Check for updates

Journal of KIIT. Vol. 22, No. 2, pp. 63-69, Feb. 28, 2024. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2024.22.2.63

실시간 멀티코어 임베디드 시스템에서 메모리 간섭 축소를 위한 실행 지연 가능 태스크 실행 모델

장혁수*, 박시형**, 김형신***

Deferrable Task Execution Model for Reducing Memory Interference in a Real-Time Multi-Core Embedded System

Hyeoksoo Jang*, Sihyeong Park**, and Hyungshin Kim***

요 약

실시간 임베디드 시스템에 요구되는 기능이 증가함에 따라 멀티코어가 적용되고 있지만 공유 메모리로 인한 코어 간 간섭으로 태스크의 실행 시간이 증가할 수 있다. 이를 해결하기 위해 AER(Acquisition Execution Restitution) 등의 태스크 실행 모델이 제안되었다. 하지만 AER 실행 모델은 처리량을 감소시키거나 전체 실행 시간이 증가하는 문제가 발생할 수 있다. 본 논문에서는 AER을 바탕으로 캐시를 활용해 동시에 실행할 수 있는 태스크의 작업을 지연시켜 여러 코어에서 동시에 실행하는 모델을 제안한다. 실험을 통해 제안한 실행 모델이 코어 간 메모리 공유로 인한 간섭을 줄이고 실행 시간을 최대 35% 감소시킬 수 있음을 보였다.

Abstract

As the requirements for real-time embedded systems continue to grow, multi-cores are being adopted However, due to interference between cores due to shared memory, the execution time of the task may increase. To solve this problem, task execution model such as Acquisition Execution Restitution(AER) has been proposed. However the proposed model may reduce the throughput or increase the overall execution time. In this paper, based on AER, we propose a task execution model that utilizes cache to delay the execution of tasks that can be executed simultaneously. Experiments showed that the proposed execution model reduces interference due to memory sharing between cores and reduced execution time by up to 35%.

Keywords

real-time systems, multi-core systems, embedded systems, task execution model, memory interference

* 충남대학교 컴퓨터공학과 박사과정
- ORCID: https://orcid.org/0009-0007-9761-5616
** 한국전자기술연구원 선임연구원
- ORCID: https://orcid.org/0000-0001-8244-4817
*** 충남대학교 컴퓨터공학과 교수(교신저자)
- ORCID: https://orcid.org/0000-0001-9615-1644

· Received: Jan. 11, 2024, Revised: Feb. 01, 2024, Accepted: Feb. 04, 2024

· Corresponding Author: Hyungshin Kim
Dept. of Computer Science and Engineering, Chungnam National University
Daejeon, Korea
Tel: +82-42-821-5446, Email: hyungshin@cnu.ac.kr

Ⅰ. 서 론

항공기, 의료 장비, 자동차 등과 같은 안전 필수(Safety critical) 혹은 미션 크리티컬(Mission critical) 시스템은 태스크가 요구하는 실행 시간 보장을 위해 실시간 운영체제(RTOS, Real-time Operating System)를 사용한다[1][2]. RTOS는 시스템의 태스크들이 정해진 마감시간(Deadline) 이내에 수행되도록 우선순위를 기반으로 스케줄링해서 예측성(Predictability)을 제공하는 것이 중요하다. 최근 이러한 시스템에 요구되는 기능이 증가함에 따라 기존의 싱글코어 기반 시스템에서 멀티코어 시스템으로 전환되고 있다[3]-[5]. 멀티코어 시스템은 다수의 태스크를 여러 코어에서 동시에 실행할 수 있어서 시스템의 처리량(Throughput)을 증가시킬 수 있다. 하지만 상용 멀티코어 임베디드 시스템은 다수의 코어가 메모리를 공유하므로 동시에 여러 코어에서 동일한 메모리에 접근하면 코어 간 간섭이 발생할 수 있다[6]. 이러한 간섭으로 인해 RTOS에서 태스크의 실행 시간이 증가해 마감시간 보장 불가와 같은 문제가 발생할 수 있다.

멀티코어에서 메모리 공유로 인한 코어 간 간섭을 줄이기 위해 시스템 수준과 스케줄링 수준의 연구들이 수행되었다. 시스템 수준 연구로는 태스크에서 동시에 메모리에 접근할 때 발생하는 간섭을 줄이기 위해 태스크에서 사용할 수 있는 메모리 대역폭이나 메모리 뱅크를 제어해서 간섭을 줄이는 방법들이 제안되었다[7]-[9]. 하지만 이러한 연구들은 메모리 접근을 제어하기 위한 추가적인 하드웨어나 운영체제의 수정이 필요한 단점이 있다. 스케줄링 수준으로 간섭을 줄이기 위해 TDMA(Time Division Multiple Access)[10][11], 3-Phase 태스크 모델 (혹은 AER(Acquisition, Execution, Restitution) 모델)[12][13] 등이 제안되었다. 이러한 태스크 모델은 태스크 스케줄링을 위해 추가적인 스케줄러 구현이나 운영체제 수정이 필요하거나, 멀티코어 시스템의 처리량 저하로 인한 전체 실행 시간 증가 등의 단점을 가진다.

본 논문에서는 추가적인 하드웨어 지원이나 운영체제의 수정 없이 상용 임베디드 멀티코어 실시간 시스템에서 메모리 공유로 인한 간섭을 줄일 방법을 제안한다. 이를 위해 기존에 제안된 AER 태스크 모델을 기반으로 간섭을 줄이면서 처리량을 증가시켜 시스템의 전체 실행 시간을 개선할 수 있는 d-AER(Deferrable AER)을 제안한다. d-AER은 AER과 동일하게 태스크의 작업을 메모리 읽기, 연산/실행, 메모리 쓰기의 3단계로 분리하지만, 메모리 쓰기 단계에서는 캐시 미스가 읽기 단계보다 적게 발생하는 점을 이용해 각 코어에서 실행되는 태스크의 메모리 쓰기 단계를 동시에 처리한다. 이를 통해 메모리 간섭을 줄이면서 시스템의 처리량을 높일 수 있다.

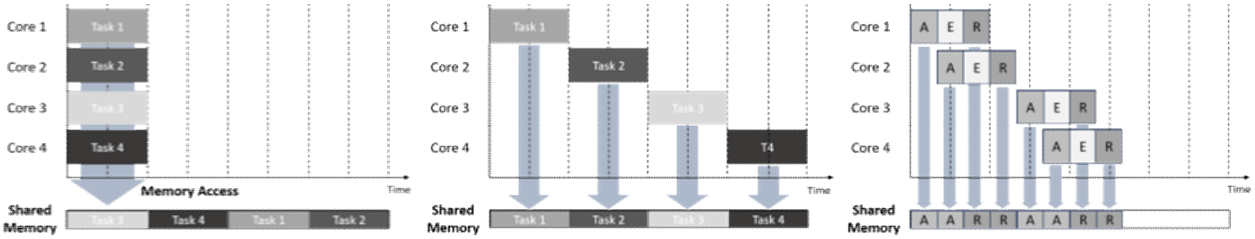
본 논문에서는 실험을 위해 ARM Cortex A9 기반 SABRE Lite (i.MX 6) 보드에서 상용 RTOS인 VxWorks 6.9를 사용해 메모리 공유로 인한 간섭을 분석하고 제안한 d-AER 태스크 실행 모델을 적용해 간섭 및 실행 시간 감소를 분석하였다. 이를 위해 메모리에 접근하는 데이터 크기에 따른 실행 시간을 측정하고 TDMA, AER, d-AER 적용에 따른 결과를 비교하였다. 제안한 d-AER 태스크 모델은 TDMA와 AER 보다 약 35%, 13% 감소한 실행 시간을 보였다.

본 논문의 구조는 다음과 같다. 2장에서는 태스크 실행 모델에 관련된 연구를 설명하고, 3장에서는 상용 임베디드 멀티코어 시스템에서 발생할 수 있는 메모리 공유로 인한 간섭 분석과 제안 태스크 실행 모델을 설명한다. 4장에서는 3장에서 제안한 태스크 실행 모델을 적용한 실험 결과와 분석을 다루며, 5장에서 결론을 맺는다.

Ⅱ. 관련 연구

2.1 시분할 다중 접근 모델(TDMA model)

멀티코어 시스템에서는 시스템의 처리량을 높이기 위해 그림 1의 (a)와 같이 다수의 코어에서 여러 태스크를 동시에 실행한다. 하지만 여러 태스크에서 동시에 메모리 읽기, 쓰기 작업을 하면 버스, 메모리 등의 하드웨어 자원 사용에 대한 간섭이 발생한다. 이러한 간섭은 각 태스크의 실행 시간 증가로 인한 예측성 저하와 같은 문제를 발생시킬 수 있으므로 RTOS에서는 간섭을 최대한 줄여야 한다.



(a) 병렬 실행 모델
(a) Parallel execution model

(b) 시분할 다중 접근 모델
(b) TDMA model

(c) 3-단계 모델
(c) 3-Phase model

그림 1. 태스크 실행 모델 비교
Fig. 1. Comparison of task execution models

멀티코어에서 동시에 메모리 작업을 요청하는 동작을 방지하기 위해서 TDMA 실행 모델이 제안되었다[10][11]. TDMA는 그림 1의 (b)와 같이 각 코어에서 실행되는 태스크들의 실행이 병렬적으로 수행되지 않고 한 번에 하나의 코어의 태스크만 실행한다. 이를 통해 각 코어의 태스크에서 메모리 접근이 발생하더라도 하나의 태스크만 실행되고 있으므로 코어 간 간섭이 발생하지 않는다. 하지만 하나의 코어의 태스크만 실행되므로 멀티코어 시스템의 처리량이 저하되는 문제가 발생한다.

2.2 3-Phase model

멀티코어에서 TDMA 실행 모델 적용에 따른 처리량 저하와 같은 문제를 해결하기 위해 3-Phase 실행 모델이 제안되었다[12][13]. 3-Phase 실행 모델은 태스크를 메모리 읽기(Am Acquisition), 실행/연산(E, Execution), 메모리 쓰기(R, Restitution)의 3단계로 구분해 실행한다.

A 단계에서는 메모리에 접근해 데이터를 가져오며, E 단계에서는 앞선 A 단계에서 가져온 데이터를 바탕으로 연산한다. 이후 R 단계에서 E 단계의 연산 결과를 메모리에 저장하는 작업을 수행한다. 이때, A와 R 단계에서는 캐시 미스로 인한 태스크 실행 시간 예측성 감소를 줄이기 위해 각 코어의 캐시(L1 캐시) 크기보다 작은 크기로 데이터 읽기, 쓰기 작업을 수행한다. 또한 A와 R 단계는 메모리에 접근하므로 다른 코어에서 실행되는 태스크의 A, R 단계와 동시에 실행되지 않는다. 반면 E 단계는 이미 가져온 데이터로 연산해서 다른 태스크의 A, R 단계와 동시에 수행될 수 있다. 이러한 부분을 통해 멀티코어에서 동시적인 메모리 접근을 없애 코어 간 간섭을 줄이고 TDMA 보다 시스템의 처리량을 높일 수 있다. 하지만 TDMA에 비해 각 코어의 캐시 크기보다 작은 단위로 메모리에 접근하므로 반복적인 실행으로 전체적인 실행 시간은 증가할 수 있지만 간섭을 줄여 RTOS의 스케줄링 예측성을 증가시킬 수 있다.

하지만 3-Phase 실행 모델로 태스크를 동작시키기 위해서는 기존의 태스크 코드를 A, E, R 단계에 맞춰 수정하거나, 별도의 스케줄러가 필요할 수 있다. 본 논문에서 제안한 지연 가능 AER 실행 모델은 기존의 3-Phase 실행 모델과는 다르게 각 태스크의 R 단계를 지연시켜 동시에 실행한다. 태스크를 3-Phase 모델로 실행하기 위해 태스크를 A, E, R 단계로 나눠서 구현하였고, 별도의 스케줄러 구현 없이 RTOS의 프로세스 간 통신 (IPC, Inter-Processor Communication) 중 이벤트를 사용해 각 단계의 실행을 제어하였다.

III. d-AER: 실행 지연 가능 AER

3.1 메모리 공유로 인한 간섭 분석

본 장에서는 본 논문에서 제안한 d-AER 실행 모델의 설명에 앞서 멀티코어에 발생할 수 있는 메모리 공유로 인한 코어 간 간섭에 대한 영향을 분석한다. 그림 2와 같이 ARM Cortex-A9 쿼드코어가 탑재된 SABRE Lite 보드에서 VxWorks 6.9 SMP(Symmetric Multiprocessing)를 사용해 실험을 수행하였다.

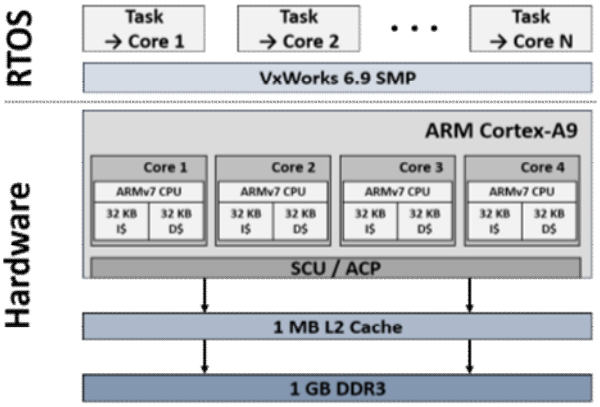


그림 2. 실험 환경
Fig. 2. Experiments environment

각 코어는 32KB의 인스트럭션 캐시와 32KB의 데이터 캐시를 가지며, 각 코어가 공유하는 1MB의 L2 캐시와 1GB의 메인 메모리가 존재한다.

메모리 공유로 인한 간섭 측정을 위해 그림 1의 (a)와 같이 VxWorks의 태스크를 코어별로 1개씩 생성해서 동시에 실행하였다. 각 태스크는 512KB 크기로 메모리 주소를 읽어서 값을 배열에 저장하는 작업을 수행한다. 메모리 동시 접근으로 인한 간섭을 확인하기 위해 동시에 실행되는 코어(태스크)의 개수를 1~4개로 변화시키면서 300회 반복해 태스크의 실행 시간(Cycle, 사이클)을 측정하였다.

그림 3은 실험 결과를 나타낸다. 1개의 코어만 사용하였을 때 평균적으로 1,464 사이클이 소요되었고 2개의 코어가 동시에 실행될 때는 약 1.1배 증가한 1,619 사이클이 소요되었다. 3개와 4개의 코어에서 동시에 태스크를 실행할 때 각각 약 1.4배 증가한 2,068 사이클, 2,065 사이클이 소요되었다.

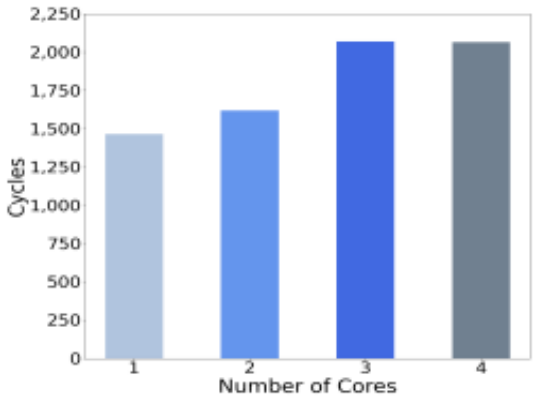


그림 3. 코어 수에 따른 메모리 접근 간섭 측정
Fig. 3. Memory access interference depending on the number of cores

실험을 통해 동시적인 메모리 접근으로 인해 간섭이 발생하여 실행 시간이 최대 1.4배까지 증가함을 확인하였다. 이에 따라 설정된 태스크의 마감 시간 내에 작업이 종료되지 못하는 문제가 발생할 수 있다.

3.2 실행 지연 가능 3-Phase 태스크 실행 모델

본 논문에서는 기존에 제안된 3-Phase 실행 모델을 바탕으로 태스크의 일부 작업을 지연시켜서 동시에 실행하는 d-AER 실행 모델을 제안한다. 기존 3-Phase 실행 모델은 태스크를 A, E, R 단계로 나누어 각 코어에서 실행될 때 메모리에 접근하는 A와 R 단계가 동시에 실행되지 않도록 태스크를 실행한다. 하지만 본 논문에서 제안하는 d-AER 실행 모델은 그림 4와 같이 각 태스크의 R 단계를 지연시켜 모든 코어의 E 단계가 끝나고 동시에 실행한다.

제안한 d-AER 실행 모델에서 R 단계를 동시에 실행하는 것은 메모리에서 데이터를 읽어오는 작업과 쓰는 과정에서 발생하는 캐시 미스가 똑같은 영향을 미치지 않기 때문이다. 메모리 읽기 작업에서 발생하는 캐시 미스는 프로세서의 스톨(Stall)을 발생시키지만, 메모리 쓰기 작업은 일반적으로 임계경로(Critical path)에 포함되지 않는다[14]. 따라서 이러한 점을 이용해 R 단계에서 발생하는 코어 간 간섭을 일부 수용해 시스템 전체의 처리량을 높여 실행 시간 감소를 목표로 한다. 하지만 본 논문에서 제안한 d-AER 실행 모델은 각각의 태스크 관점으로 보면 그림 4와 같이 코어 1~3의 경우 기존의 AER 실행 모델과 비교했을 때 태스크의 실행 시간이 증가할 수도 있는 한계점을 가진다.

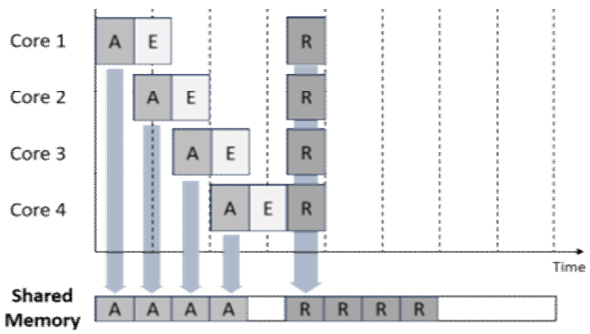


그림 4. d-AER 태스크 실행 모델
Fig. 4. d-AER task execution model

d-AER 실행 모델의 구현을 위해 그림 5와 같이 4개의 코어에 각 1개의 태스크가 할당되고, 태스크의 코드는 A, E, R 단계로 구성되어 있다. 코어별 각 코어의 태스크는 코어 1부터 4까지 차례대로 실행되며 동일한 우선순위를 가지도록 설정하였다. 태스크의 A 단계(acquisition() 함수)는 메모리에 L1 캐시 이하의 데이터를 읽어와 지역 변수(배열)에 저장하고, E 단계(execution() 함수)에서는 덧셈, 곱셈 등의 임의 연산으로 배열의 값을 수정하고 R 단계(restitution() 함수)는 연산된 값을 다시 특정 메모리 주소에 저장한다. 각 태스크의 A, E, R 단계의 실행 제어를 위해 VxWorks 6.9의 IPC 기능 중 메시지 함수 eventSend(), eventReceive()를 사용하였다.

코어 1의 태스크가 실행되기 전까지 다른 태스크는 eventReceive() 함수를 사용해 대기한다. 코어 1의 A 단계가 종료되면 코어 2에 eventSend() 함수를 통해 이벤트를 전송하고, 코어 2는 eventReceive() 함수가 이벤트를 받아서 A 단계를 실행한다. 이때 코어 1은 E 단계를 실행하고 eventReceive() 함수를 통해 대기를 한다. 이와 같은 과정을 코어 4의 E 단계 실행까지 반복하고, E 단계가 종료되면 모든 코어의 태스크에 eventSend() 함수를 사용해 이벤트를 전송한다. 이후 코어 1~4의 R 단계가 동시에 실행되고 종료 이후 다시 eventReceive() 함수를 사용해 실행을 대기한다.

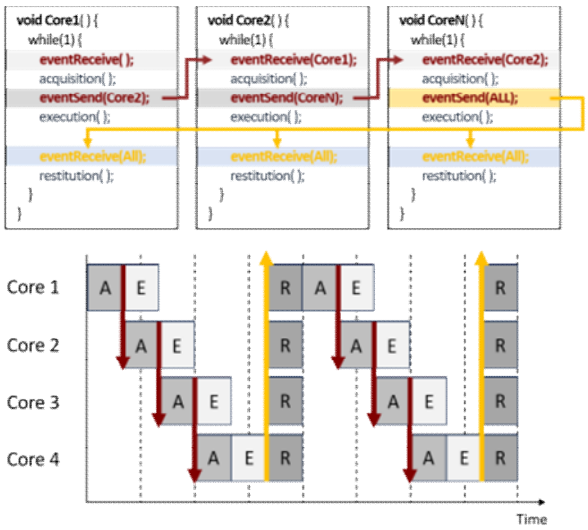


그림 5. d-AER 태스크 실행 모델 구현 방법
Fig. 5. Implementation of d-AER task execution model

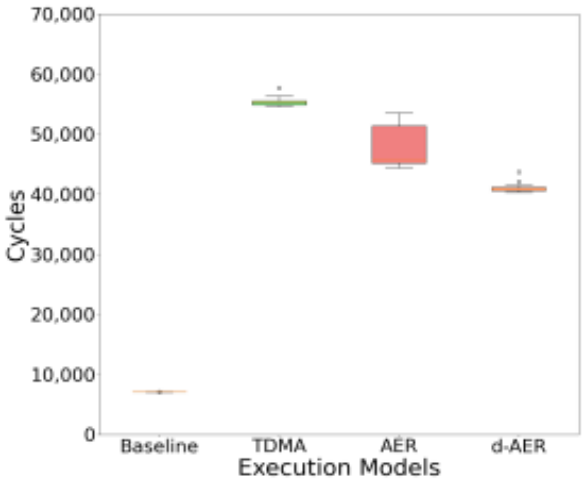


그림 6. 태스크 실행 모델별 실행 시간 비교
Fig. 6. Comparison of execution time by task execution models

IV. 실험 및 결과

본 논문에서 제안하는 d-AER 실행 모델의 성능 분석을 위해 그림 1의 실행 모델과 실행 시간 비교를 수행하였다. 실행 시간 측정은 코어 1의 태스크부터 코어 4의 태스크가 모두 종료될 때까지의 시간을 기준으로 하였다. 실행 시간은 ARM Cortex-A9의 PMU(Performance Monitor Unit)의 Cycle Count Register[15]를 사용해 각 태스크의 시작부터(A 단계) 종료 (R 단계)까지의 사이클을 읽어 소요된 사이클을 계산하였다. 각 실행 모델의 실행 시간은 15회 반복해 측정하였다.

그림 6은 각 실행 모델에 대한 실험 결과를 나타낸다. 그림 6의 Baseline은 그림 1의 (a)와 같이 동시에 여러 코어에서 태스크를 수행한 결과이다. 그림 6과 같이 각 실행 모델의 평균 실행 시간은 7,091 사이클, 55,407 사이클, 47,632 사이클, 41,119 사이클로 TDMA가 가장 긴 실행 시간을 보였다. 본 논문에서 제안한 d-AER 실행 모델은 TDMA와 AER 실행 모델보다 약 35%, 13% 감소한 실행 시간을 보였다. AER의 실행 시간의 분포가 큰 것은 본 논문에서 사용한 시스템에서 코드를 구현할 때 TDMA와 d-AER 보다 약 2배 이상의 이벤트 요청이 필요하였기 때문에 해당 부분의 오버헤드가 발생한 것으로 분석되었다.

따라서 본 논문에서 제안한 d-AER 실행 모델을 사용하면 기존 멀티코어에서 공유 메모리 동시 접근으로 인한 간섭을 줄이기 위해 제안된 태스크 실행 모델보다 전체 실행 시간을 개선할 수 있음을 보였다.

V. 결 론

본 논문에서는 상용 임베디드 멀티코어 기반 실시간 시스템에서 발생할 수 있는 메모리 공유로 인한 간섭을 분석하였다. 이러한 간섭으로 인해 태스크의 실행 시간이 증가해 태스크의 마감 시간을 보장하지 못하는 문제가 발생할 수 있다. 본 논문에서는 간섭으로 인한 문제를 해결하기 위해 제안된 태스크 실행 모델을 적용해 비교를 수행하였다. 특히 TDMA와 AER 태스크 실행 모델을 적용하였을 때 시스템의 처리량이 저하돼 실행 시간이 증가하는 문제를 해결하기 위해 태스크의 일부 작업을 지연시켜 동시에 실행하는 d-AER 태스크 실행 모델을 제안하였다. 본 논문에서 제안하는 d-AER 실행 모델을 적용하였을 때, 기존에 제안된 TDMA와 AER 보다 약 35%, 15% 감소한 실행 시간을 보였다.

본 논문에서는 임의의 작은 코드 수준의 태스크를 사용해 각 실행 모델의 실행 시간만 비교하였는데, 향후 연구로 벤치마크 혹은 실제 메모리 사용이 빈번하게 발생하는 응용프로그램을 사용해 제안한 d-AER 실행 모델의 성능을 분석할 계획이다.

References

[1] J. L. Herman, C. J. Kenna, M. S. Mollison, J. H. Anderson, and D. M. Johnson, "RTOS Support for Multicore Mixed-Criticality Systems", 2012 IEEE 18th Real Time and Embedded Technology and Applications Symposium, Beijing, China, pp. 197-208, Apr. 2012. https://doi.org/10.1109/RTAS.2012.24.

[2] S. Avramenko and M. Violante, "RTOS Solution for NoC-Based COTS MPSoC Usage in Mixed-Criticality Systems", Journal of Electronic Testing, Vol. 35, pp. 25-44, Feb. 2019. https://doi.org/10.1007/s10836-019-05779-y.

[3] S. Saidi, R. Ernst, S. Uhrig, H. Theiling, and B. D. Dinechin, "The shift to multicores in real-time and safety-critical systems", 2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS), Amsterdam, Netherlands, pp. 220-229, Oct. 2015. https://doi.org/10.1109/CODESISSS.2015.7331385.

[4] R. Medina, E. Borde, and L. Pautet, "Generalized Mixed-Criticality Static Scheduling for Periodic Directed Acyclic Graphs on Multi-Core Processors", IEEE Transactions on Computers, Vol. 70, No. 3, pp. 457-470, Apr. 2020. https://doi.org/10.1109/TC.2020.2990229.

[5] M. Bottaro and T. Vardanega, "Evaluating a multicore Mixed-Criticality System implementation against a temporal isolation kernel", Journal of Systems Architecture, Vol. 130, pp. 1-14, Sep. 2022. https://doi.org/10.1016/j.sysarc.2022.102688.

[6] T. Lugo, S. Lozano, J. Fernandez, and J. Carretero, "A Survey of Techniques for Reducing Interference in Real-Time Applications on Multicore Platforms", IEEE Access, Vol. 10, pp. 21853-21883, Feb. 2022. https://doi.org/10.1109/ACCESS.2022.3151891.

[7] H. Yun, R. Mancuso, Z. P. Wu, and R. Pellizzoni, "PALLOC: DRAM bank-aware memory allocator for performance isolation on multicore platforms", 2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS), Berlin, Germany, pp. 155-166, Apr. 2014. https://doi.org/10.1109/RTAS.2014.6925999.

[8] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, and L. Sha, "MemGuard: Memory bandwidth reservation system for efficient performance isolation in multi-core platforms", 2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS), Philadelphia, PA, USA, pp. 55-64, Apr. 2013. https://doi.org/10.1109/RTAS.2013.6531079.

[9] J. Yi, B. Dong, M. Dong, R. Tong, and H. Chen, "MT^2: Memory Bandwidth Regulation on Hybrid NVM/DRAM Platforms", 20th USENIX Conference on File and Storage Technologies (FAST 22), pp. 199-216, Feb. 2022. https://dblp.org/rec/conf/fast/YiD0T022.

[10] F. Boniol, H. Casse, E. Noulard, and C. Pagetti, "Deterministic Execution Model on COTS Hardware", International Conference on Architecture of Computing Systems, Munich, Germany, Vol. 7179, pp. 98-110, Feb. 2012. https://doi.org/10.1007/978-3-642-28293-5_9.

[11] G. Yao, R. Pellizzoni, S. Bak, E. Betti, and M. Caccamo, "Memory-centric scheduling for multicore hard real-time systems", Real-Time Systems, Vol. 48, pp. 681-715, Jun. 2012. https://doi.org/10.1007/s11241-012-9158-9.

[12] C. Maia, L. Nogueira, L. M. Pinho, and D. G. Perez, "A closer look into the AER Model", 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), Berlin, Germany, pp. 1-8, Sep. 2016. https://doi.org/10.1109/ETFA.2016.7733567.

[13] J. Arora, S. A. Rashid, C. Maia, and E. Tovar, "Analyzing Fixed Task Priority Based Memory Centric Scheduler for the 3-Phase Task Model", 2022 IEEE 28th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), Taipei, Taiwan, pp. 51-60, Aug. 2022. https://doi.org/10.1109/RTCSA55878.2022.00012.

[14] S. Khan, A. R. Alameldeen, C. Wilkerson, O. Mutluy, and D. A. Jimenezz, "Improving cache performance using read-write partitioning", 2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA), Orlando, FL, USA, pp. 452-463, Feb. 2014. https://doi.org/10.1109/HPCA.2014.6835954.

[15] Y. W. Kwon and J. Y. Choi, "Accurate Time Measurement Method for ARM-based Embedded Linux", Journal of KIIT. Vol. 18, No. 2, pp. 61-67, Feb. 2020. https://doi.org/10.14801/jkiit.2020.18.2.61.

저자소개

장 혁 수 (Hyeoksoo Jang)

2018년 2월 : 충남대학교 컴퓨터공학과(공학사)
2019년 12월 ~ 2023년 6월 : 국방과학연구소 기술원
2020년 2월 : 충남대학교 컴퓨터공학과(공학석사)
2021년 3월 ~ 현재 : 충남대학교 컴퓨터공학과 박사과정
관심분야 : 임베디드 시스템, 실시간 운영체제

박 시 형 (Sihyeong Park)

2014년 2월 : 충남대학교 컴퓨터공학과(공학사)
2016년 2월 : 충남대학교 컴퓨터공학과(공학석사)
2021년 2월 : 충남대학교 컴퓨터공학과(공학박사)
2021년 3월 ~ 현재 : 한국전자기술연구원 선임연구원
관심분야 : 임베디드 시스템, 실시간 운영체제, 펌웨어, 온디바이스-AI

김 형 신 (Hyungshin Kim)

1990년 : 한국과학기술원 전산학과(공학사)
1991년 : University of Surrey 위성통신학과(공학석사)
1992년 ~ 2001년 : 한국과학기술원 인공위성연구센터 선임연구원
2003년 : 한국과학기술원 전산학과(공학박사)
2003년 ~ 2004년 : Carnegie Mellon University 박사후연구원
2004년 ~ 현재 : 충남대학교 컴퓨터공학과 교수
관심분야 : 실시간 임베디드 SW, 임베디드 AI 컴퓨팅