The Journal of the Convergence on Culture Technology (JCCT)
Vol. 9, No. 1, pp.771-776, January 31, 2023. pISSN 2384-0358, eISSN 2384-0366

http://dx.doi.org/10.17703/JCCT.2023.9.1.771
JCCT 2023-1-95

스크래치패드 메모리를 위한 데이터 관리 기법 리뷰

A Review of Data Management Techniques for Scratchpad Memory

조 두 산*

DOOSAN CHO*

요 약 스크래치패드 메모리는 소프트웨어 제어 온칩 메모리로서 기존의 캐시 메모리의 단점을 완화할 수 있게 설계되어 이용되고 있다. 기존의 캐시 메모리는 태그 관련 하드웨어 제어 로직이 있어 캐시 미스를 사용자가 직접 제어할 수 없으며, 사이즈가 크고 에너지 소모량이 상대적으로 많다. 스크래치패드 메모리는 이러한 하드웨어 오버헤드를 제거하였기 때문에 사이즈, 에너지 소모량에서 장점이 있으나 데이터 관리를 소프트웨어가 해야하는 부담이 존재한다. 본 연구에서는 스크래치패드 메모리의 데이터 관리 기법들을 분류하여 살펴보고 그 장점을 극대화할 수 있는 방안에 대하여 논의하였다.

주요어 : 스크래치패드 메모리, 데이터 매니지먼트, 성능, 최적화, 데이터 할당, 컴파일러

Abstract Scratchpad memory is a software-controlled on-chip memory designed and used to mitigate the disadvantages of existing cache memories. Existing cache memories have TAG-related hardware control logic, so users cannot directly control cache misses, and their sizes are large and energy consumption is relatively high. Scratchpad memory has advantages in terms of size and energy consumption because it eliminates such hardware overhead, but there is a burden on software to manage data. In this study, data management techniques of scratchpad memory were classified and examined, and ways to maximize the advantages were discussed.

Key words : scratchpad memory, data management, performance, optimization, data allocation, compiler

Ⅰ. 서 론

온칩 메모리의 일종인 스크래치 패드 메모리 (SPM, Scratch Pad Memory)는 지난 십여년의 시간동안 다양한 아키텍처 설계에 포함되었다. 이러한 이유는 최신의 전자기기가 대부분 휴대성을 강조하기 때문에 성능과 함께 배터리의 지속시간이 중요해짐에 따라 에너지 소비율이 중요한 설계 팩터로 자리매김함에 있다. 스크래치 패드 메모리는 기존의 캐시 메모리와 비교하였을 때 그 크기가 50% 작은 반면 접근 지연 시간은 더 짧다. 따라서 소비 전력이 기존의 캐시와 비교하여 더 작으며 더 탁월한 성능을 제공함으로 인기있는 설계 요소가 되었다. 하지만 데이터를 관리하는 하드웨어 제어 컴포넌트가 없기 때문에 컴파일러에 의하여 데이터가 관리되어야 하는 부담이 존재한다. 스크래치 패드 메모리는

데이터를 컴파일러와 같은 소프트웨어로 관리하기 때문에 매유 정교한 소프트웨어 기법이 요구된다.

본 연구에서는 현재까지 발표된 스크래치 패드 메모

*정회원, 순천대학교 전자공학과 교수 (제1저자)
접수일: 2022년 12월 30일, 수정완료일: 2023년 1월 8일
게재확정일: 2023년 1월 13일

Received: December 30, 2022 / Revised: January 8, 2023
Accepted: January 13, 2023
*Corresponding Author: dscho@scnu.ac.kr
Dept. of EE, Sunchon National Univ, Korea

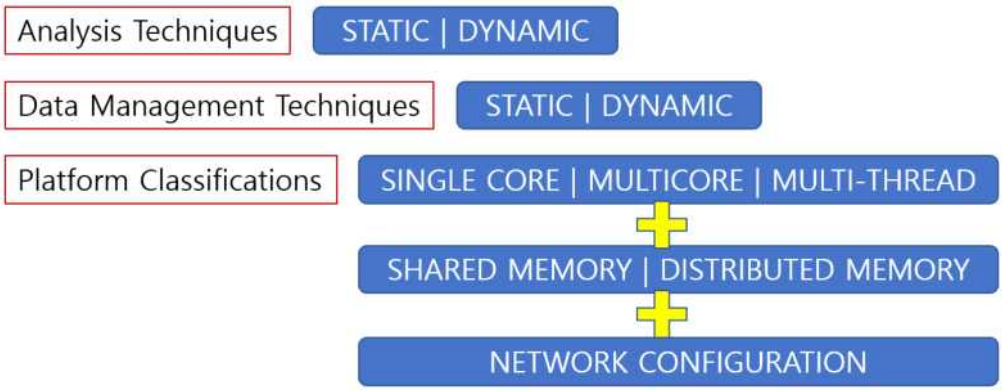

그림 1. 데이터 관리 기법의 분류
Figure 1. Classification for Data Management Techniques

리를 위한 다양한 데이터 관리기법을 리뷰하기로 한다. 그림 1은 데이터 관리를 위한 다양한 기법들을 분류하여 보이고 있다. 먼저 분석 기법은 정적/동적 기법으로 분류된다. 정적 기법은 컴파일러를 이용하여 프로그램 코드를 분석하는 것을 말한다. 컴파일러에는 제어 흐름 분석 및 데이터 흐름 분석을 위한 다양한 기법들이 구현되어 있어 데이터 관리를 위한 정적 분석을 효과적으로 수행할 수 있다.

동적 분석은 통상 프로파일링을 말한다. 프로파일러란 프로그램을 수행하여 실행중에 수집된 정보를 잘 정리하여 제공하는 소프트웨어들을 말한다. 프로파일러를 사용하면 프로그램을 정적 분석하는 대신 프로그램을 실행하여 전역 변수, 로컬변수, 정적 변수 등 메모리의 스택, 힙, 글로벌 영역에 저장된 데이터의 사용 패턴을 수집하여 제공할 수 있게 된다. 이러한 정보를 사용하면 데이터 관리가 효율적으로 수행될 수 있다.

데이터 관리 기법은 정적, 동적 분석으로 분류 될 수 있다. 프로그램에서 사용되는 데이터는 컴파일 시간에 데이터의 위치를 미리 지정하여 사용될 수 있다. 이러한 기법을 정적 기법이라 한다. 데이터는 프로그램 실행중에 이동하면서 그 위치가 정해질 수 있다. 이러한 기법을 동적 데이터 할당 기법이라 한다. 그리고 스트래치 패드를 위한 데이터 관리 기법들은 시스템 아키텍처의 구조에 따라 다시 분류될 수 있다. 단일 코어, 멀티코어, 멀티쓰레드를 위한 데이터 관리 기법들은 로컬 메모리와 원격 메모리 그리고 공유 메모리와 분산 메모리로 구성될 수 있다. 이렇게 플랫폼의 구조에 따라 데이터 관리 기법이 다르게 설계 분류된다. 그리고 온칩 네트워크의 구조에 따라 데이터 이동 지연시간이 다르게 구성될 수 있음으로 온칩 네트워크의 구성 또한 데이터 관리 기법을 분류하는 기준이 되었다.

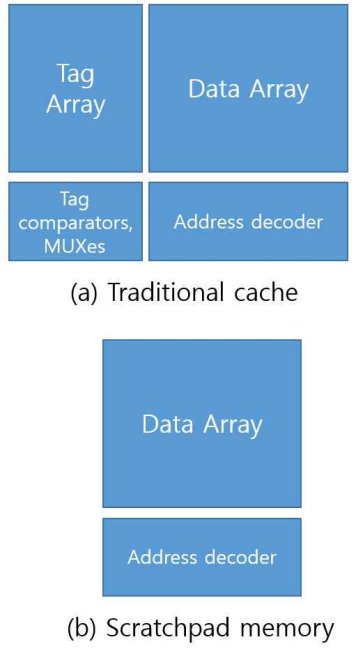

그림 2. 캐시와 SPM의 차이
Figure 2. Difference between cache and SPM

II. 데이터 관리 기법의 분류

스크래치 패드 메모리는 그림 2에 나타난 바와 같

이 전통적인 캐시 메모리와 비교하여 그 사이즈가 약50%정도 작으며, 따라서 에너지 소모량도 약 50% 정도 작게 소모한다. 왜냐하면 스크래치 패드 메모리는 Tag array, Tag comparators, MUXes가 없기 때문이다. 그렇기 때문에 스크래치 패드 메모리는 다이렉트 액세스 메모리 엔진 (DMA, Direct Access Memory Engine)으로 데이터가 메인 메모리와 스크래치 패드 메모리 사이를 이동하게 된다.

이렇게 DMA 명령어가 메인 메모리와 스크래치 패드 메모리 사이를 이동하기 위하여 프로그램에 코드로 적정한 위치에 생성되어야 한다. 이때 적정한 위치란 해당 데이터가 사용하기 전까지 정해진 스크래치 패드 메모리 주소상에 저장되어야 한다는 의미이다. 이렇게 데이터가 적절하게 이동하기 위해서는 크게 두 가지 문제를 컴파일러가 해결해야 한다. 첫째, 프로그램이 사용하는 데이터 중에서 스크래치 패드로 이동할 데이터를 분류해야 한다. 두 번째는 이동할 데이터를 분류하고 나서 제한된 용량의 스크래치 패드 메모리를 효율적으로 사용할 수 있도록 하는 데이터 배치를 찾아야 한다. 마지막으로 데이터 배치까지 정해지면 필요한 명령어가 사용하기 전까지 데이터 이동이 완료될 수 있도록 DMA명령이 생성되면 된다.

첫째 문제는 프로파일링을 이용하여 프로그램에서 많이 사용되는 데이터들을 분류하는 것으로 정해진다[1, 2, 3]. 프로파일링은 프로그램에서 사용되는 데이터의 읽기 횟수와 쓰기 횟수를 프로그램의 함수 단위로 저장해주는 기법이다. 각 변수들의 읽기 쓰기 횟수를 저장하면 그 다음은 많이 사용되는 변수들 순서대로 스크래치패드 메모리 공간에 할당하는 문제를 해결해야 한다. 이때 메모리 공간 크기가 한정되어 있기 때문에 할당 가능한 크기의 변수들을 할당하며, 정적 할당과 동적 할당에 따라 데이터의 배치가 다르게 결정된다. 데이터의 정적할당[4-9]은 컴파일 시간에 메모리 공간에 할당될 데이터의 배치를 결정하는 것으로 프로그램 실행시간에 데이터의 배치가 변하지 않는다. 동적할당[10-21]은 정적분석과 프로파일링 정보를 이용하여 프로그램 실행시간에 데이터 배치가 변하도록 데이터를 할당하는 기법을 말한다.

III. 정적 및 동적 데이터 할당 기법

Verma [8]은 대표적인 정적 할당 방식 기법을 소개하고 있다. 기본적으로 스크래치 패드 메모리에 할당할 대상인 메모리 오브젝트로 프로그램 메모리 오브젝트 (PMO)와 데이터 메모리 오브젝트 (DMO)를 정의한다. PMO는 함수와 베이직 블록이고 DMO는 변수들을 의미한다. 여기서 변수는 배열 변수를 포함하는데 배열 전체를 할당 대상으로 하기에는 그 사이즈가 크기 때문에 여기서 배열 분리 기법을 소개하고 있다. ILP (Integer Linear Programming) 기법을 사용하여 에너지 소비량을 기준으로 에너지를 최소화하는 배열의 분리 지점을 찾는다. 배열을 분리하여 스크래치 패드 메모리 할당 대상으로 할당 알고리즘을 실행한다.

Verma [17]은 대표적인 동적할당 연구로 데이터 할당문제를 ILP로 공식화하여 할당 알고리즘을 구성하였다. 먼저 스크래치패드 메모리 공간을 시간축을 고려한 3차원 공간으로 고려하고, 데이터들의 Liveness 분석을 이용하여 각 변수들의 사용 흐름을 분석한다. 많이 사용되는 변수를 스크래치패드 메모리에 할당하고, 그 사용이 끝났을 때 다른 변수를 해당 공간에 할당하면 공간 사용 효율이 높아진다. 보통의 경우 정적할당보다 동적할당이 메모리 공간 사용 효율이 높아진다.

Udayakumaran [10]은 컴파일러에 의하여 결정되는 동적할당 방식에 대한 알고리즘을 소개하고 있다. 데이터가 스크래치패드 메모리에 할당되었을 때 얻는 이득을 계산하고 데이터 할당을 한다. 이때 데이터 사용이 끝난 데이터 집합은 스크래치패드 메모리에서 SWAP-OUT되어야 하기 때문에 사용이 끝난 데이터를 SWAP-OUT-SET에 저장한다. 데이터 SWAP-IN 그리고 SWAP-OUT 명령어 생성을 위하여 제안된 알고리즘은 프로시저 단위로 인-아웃을 수행한다. 이때 데이터 전송 지연 시간을 고려하여 데이터가 필요한 시점에 맞추어 데이터가 이동될 수 있도록 하는 시점에 명령어가 생성이 된다.

Dominguez [11]는 먼저 기존의 스크래치패드 메모리의 데이터 할당 기법들이 전역 변수 및 스택 변수들을 대상으로 컴파일러 기법들을 소개한 부분을 지적하고, 힙 데이터는 DRAM에 의존하여 성능이 저하되는 문제를 지적하였다. 따라서 힙 데이터에 대한 메모리 할당 기법이 요구되었다. 하지만 힙 데이터는 컴파일

시간에 그 크기를 분석할 수 없기 때문에 이를 메모리 할당 대상으로 기술을 개발하기에 어려움이 있었다. 본 연구에서는 프로그램을 프로시저 영역 (region)으로 분리하고, 영역 안에서 사용하는 데이터들을 바이트 단위로 사용 횟수를 프로파일링하여 이러한 데이터를 스크래치패드 메모리에 할당할 때 얻는 이득을 계산할 수 있게 하였다. 이때 동적할당이 되는 힙 데이터는 그대로 할당하기 어렵기 때문에 스택 데이터로 변경하면 deallocate하는 오버헤드를 줄일 수 있다. 스크래치패드 메모리를 여러개의 BIN으로 정의하여 BIN PACKING 문제의 솔루션으로 데이터 할당을 진행한다. 본 알고리즘은 기본적으로 많이 사용되는 데이터를 바이트단위로 할당하여 온칩메모리 사용율을 개선하고 시스템 성능 및 에너지 사용을 절감한다.

Lian Li [12]는 온칩 메모리 할당문제를 그래프 컬러링 알고리즘으로 해결하고 있다. 우선 스크래치패드 메모리 공간을 컬러링하기 위해서는 메모리 공간을 균등한 공간으로 분리해야 한다. 프로파일링을 사용하여 애플리케이션에서 많이 사용되는 변수의 크기들을 분류한다. 많이 사용된 순서대로 그 비중에 맞추어 스크래치패드 메모리 공간을 분할한다. 이렇게 분할된 공간을 많이 사용된 데이터로 컬러링하는 방식으로 데이터 할당 문제가 해결된다. 그래프컬러링으로 온칩 메모리 공간을 동적으로 이용할 수 있게 된다.

Francesco [15]는 하드웨어 소프트웨어 통합 방식으로 스크래치패드 메모리를 동적 관리하는 기법을 소개한다. 기본적으로 스크래치패드에 데이터를 할당하는 애플리케이션 프로그래머 인터페이스 SMmalloc를 제공하여 데이터를 개발자가 소프트웨어 명시적으로 할당할 수 있다. 할당이 결정된 데이터는 DMA를 사용하는 DMAJob, DMAM2add, DMAnewstate 애플리케이션 프로그래머 인터페이스들을 사용하여 메인메모리에서 스크래치 패드 메모리에 이동시킬 수 있다. 또한 스크래치패드 메모리에 데이터를 할당하는 문제를 knapsack 문제로 매핑하여 데이터 할당을 해결하였다. 이때 knapsack 문제는 greedy 휴리스틱을 사용하여 해결하였다. 제안된 기법은 행렬 연산과 이미지 필터에 적용하여 캐시 메모리 대비 실행시간 개선을 보여주었다.

Udayakumaran [16]은 프로그램에서 사용하는 글로벌 그리고 스택 데이터 오브젝트들을 대상으로 스크래치패드 메모리를 효율적으로 이용하는 컴파일러 기법에 대하여 소개하고 있다. 기존의 기법들과의 차이점은 소트프웨어 태그를 사용하지 않으며, 메모리 액세스 시간 예측이 가능하고, 오버헤드가 적으면서, 실시간 검사가 필요없다는 점이다. 프로그램을 영역(region)으로 나누어 각 영역에서 할당 데이터를 관리한다. 영역 안에서 타임스탬프 정보를 추출하여 관리하면 데이터를 동적으로 할당하여 메모리 공간 사용 효율을 높일 수 있다.

Manish Verma [17]은 동적 데이터 할당 문제의 솔루션을 글로벌 레지스터 할당문제에서 찾았다. 데이터는 글로벌 스칼라와 코드 세그먼트를 대상으로 하며, 이러한 데이터들의 Liveness 분석을 이용하여 스크래치패드 메모리에 상주 시간을 얻게 된다. 각 데이터의 상주 시간을 알게 되면 동적으로 스크래치패드 메모리 공간을 사용할 수 있게 된다. 제한된 메모리 공간을 효율적으로 사용하기 위하여 데이터의 로딩 시간과 스필아웃 시간을 계산하여 할당된 데이터를 위한 Load/Store 명령어를 생성하게 된다. 이 연구는 기존의 슼크래치패드 메모리 관련 연구들 중에서 가장 실질적으로 이용가능한 할당 기법을 제시하고 있다. 다만, 동적 할당의 단점인 메모리 단편화 (fragmentation) 문제에 대한 해법이 없기 때문에 기존의 기법에 단편화 해결방안을 추가한다면 가장 상용화에 가까운 기법으로 손색이 없다.

Arun Kannan [19]의 연구는 스크래치패드 메모리 공간을 스택으로 사용하도록 scratchpad memory manager를 제안하고 있다. 이 기법은 기존의 기법과 다르게 프로파일링을 사용하지 않으며 하드웨어 변경이 필요 없다. 스크래치패드 메모리 공간을 스택으로 사용하여 로컬 변수들을 할당하고 함수코드들을 할당하여 공간 사용을 효율적으로 만들 수 있는 기법을 제안하였다.

Ke Bai [20, 21]은 힙데이터를 스크래치패드 메모리에 할당하여 관리하는 기법을 제안하였다. 멀티코어 환경에서 제한된 로컬 메모리의 관리를 동적 메모리 할당으로 한다. 기본적으로 동적 메모리 할당은 글로벌 메모리 주소에 위치하게 된다. 프로그래머는 명시적으로 로컬 스크래치패드 메모리에 동적 메모리 할당을 사용할 수 있다. 이때 제한된 용량의 관리를 위하여 테이블을 사용한다. 이 테이블을 사용하여 할당된 공간을 반환할 수 있으며, 만약 공간이 가득찬 상태에서 할당

이 호출되면 가장 오래된 데이터를 글로벌 메모리로 스필하게 된다. 이렇게 힙데이터를 사용하도록 스크래치패드 메모리 공간을 효율적으로 이용하는 기법을 제안하였다.

Suhendra [22]은 WCET를 최소화하는 스크래치패드 메모리 관리기법을 제안하고 있다. 이것은 제어흐름 그래프를 구성하고, WCET를 결정하는 패스를 중심으로 데이터를 스크래치패드 메모리 공간에 할당한다. 결과적으로 WCET를 최소화하게 된다. 데이터 할당 방식으로 greedy 휴리스틱과 branch and bound를 제안하였다. 기존의 대부분의 기법이 액세스 횟수가 많은 데이터를 중심으로 데이터 관리를 하였으나 WCET를 고려하여 메모리를 관리한다는 점에서 기존 연구 결과와의 차별성이 있었다.

Ⅳ. 결 론

스크래치패드 메모리는 소프트웨어 제어 온칩 메모리로서 기존의 캐시 메모리의 단점을 완화할 수 있게 설계되어 이용되고 있다. 기존의 캐시 메모리는 TAG 관련 하드웨어 제어 로직이 있어 캐시 미스를 사용자가 직접 제어할 수 없으며, 사이즈가 크고 에너지 소모량이 상대적으로 많다. 스크래치패드 메모리는 이러한 하드웨어 오버헤드를 제거하였기 때문에 사이즈, 에너지 소모량에서 장점이 있으나 데이터 관리를 소프트웨어가 해야하는 부담이 존재한다. 본 연구에서는 스크래치패드 메모리의 데이터 관리 기법들을 분류하여 살펴보고 그 장점을 극대화할 수 있는 방안에 대하여 논의하였다. 스크래치패드 메모리를 위한 데이터 할당 기법은 제한된 공간을 효율적으로 이용하기 위한 기법과 동시에 메모리 단편화 문제를 해결하여야 한다. 두 가지 문제를 동시에 해결한 기법은 아직 제안되지 않았기 때문에 실제의 응용 프로그램에 적용 가능한 기법의 개발이 요구된다.

References

[1] Avissar, Barua, Stewart, “An Optimal Memory Allocation Scheme for Scratch Pad Based Embedded Systems,” ACM Trans. on Embedded Computing Systems (TECS), Vol. 1, No. 1, pp. 6-26, (2002). https://doi.org/10.1145/581888.581891

[2] Baiocchi, Childers, Davidson, Hiser, Misurda, “Fragment cache management for dynamic binary translators in embedded systems with scratchpad” Proc. of the 2007 Int. Conf. on Compilers, Architecture, and Synthesis for Embedded Systems (2007). https://doi.org/10.1145/1289881.1289898

[3] Rajeshwari Banakar, Stefan Steinke, Bo-Sik Lee, M. Balakrishnan, and Peter Marwedel, “Scratchpad memory: design alternative for cache on-chip memory in embedded systems,” In Proceedings of the tenth international symposium on Hardware/software codesign, 73 - 78, (2005). https://doi.org/10.1145/774789.774805

[4] Preeti Ranjan Panda, Nikil D. Dutt, and Alexandru Nicolau. “Efficient Utilization of Scratch-Pad Memory in Embedded Processor Applications,” In Proceedings of the European conference on Design and Test, (1997). https://doi.org/10.5555/787260.787762

[5] Sjodin, von Platen, “Storage allocation for embedded processors,” proceedings of the international conference on compilers, architecture, and synthesis for embedded systems, (2001).http://doi.org/10.1145/502217.502221

[6] Oren Avissar, Rajeev Barua, and Dave Stewart. “Heterogeneous memory management for embedded systems,” In Proceedings of the international conference on Compilers, architecture, and synthesis for embedded systems, (2001).

[7] Nghi Nguyen, Angel Dominguez, and Rajeev Barua. “Memory allocation for embedded systems with a compile-time-unknown scratch-pad size,” In Proceedings of the international conference on Compilers, architectures and synthesis for embedded systems, (2005).

[8] Manish Verma, Stefan Steinke, and Peter Marwedel. “Data partitioning for maximal scratchpad usage,” In Proceedings of the Asia and South Pacific Design Automation Conference, (2003).

[9] Stefan Steinke, Lars Wehmeyer, Bo-Sik Lee, Peter Marwedel, “Assigning Program and Data Objects to Scratchpad for Energy Reduction,” In Proceedings of Design, Automation & Test in Europe Conference & Exhibition, (2002).

[10]Sumesh Udayakumaran and Rajeev Barua. “Compiler-decided dynamic memory allocation for scratch-pad based embedded systems,” In

Proceedings of the international conference on Compilers, architecture and synthesis for embedded systems (2003).

[11]Angel Dominguez, Sumesh Udayakumaran, and Rajeev Barua, “Heap data allocation to scratch-pad memory in embedded systems,” J. Embedded Comput. 1, 4 (2005).

[12] Lian Li, Lin Gao, Jingling Xue, "Memory coloring: a compiler approach for scratchpad memory management," In proceedings of International Conference on Parallel Architectures and Compilation Techniques, (2005).

[13]M. Kandemir, J. Ramanujam, J. Irwin, N. Vijaykrishnan, I. Kadayif, and A. Parikh, “Dynamic management of scratch-pad memory space,” In Proceedings of the 38th annual Design Automation Conference (2001).

[14]Jingtong Hu, Chun Jason Xue, Qingfeng Zhuge, Wei-Che Tseng, Edwin H.-M. Sha, "Towards energy efficient hybrid on-chip Scratch Pad Memory with non-volatile memory," Design, Automation & Test in Europe (2011)

[15]Poletti Francesco, Paul Marchal, David Atienza, Luca Benini, Francky Catthoor, and Jose M. Mendias. “An integrated hardware/software approach for run-time scratchpad management,” In Proceedings of the 41st annual Design Automation Conference (2004).

[16]Sumesh Udayakumaran, Angel Dominguez, and Rajeev Barua, “Dynamic allocation for scratch-pad memory using compile-time decisions,” ACM Trans. Embed. Comput. Syst. 5, 2 (2006), 472 - 511. https://doi.org/10.1145/1151074.1151085

[17]Manish Verma and Peter Marwedel, “Overlay techniques for scratchpad memories in low power embedded processors,” IEEE Trans. Very Large Scale Integr. Syst. 14, 8 (2006), 802 - 815.https://doi.org/10.1109/TVLSI.2006.878469

[18]Bernhard Egger, Jaejin Lee, and Heonshik Shin, “Dynamic scratchpad memory management for code in portable systems with an MMU,” ACM Trans. Embed. Comput. Syst. 7, 2, Article 11 (2008). https://doi.org/10.1145/1331331.1331335

[19]Arun Kannan; Aviral Shrivastava; Amit Pabalkar; Jong-eun Lee, "A software solution for dynamic stack management on scratch pad memory," Asia and South Pacific Design Automation Conference (2009)https://doi.org/10.1109/ASPDAC.2009.4796548

[20]Ke Bai and Aviral Shrivastava, “Heap data management for limited local memory (LLM) multi-core processors,” In Proceedings of the eighth IEEE/ACM international conference on Hardware/software codesign and system synthesis (2010). https://doi.org/10.1145/1878961.1879015

[21]Ke Bai and Aviral Shrivastava, “Automatic and efficient heap data management for limited local memory multicore architectures,” In Proceedings of the Conference on Design, Automation and Test in Europe (2013).https://doi.org/10.7873/DATE.2013.130

[22]V. Suhendra, T. Mitra, A. Roychoudhury, Ting Chen, “WCET centric data allocation to scratchpad memory,” In the proceedings of the 26th IEEE International Real-Time Systems Symposium (2005). https://doi.org/10.1109/RTSS.2005.45

※ This work was supported by a Research promotion program of SCNU.