

## 5. 결 론

모바일 임베디드 시스템의 사용이 확대되고 있다. 많은 경우에 모바일 시스템은 일차 또는 이차전지를 이용하게 되기에 특히 전력 소모에 대해서 예민하며, 시스템 성능 손실을 최소로 줄이면서 전력 소모를 줄이는 것이 활발하게 연구되고 있다.

본 논문은 이러한 모바일 임베디드 시스템에서 프로세서 내부의 스크래치 패드 메모리에 저장되는 데이터를 압축하여 외부 메모리 접근을 줄임으로써 수행 성능의 손실을 최소화하고 전력 소모를 줄일 수 있는 방법을 제시하였다.

실제 모바일 스트리밍을 위하여 설계된 프로세서에서 이와 같은 기법을 적용하여 본 결과 기존의 시스템에 비해서 동등한 수행 사이클을 나타내면서, 디지털 부분의 전력 소모를 13.3% 감소시킬 수 있었다. 이와 같은 결과에 기반 하여, 차후 연구 단계로 지역적 참조 특성에 따라서 메모리 배치를 자동화하여 전력 소모 개선을 얻을 수 있는 개발 툴을 고안하는 과정이 진행 중이다.

## 참 고 문 헌


[1] M. Ekman, P. Stenstrom, “A Robust Main-memory Compression Scheme,” Int'l Symp. Computer Architecture, pp.74-85, 2005.

[2] Y. Yang, R. Gupta, “Frequent-Value Locality and its Applications”, ACM Trans. on Embedded Computing Systems, Vol.1, pp.79-105, 2002.

[3] W. Wolf, Computers as Components: Principles of Embedded Computing System Design, Morgan Kaufmann Publishers Inc., 2001.

[4] M. Farrens, A. Park, “Dynamic Base Register Caching: A technique for Reducing Address Bus Width”, Int'l Symp. Computer Architecture, 1991.

[5] G. Keramidas, K. Aisopos, S. Kaxiras, “Dynamic Dictionary-Based Data Compression for Level-1 Caches”, Lecture Notes in Computer Science, Vol3894/ 2006, pp.114-129, 2006.

[6] F. Douglis, “The Compression Cache: Using On-line Compression to Extend Physical memory” USENIX Conf., pp.519-529, 1993.

[7] M. Kjelso, M. Gooch, S. Jones. “Performance Evaluation of Computer Architectures with Main Memory Data Compression”, Journal of Systems Architecture. Vol.45, pp.571-590, 1999.

[8] P. Wilson, S. Kaplan, Y. Smaragdakis, “The Case for Compressed Caching in Virtual Memory Systems”, USENIX Ann. Technical Conf., pp.101-116, 1999.

[9] R. S. De Castro, A. P. Do Lago, D. da Silva, “Adaptive Compressed Caching: Design and Implementation” Symp. Computer Architecture and High-Performance Computing, pp.10-18, 2003.

[10] B. Abali, H. Franke, X. Shen, D. Poff, B. Smith, “Performance of Hardware Compressed Main Memory”, Int'l Symp. High-Performance Computer Architecture, pp.73-81, 2001.

[11] R.B. Tremaine, P.A. Franaszek, J.T. Robinson, C.O. Schulz, T. B. Smith, M. E. Wazlowski, P.M. Bland, “IBM Memory Expansion Technology (MXT)”, IBM Journal Research & Development, Vol.45 No.2, pp.271-285, 2001.

[12] J.-S. Lee, W.-K. Hong, S.-D. Kim, “Design and Evaluation of a Selective Compressed Memory System,” Int'l Conf. Computer Design, pp.184-191, 1999.

[13] Y. Zhang, J. Yang, and R. Gupta, “Frequent Value Locality and Value-centric Data Cache design”, Int'l Conf' Architectural Support for Programming Languages and Operating Systems, pp.150-159, 2000.

[14] A.R. Alameldeen, D.A. Wood, “Frequent Pattern Compression: a Significance-based Compression Scheme for L2 Caches”, Technical Report 1500, Computer Sciences Dept,, Univ, Wisconsin-Madison, 2004.

[15] M. Kjelso, M. Gooch, S. Jones, “Design and Performance of a Main Memory Hardware Data Compressor”, EUROMICRO Conf., pp.423-430, 1996.

[16] P. R. Panda, N. D. Dutt, A. Nicolau, “Efficient Utilization of Scratch-pad Memory in Embedded Processor Applications”, European Design and Test Conf., 1997.

[17] M. Kandemir, J. Ramanujam, M.J. Irwin, N. Vijaykrishnan, I. Kadayif, A. Parikh, “Dynamic Management of Scratch-pad Memory Space”, Design Automation Conf., pp.690-695, 2001.

[18] L. Wang, W. Tembe, S. Pande, “Optimizing On-chip Memory Usage Through Loop Restructuring for Embedded Processors”, Int.l Conf. Compiler Construction, pp.141-156, 2000.

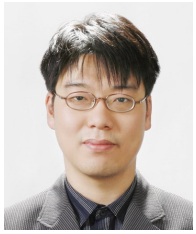
[19] M. Balakrishnan, P. Marwedel, L. Wehmeyer, N. Grunwald, R. Banakar, S. Steinke, “Reducing Energy Consumption by Dynamic Copying of Instructions onto Onchip Memory”, Int'l Symp. System Synthesis, pp.213-218, 2002.

[20] O. Ozturk, M. Kandemir, I. Demirkiran, G. Chen, M.J. Irwin, “Data Compression for Improving SPM Behavior”, Design Automation Conf. pp.401-406, 2004.

[21] GDM1202 Developer's Manual, GCT Semiconductor.

[22] Bluetooth Specification 1.2, Bluetooth SIG. Inc.

[23] T. A. Welch, “A Technique for High Performance Data Compression,” IEEE Computer, Vol.17, No.6, pages 8-19, 1984.


서 효 중

e-mail : hjsuh@catholic.ac.kr

1991년 서울대학교(학사)

1994년 서울대학교 컴퓨터공학과 (공학석사)

2000년 서울대학교 컴퓨터공학과 (공학박사)

2000년~2003년 지씨티 리서치 선임연구원

2003년~현재 서울대학교 컴퓨터연구소 객원연구원

2003년~현재 가톨릭대학교 컴퓨터정보공학부 조교수

관심분야 : 컴퓨터 구조, 컴퓨터 시스템, 내장형시스템