On-Chip SRAM을 이용한 임베디드 시스템 메모리 계층 최적화*

김정원 김승균 이재진 정창희† 우덕균†
서울대학교 컴퓨터공학부, †한국전자통신연구원 임베디드S/W연구단
{jungwon, seungkyun, jlee}@aces.snu.ac.kr, {chjung, dkwu}@etri.re.kr

Memory Hierachy Optimization in Embedded Systems Using On-Chip SRAM

Jungwon Kim Seungkyun Kim Jaejin Lee Changhee Jung† Duk-Kyun Woo†
School of Computer Science and Engineering, Seoul National University,
†Embedded Software Research Division, ETRI

1. 서론

데스크톱과 마찬가지로 임베디드 컴퓨터 시스템 환경에서 SRAM과 DRAM은 코드나 데이터를 저장하는 가장 보편적인 메모리다. SRAM은 빠르지만 가격이 비싼 반면, DRAM은 느리지만 가격이 저렴하다. 이 두 메모리의 장점을 동시에 활용하기 위해서는 크기가 큰 DRAM과 크기가 작은 SRAM을 동시에 갖추고 자주 접근하는 코드나 데이터를 SRAM에 할당하는 방법을 사용해야 한다. 컴퓨터 시스템에서 On-Chip SRAM을 사용하는 방법은 하드웨어가 관리하는 캐시(Cache) 메모리를 사용하는 방법과 하드웨어 도움 없이 소프트웨어가 직접 관리하는 스크래치패드(Scratchpad) 메모리를 사용하는 방법으로 나누어진다. 데스크톱에서는 캐시가 주로 사용되지만, 데스크톱 환경보다 자원이 제한적인 임베디드 시스템 환경에서는 스크래치패드 메모리의 단순한 내부 구조와 작은 크기, 지연시간의 예측 가능성 때문에 캐시보다 유리할 수 있다. 스크래치패드를 이용한 임베디드 시스템 최적화는 많은 연구가 되어왔다[1, 2]. 하지만 리눅스 기반의 임베디드 시스템에서의 스크래치패드 메모리에 관한 연구는 전무한 상황이다. 본 논문에서는 리눅스 기반 임베디드 시스템과 같은 메모리 관리 장치(Memory Management Unit, MMU)를 갖춘 임베디드 시스템에서의 메모리 계층, 특히 On-Chip SRAM인 스크래치패드 메모리를 이용하여 시스템 성능 향상과 동시에 에너지 소비 감소를 이루고자 한다.

2. 본론

On-Chip SRAM(이하 SRAM)을 이용한 임베디드 시스템 메모리 계층 최적화는 링킹 단계와 메모리 매핑 단계로 구성된다. 메모리 계층 최적화를 하기 위해 프로그램 개발자는 SRAM에 할당될 데이터나 코드를 선택한다. SRAM에 할당할 데이터와 코드 선택 후 링킹 단계를 거치면 선택한 데이터와 함수가 모두 sram 섹션으로 합쳐진 ELF 프로세스 이미지가 생성된다. 생성된 프로세스 이미지 실행 시 메모리 매핑 단계가 진행되는데, 이 단계에서 프로세스 이미지의 sram 섹션이 실제 SRAM에 매핑된다. 링킹 단계에서는 원하는 데이터나 함수를 sram 섹션으로 모으는 작업을 수행한다. 링킹 단계에서 가장 중요한 문제는 “sram 섹션을 ELF 이미지 어느 곳에 위치시키는가”이다. ELF 파일 포맷의 대표적인 섹션은

*본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장동력핵심기술개발사업[2006-S-040-01 Flash Memory 기반 임베디드 멀티미디어 소프트웨어 기술개발, 2006-S-038-02 모바일 컨버전스 컴퓨팅을 위한 단말적응형 임베디드 운영체제 기술]과 한국학술진흥재단의 BK21 사업의 일환으로 수행하였습니다. 또한 이 연구를 위해 연구장비를 지원하고 공간을 제공한 서울대학교 컴퓨터연구소에 감사드립니다.

bss, data, text가 있다. bss와 data 섹션은 데이터 세그먼트로 통합되고, text 섹션은 텍스트 세그먼트를 이룬다. 데이터 세그먼트는 읽기/쓰기/실행 가능 속성을 가지고 있는 반면, 텍스트 세그먼트는 읽기/실행 가능 속성을 지니고 있다. 본 논문에서 제안하는 최적화 기법은 데이터(읽기/쓰기)와 코드(읽기/실행)를 동시에 SRAM에 할당할 수 있어야하기 때문에 읽기/쓰기/실행 가능 속성을 가진 데이터 세그먼트에 sram 섹션을 위치시켜야 한다. 이제 데이터 세그먼트 어느 부분에 sram 섹션을 위치시켜야 하는가가 문제가 된다. sram 섹션이 위치 가능한 곳은 data 섹션 앞, bss 섹션 뒤, data섹션과 bss 섹션 사이이다. 우선 data 섹션 앞은 여러 섹션들의 의존 관계 때문 불가능하다. 또한 bss 섹션 뒤는 힙 영역과 충돌하기 때문에 역시 불가능하다. data 섹션과 bss 섹션 사이는 어떠한 의존 관계도 존재하지 않고, 실행 시 변동이 없기 때문에 sram 섹션이 위치할 수 있다. 링킹 단계 마지막으로 sram 섹션 시작 주소와 끝 주소에 각각 sram_start와 sram_end 라는 전역 변수를 정의한다. 메모리 매핑 단계는 링킹 단계에서 만들어진 ELF 이미지의 sram 섹션을 실행 시 SRAM에 적재하고 두 공간을 매핑하는 작업을 수행한다. 이 작업을 수행하는 것은 SRAM ALLOCATOR 모듈로, 타깃 어플리케이션과 같이 빌드된다. SRAM ALLOCATOR는 우선 프로그램이 시작하기 전에, ELF 이미지의 sram 섹션 전체를 힙 영역으로 덤프한다. sram 섹션의 시작과 끝 주소는 링킹 단계에서 정의된 sram_start와 sram_end 변수를 이용한다. 그리고 SRAM를 sram 섹션에 메모리 매핑시킨다. 끝으로 앞서 힙 영역에 덤프했던 기존의 sram 섹션 데이터를 다시 sram 섹션으로 복사하여 sram 섹션을 복구한다. 이제 메모리 매핑이 완료된 해당 프로세스가 자신의 sram 섹션에 접근하게 되면, 즉 sram 섹션에 있는 데이터를 읽고 쓰거나, sram 섹션에 있는 함수를 호출하게 되면 해당 프로세스에 투명하게 시스템의 SRAM에 할당된 데이터와 코드를 읽고 쓰고 실행하게 된다. 최적화가 적용된 프로세스는 적용되지 전과 동일하게 수행되지만, SRAM ALLOCATOR의 도움으로 프로세스가 모르게 SRAM 이용이 가능해지는 것이다. 실험은 총 9개 어플리케이션에 최적화 기법을 사용하지 않았을 경우와 최적화 기법을 사용하였을 때의 수행 시간과 에너지 소비를 측정하였다. 실험 결과, 수행 시간 측면에서는 최적화 기법을 사용하였을 때가 사용하지 않았을 때보다 평균 약 14%의 향상을 보였다. 에너지 측면에서는 최적화 기법을 사용한 결과가 그렇지 않은 결과에 비해 평균 약 15%의 에너지 소비 감소를 나타내었다. 특히 데이터 캐시 미스를 강제로 발생시키는 벤치마크의 경우에는 시스템 성능의 35% 향상과 동시에 40%의 에너지 소비의 감소를 보였다.

3. 결론

본 논문이 제안한 On-Chip SRAM을 이용한 임베디드 시스템 메모리 계층 최적화 기법은 프로그램 소스 상에서 코드나 데이터에 섹션 속성 추가와 SRAM ALLOCATOR와의 빌드만으로 시스템의 성능 향상과 에너지 소비의 감소를 이끌어 낼 수 있는 효과적인 방법이다. 또한 본 메모리 계층 최적화 기법은 처음으로 리눅스 기반의 임베디드 시스템에 적용한 스크래치패드 메모리 관리 기법으로 성능을 실측하여 그 효과를 보였다는 데에 그 의의가 있다.

참고 문헌

[1] Rajeshwari Banakar, Stefan Steinke, Bo-Sik Lee, M. Balakrishnan and Peter Marwedel. Scratchpad Memory: A Design Alternative for Cache Onchip memory in Embedded Systems. In CODES '02: Proceedings of the 10th International Workshop on Hardware/Software Codesign, pages 73-78, May 2002.

[2] Bernhard Egger, Jaejin Lee, and Heonshik Shin. Scratchpad memory management for portable systems with a memory management unit. In EMSOFT '06: Proceedings of the 6th ACM & IEEE International conference on Embedded software, pages 321-330, October 2006.