MPSoC 기반 임베디드 시스템을 위한 마스터-슬레이브 구조의 멀티프로세서 운영체제

김하석○ 서민열 맹지찬 유민수
한양대학교 전자컴퓨터통신공학부
{hskim, myseo, jcmaeng, msryu}@rtcc.hanyang.ac.kr

A Master-Slave Multiprocessor Operating System for MPSoC-based Embedded Systems

Haseok Kim○ Minyeol Seo Jichan Maeng Minsoo Ryu
Department of Electronics & Computer Engineering, Hanyang University

1. 연구동기 및 목표

최근 임베디드 시스템에서도 고용량 및 고속의 데이터 처리가 요구되면서 멀티프로세서 시스템이 점차 일반화 되어 가고 있다. 이는 단일 프로세서가 다음과 같은 문제점을 가지고 있기 때문이다. MPSoC는 프로세서의 구성에 따라 동종의(homogeneous) 프로세서를 사용하는 플랫폼과 이종의(heterogeneous) 프로세서를 사용하는 플랫폼으로 구분할 수 있다. 전자와 같이 구성된 프로세서의 예는 ARM사의 MPCore[7]를 들 수 있다. 후자와 같이 이종의 프로세서로 구성된 시스템으로는 ARM 프로세서와 TI DSP로 구성된 TI의 OMAP이나, PowerPC 프로세서 코어와 8개의 Synergistic Processor로 구성된 STI(Sony/Toshiba/IBM)의 Cell[4]을 들 수 있다.

멀티프로세서 OS는 다음과 같이 세 가지로 나눌 수 있다. 첫째, 각 프로세서가 OS를 독립적으로 수행하는 방법; 둘째, 하나의 마스터 프로세서가 통상적인 OS를 수행하고 다수의 슬레이브 프로세서는 최소한의 기능만을 제공하는 경량의 OS를 수행하는 방법(마스터/슬레이브 OS); 셋째, 모든 프로세서가 동일한 OS를 공유하는 방법(SMP OS)으로로 구분할 수 있다[1]. 첫 번째의 경우에는 서로 다른 프로세서에서 수행되는 태스크들이 통신하기 위해서는 별도의 네트워크 프로그래밍이나 미들웨어의 구현이 필요하며, 또한 이로 인해 프로세서간의 통신에 요구되는 오버헤드가 아주 크다는 문제점이 있다. 반면에, 두 번째의 SMP 방식은 단일 프로세서와 동일한 프로그래밍 모델을 제공할 뿐만 아니라 우수한 성능을 제공함에 따라 현재까지 널리 사용되어 왔다.

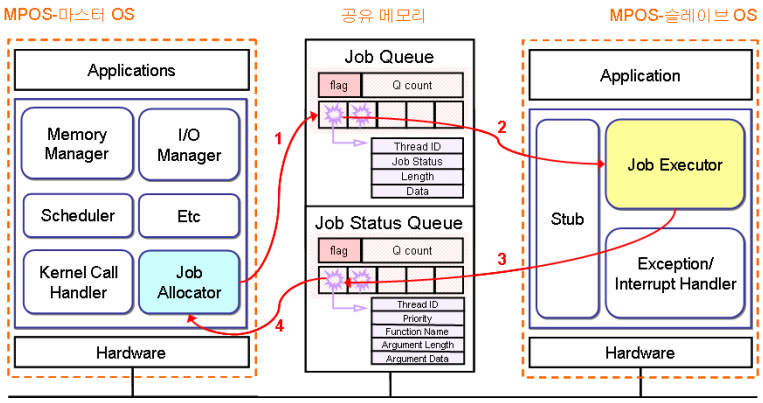
하지만, SMP 방식은 동종의 프로세서로 구성된 MPSoC에만 적용될 수 있으며 아울러 버스의 공유로 인해 프로세서의 개수에 제한이 있다는 문제점을 가진다. 본 연구에서는 이종의 프로세서로 구성된 MPSoC에 적용시킬 수 있으며 동시에 확장성 측면에서 우월한 마스터/슬레이브 형태의 멀티프로세서용 OS인 MPOS (Multi-Processor Operating System)를 개발하였다. 이를 위해 우선 마스터/슬레이브 구조에 적합한 커널의 내부 구조와 마스터와 슬레이브간의 효과적인 통신 방법을 제안하였으며, 이를 기반으로 MPSoC에 적합한 멀티프로세서 OS인 MPOS(Multi-Processor OS)를 개발하고 실험을 통해 그 유용성을 입증하였다.

2. MPOS의 구조 및 동작 방식

2.1. Job 할당

마스터 운영체제는 슬레이브 운영체제에 Job 할당을 요청할 수 있다. 마스터 운영체제의 Job Allocator는 Job을 할당하기 위해 해당 슬레이브의 Job Queue에 Job을 등록 한다. 슬레이브 운영체제의 Job Executor는 자신의 Job Queue를 검사하여, 요청된 Job이 있으면 Job Queue에 등록된 Job을 가져와 쓰레드를 생성하여 수행시키고, 마스터의 Job Status Queue에 job의 정보를 running 상태로 기록한다. Job이 완료 Job Executor는 Job Status Queue의 job 정보를 complete로 기록한 후, return data 저장한

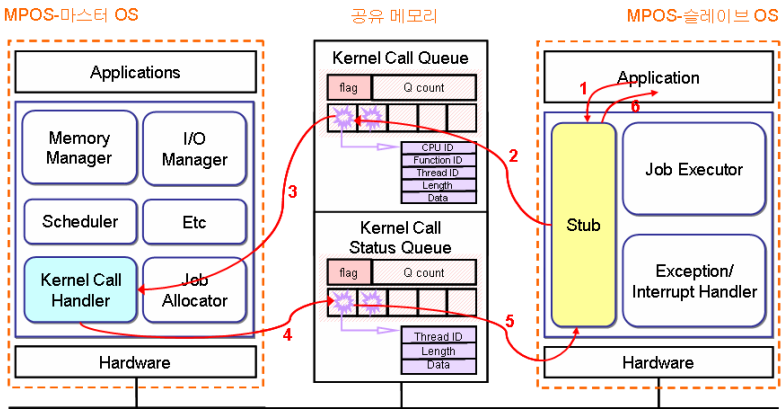
다. (<그림 1>)



<그림 1. Job 할당 및 처리 흐름도>

2.2. 커널 콜 서비스 요청 및 처리

슬레이브 운영체제에서 수행되는 Job(application)이 커널 함수를 수행할 경우 마스터 운영체제에 커널 콜 서비스를 요청 한다. 슬레이브 운영체제의 Stub은 해당 어플리케이션을 블록 시키고 대기 큐에 등록한다. Stub은 서비스를 요청할 함수의 ID와 요청하는 thread의 ID, 함수의 argument, return type을 마스터의 Kernel Call Queue에 보낸다. 마스터 운영체제의 Kernel Call Handler는 Kernel Call Queue에 슬레이브 운영체제가 요청한 커널 콜 서비스가 있다면 해당 커널 함수를 수행한다. 처리가 완료되면 kernel Call Handler는 결과 값을 해당 슬레이브의 Kernel Call Status Queue에 보낸다. 이후 Stub은 Kernel Call Status Queue에 반환된 결과 값이 있는지 검사하고 있으면 블럭된 Job을 ready 상태로 변경하고, 결과 값을 전달한다. (<그림 2>)



<그림 2. 커널 콜 서비스 요청 및 처리 흐름도>

참고문헌

[1] Andrew S. Tanenbaum, “Modern Operating Systems, 2nd ed.,”PrenticeHall, 2001

[2] J. A. Kahle, M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, D. Shippy, “Introduction to the Cell Multiprocessor,” IBM Journal of Research and Development Vol. 49, No. 4/5, 2005

[3] 김지민, 유민수, “SoC 설계와 검증을 지원하는 실시간 운영체제,” 한국정보처리학회 춘계학술발표회 논문집, 2005년 5월