논문 2010-05-03

빠른 문맥전환을 위한 임베디드 시스템 구조

(Fast Context Switching Architecture in Embedded Systems)

손 정 호*
(Jeongho Son)

Abstract : In real-time embedded systems, the responsibility is the most important thing because it is related to human life. Context switching is a part of which can slow down the responsibility. We therefore should minimize the amount of state that needs to be saved during context switching. In this paper, we introduce a new architecture (Register Farm) for context switching which can exchange two contexts in one cycle time. Although it might increase the cost of MCU design and the complexity of circuit, it cannot miss any interrupt during context switching.

Consequently, Register Farm architecture can make embedded systems spread out in human life because it can increase reliability and responsibility in real time embedded systems.

Keywords : register farm, context switching, embedded system

Ⅰ. 서 론

최근 임베디드 시스템은 다양한 분야로 급속히 확산되고 있다. 특히 자동차 및 항공기 등의 분야에서는 사용자의 편의 및 안전에 관련된 요구사항이 크게 증가하고 이 요구사항은 대부분 ECU와 ECU를 제어하기 위한 소프트웨어의 요구사항으로 나타난다. 만약 브레이크, 에어백 등의 동작에서 ECU와 소프트웨어가 사용자가 원하는 시간 내에 응답하지 못하면 큰 사고로 이어져 인간의 생명 및 많은 재산 피해를 유발할 수 있다. 따라서 임베디드 시스템에서 실시간성이 보장된 신뢰성은 매우 중요한 요소로 자리 잡고 있다.

1. 실시간 임베디드 시스템의 과제

임베디드 시스템을 사용하는 사용자 측면에서 시스템을 구성하는 장치들의 고급화에 대한 멀티태스킹, 우선순위 기반의 선점 스케쥴링, 인터럽트 중단 기간의 최소화 , 빠른 문맥전환, 메모리 관리 기능 등 고급 요구사항들이 증가하고 있다[1]. 이러한 임베디드 시스템에서는 최근 '실시간 응답성' 이 가장 큰 과제로 나타나고 있다. 하지만 대부분의 임베디드 시스템의 MCU는 문맥전환에 매우 많은 시간을 소모한다[1-4]. 이전 태스크의 상태정보를 저장하고 수행되어야 할 태스크의 정보를 복원하는데 긴 시간이 소요된다. 또, 문맥전환에서 인터럽트의 발생을 금지하므로 인터럽트를 놓치는 경우가 발생하여 응답을 제공하지 못하기도 한다.

* 교신저자(Corresponding Author)
논문접수 : 2010. 1. 18., 수정일 : 2010. 02. 09.,
채택확정 : 2010. 2. 22.
손정호 : 한국전자통신연구원

2. 관련 연구

1990년대부터 지금까지 문맥전환을 최소화하기 위해 많은 시스템 구조가 제안되었다. 그 중 Winfried Grunewald 등은 실행 중 활성대기(Active waiting)를 없애는 Rhamma Processor[4]를 도입하였다. 이 구조는 로드/저장, 동기화 과정이 서로 다른 장치로서 분리되어 동작하도록 설계되었다. 결과적으로 이 시스템 구조를 통해 문맥전환 시간이 1 클록 사이클로 단축되었다.

Peter R. Nuth 등은 문맥을 위한 캐시(Context Cache)를 제안하였다[5]. 이 구조는 개별 레지스터에 변수를 연결하여 필요할 때만 실제 레지스터에 값을 로드하거나 저장하는 것을 수행하였다. 시뮬레이션 결과에서 기존의 레지스터 파일과 비교해서 제안된 시스템이 문맥전환의 속도를 7% 향상시켰다.

Wang Qin은 그림 1과 같이 문맥전환의 비용을 줄이기 위해서 두 개의 문맥구조를 처리할 수 있는

마이크로프로세서(DCPA)를 제안하였다[7]. 두 개의 문맥에 대해 각각 패치 유닛, 디코드 유닛, 확장 유닛을 가지고 있어 병렬 처리까지 가능한 구조를 가지고 있다. 또, 두 문맥간의 전환에 소요되는 비용은 거의 무시할 수 있다. 하지만, 이 구조는 각 패치 유닛 등의 중복과 병렬 처리에 대한 처리의 복잡도가 증가하는 단점이 있다.

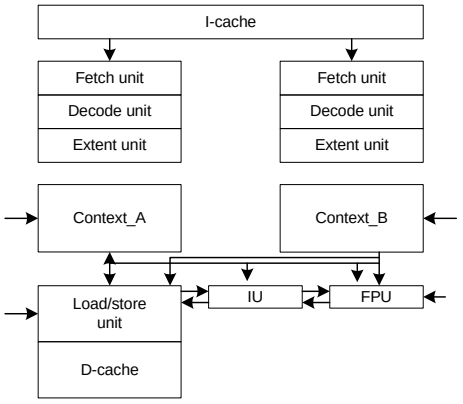


그림 1. DCPA processor의 블록 다이어그램
Fig. 1. Block diagram of DCPA processor

II. 본 론

1. 제안된 레지스터 팜 구조

우리는 1장에서 실시간 임베디드 시스템이 가진 '실시간성'에 관한 문제가 많은 부분 문맥전환에 있다는 것을 언급하였다. 문맥전환의 시간을 단축하기 위해서 스택에 상태정보를 저장 하고 인터럽트를 놓치지 않기 위해서 최소한의 결정적 구간을 찾아 인터럽트를 금지하기도 한다. 하지만, 성능을 향상시킬 수 있을 뿐, 여전히 문맥전환 과정에서 인터럽트를 놓칠 수 있다.

본 논문에서는 실시간 임베디드 시스템이 PC와 달리 지정된 몇 개의 태스크가 이미 메모리에 상주하고 동작한다고 가정하였다. 이 가정을 바탕으로 각각의 태스크에 해당하는 레지스터 집합을 중복적으로 MCU 안에 삽입하고 이를 '레지스터 팜'이라 명명하였다. 이 레지스터 팜에서 특정 레지스터의 집합을 선택하기 위해서 '선택자'를 도입하였으며, 이 선택자를 사용하여 1 사이클 시간에 문맥전환이 가능한 시스템을 제안한다.

그림 2는 '레지스터 팜'을 도입한 MCU의 내부 구조를 보여준다. 각 태스크 당 프로그램 카운터, 스택, 범용 레지스터를 두고 이들을 하나의 레지스터 집합으로 구성하며 각 레지스터의 집합을 선택자가 선택할 수 있는 구조이다. 선택자는 상태 및 제어레지스터를 통해 각 태스크에 대한 정보를 얻을 수 있다.

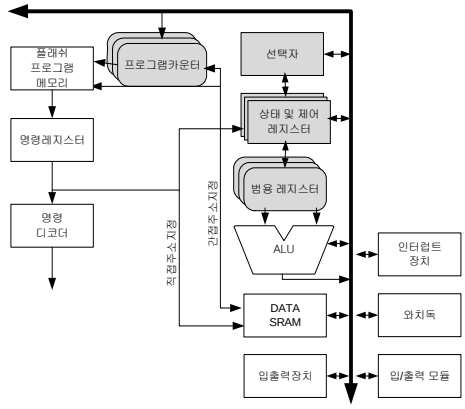


그림 2. 제안된 레지스터 팜 MCU 구조
Fig. 2. Architecture of register farm MCU

실제 레지스터 팜에는 그림 3과 같이 데이터 버스의 수를 줄이기 위해 하나의 레지스터로 통합된 형태로 구성하였다. 하나의 레지스터 집합을 살펴보면 개념적인 부분과 차이가 있다는 것을 발견할 수 있다. PC, SP를 범용 레지스터들과 같이 하나의 레지스터 집합에 속하도록 함으로써 관리가 용이하다.

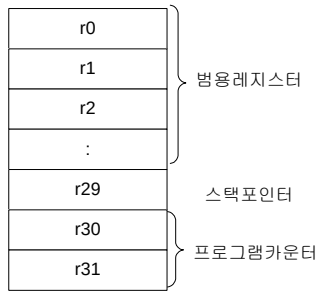


그림 3. 레지스터 집합
Fig. 3. A register set

초기 PC와 SP를 레지스터 팜의 외부에 두고 설계되어 PC를 통한 주소를 디코딩하는 방식으로 레지스터 팜의 하나의 집합을 다루었다. 하지만, 이

방식은 PC 와 SP에 접근하는 시간을 증가시키고 인터럽트 발생을 막는 구간이 발생하여 1 사이클에 문맥전환을 수행하는 것이 힘들고 관리하는 방법이 까다롭다는 단점이 있어 그림 2와 같은 구조로 개선하였다.

상태 및 제어레지스터 내부에는 선택자에 관련된 레지스터를 포함하고 있다. 이 레지스터는 선택자가 어떤 레지스터 집합을 선택할 것인지 결정하기 위해 필요한 정보를 가지고 있다. 초기 각 태스크는 자신이 수행하는 명령의 진입점, 자신이 사용해야하는 스택 지점, 레지스터 집합과 태스크와의 맵핑정보 등을 통해 선택자가 전용 레지스터 집합을 결정하고 접근을 통제한다.

2. 명령어 요구사항

제안된 시스템에서 효율적인 문맥전환을 수행하기 위해 (1)선택자가 특정 레지스터 집합에 접근하기 위한 명령, (2)상태 및 제어 레지스터에서 선택자와 관련된 부분을 제어하기 위한 명령, (3)문맥전환을 위한 명령, 전용 레지스터 집합을 사용하지 않는 태스크를 수용하기 위해서 (4) 다른 레지스터 집합에 접근할 수 있는 명령이 추가되었다.

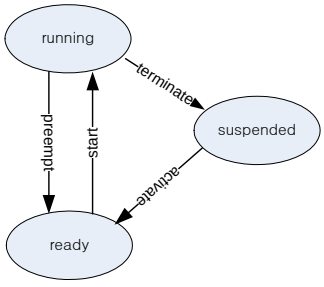
(1)은 문맥전환 명령에 의해서 선택자가 특정 레지스터 집합으로 접근 버스를 옮기는 기능을 수행한다. 이 때 상태 및 제어 레지스터에 기록된 정보를 바탕으로 레지스터 집합을 선택한다. (2)는 임베디드 시스템의 Start Up 코드에서 상태 및 제어 레지스터에 태스크 정보에 해당하는 값을 설정하고 이 값에 의해서 선택자가 동작할 수 있도록 지원한다. (3)은 태스크에서 발생하는 문맥전환 명령코드로 선택자가 레지스터 집합의 접근을 옮기는 기능을 수행한다. (4)의 경우는 전용 레지스터 집합을 사용하지 않는 태스크를 지원하기 위해서 제공되어야 하는 명령어로 범용으로 활용하는 레지스터의 로드/저장 하는 기능을 수행하는 명령이다.

다시 말해, 전용 레지스터 집합을 사용하는 태스크간의 문맥전환에서는 1 사이클의 시간이 소요된다. 이미 언급한 대로, 문맥전환 중 2사이클 이상이 요구되면 결국 임계영역을 만들어 무결성을 보장하기위한 조치가 필요하기 때문에 이를 방지하고 인터럽트를 놓치지 않도록 하기 위해서 반드시 요구되는 조건이다. 만약 문맥전환과 같은 사이클에 인터럽트가 발생하였다면, 일반적으로 문맥전환 직후에 즉시 인터럽트 서비스 루틴을 수행한다.

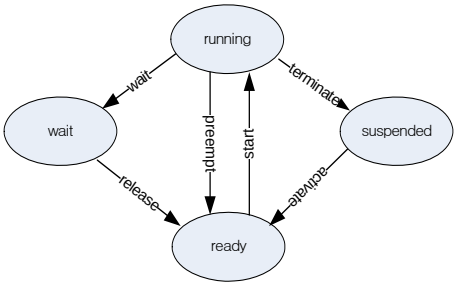
만약 전용 레지스터를 할당받지 못한 태스크가 존재하는 경우 레지스터 집합들 중 하나를 범용으로 사용하며 전용 레지스터를 사용하는 태스크와 범용 레지스터를 사용하는 태스크간의 문맥전환을 위해 이들 레지스터 집합 간의 접근을 허용하는 명령이 사용된다. 때문에 태스크의 수가 레지스터 팜 내의 레지스터 집합의 수보다 작거나 같도록 운영하는 것이 매우 중요하다. 만약 전용 레지스터 집합을 사용하지 않는다면 선택자는 하나의 레지스터 집합을 범용으로 사용하므로 일반적인 문맥전환과 동일하게 수행한다.

3. 동작

임베디드 시스템은 메모리배치는 물론 태스크의 스택공간 및 진입지점이 프로그래밍, 컴파일, 링커 단계에서 확정된다. 초기 임베디드 시스템이 시작하는 Start up 단계에서 각 레지스터 집합이 할당되고 이 맵핑정보를 선택자와 관련된 상태 및 제어 레지스터에 기록한다. 다음 그림 4는 OSEK OS의 태스크 상태 천이도를 통해 Start up 단계 이후의 레지스터 팜의 동작을 기술한다.



(a) wait 상태가 없는 천이도



(b) wait 상태가 존재하는 천이도

그림 4. OSEK OS의 태스크의 상태 천이도
Fig. 4. State transition diagram of OSEK OS

(1) 초기 suspend 상태의 태스크는 자신의 스택포인터, 프로그램 카운터의 값을 할당된 레지스터

집합에 초기 값을 보관한다. 만약 모든 태스크가 running상태에 진입하기 전이라면 레지스터 집합 중 하나는 임베디드 시스템의 Start up 동작을 위해 사용된다. 이 후 IDLE태스크가 이 레지스터 집합을 전용으로 사용한다.

(2) 초기 시작 태스크 혹은 인터럽트 등 기타 조건에 의해서 suspend에서 ready상태로 천이하는 경우 문맥전환 명령을 통해 IDLE 태스크로 문맥을 전환한다.

(3) 스케쥴 함수의 명시적 호출이나 기타 스케쥴링이 요구되는 조건에서 ready큐의 태스크는 경쟁에 의해서 최우선 순위의 태스크가 선별되고 문맥전환을 위한 명령을 사용하여 선택자가 다음 태스크의 전용 레지스터 집합을 선택하도록 한다.

(4) 이 태스크가 그림 4(b)와 같이 wait상태로 이동하는 경우 (3)의 설명과 같이 스케줄 발생이 요구되는 것과 같은 동작을 수행한다.

만약 전용 레지스터의 개수보다 많은 태스크가 수행되는 경우 위 (1)~(4)와는 다른 예외 처리가 요구되며, 일반적인 태스크의 절차와 조합하여 처리가 가능하다.

4. 문맥전환에 소요되는 비용

표 1에서는 전용 레지스터 집합을 사용한 경우와 전용/범용 레지스터를 사용하는 경우, 범용 레지스터만 사용하는 경우에 대한 문맥전환 비용을 비율로 나타내고 있다. 제안된 전용 레지스터 집합을 사용하는 시스템 구조는 초기 상태 및 제어레지스터의 설정비용을 배제할 경우 의 태스크간의 문맥전환은 문맥전환을 위한 전용 명령어를 이용하므로 1 사이클에 문맥전환을 수행할 수 있다.

표 1. 문맥전환에 소요되는 비용
Table 1. The cost of context switching

	전용레지스터 사용	범용레지스터 사용
전용레지스터 사용	1 사이클	$1/2^{(*)}$
범용레지스터 사용	$1/2^{(*)}$	$1^{(*)}$

($^{(*)}$: 범용 레지스터 사용 태스크간의 비용에 대한 비)

일반 범용 레지스터를 이용한 문맥전환을 1이라 할 때, 전용/범용 레지스터를 사용하는 경우 일반 범용만을 이용할 때의 문맥전환 비용에 대해 1/2정도의 비용이 든다는 것을 알 수 있다. 전용/범용 레지스터를 사용하는 경우에는 이전 태스크가 전용을 사용하는 태스크이면 레지스터에 저장하는 비용이 절약되고 다음 범용 레지스터를 사용하는 태스크의 로드 비용이 요구되므로 전용 레지스터만 사용하는 경우에 비해서 1/2의 비용이 발생하는 것이다. 반대의 경우에도 비슷하게 저장하는 비용이 들고 로드하는 비용이 요구되지 않아 1/2의 비용이 소모된다. 전용 레지스터 집합을 사용하는 태스크와 범용 레지스터 집합을 사용하는 일반 태스크간의 문맥전환은 하나의 문맥에서 두 개의 레지스터 집합에 접근해야 하므로 다소 복잡도가 증가하지만 문맥전환에 소요되는 시간을 1/2로 줄여 인터럽트를 놓치는 경우를 줄여 임베디드 시스템의 신뢰성을 향상시킨다.

본 제안된 '레지스터 팜'을 효과적으로 이용하기 위해서는 한정된 태스크를 통해 전용레지스터만을 사용하여 1사이클 문맥전환을 수행하면 인터럽트를 놓치지 않아 임베디드 시스템의 문맥전환에서 신뢰성을 보장할 수 있다.

5. 제안된 구조의 장점과 단점

제안된 시스템 구조는 개념적으로 매우 간단한 방법을 사용하고 있지만 레지스터의 집합을 중복 배치하므로 발생하는 비용이 증가하고 하드웨어 구조는 더 복잡해졌다. 또, 문맥전환을 위한 전용명령어도 추가되었다. 하지만, 멀티 태스크 프로그래밍에서 태스크 전환을 위한 전용명령어를 이용하면 사용자 투명성을 가진 선택자가 끊김 없는 레지스터 접근을 제공하므로 프로그래머는 손쉽게 원하는 기능을 수행할 수 있다. 또, 프로그램의 디버깅에서 선택자를 확인하면 태스크의 전환을 쉽게 파악할 수 있다는 장점이 있다. 또, 임베디드 시스템에서 타이머는 매우 중요한 역할을 수행한다. 지정된 시간에 특정동작을 수행하도록 설계된 경우에서 태스크의 전환 중에 금지된 인터럽트로 인해 지정된 기능을 수행하지 못하는 문제가 발생할 수 있다. 전체적인 시간을 측정하는 경우에도 타이머의 인터럽트를 놓쳐 시간의 오차가 발생하는 문제가 발생하고 이로 인해 임베디드 시스템간의 통신 혹은 협업에 문제를 야기할 수 있다. 제안된 시스템 구조는 이런 문제를 최소화하는데 크게 기여하고 있다.

제안된 구조는 빠른 문맥전환을 바탕으로 문맥전환 시 프로세서의 상태 값의 로드와 프로그램카운터의 설정시간 사이에 발생할 수 있는 정의되지 않은 상태로의 천이를 방지할 수 있어 임베디드 시스템의 신뢰성을 제공할 수 있으며 우선순위 정보

를 태스크 정보(태스크 제어 블록: TCB)에 보존하므로 유연한 우선순위 기반 문맥전환을 제공하고 다양한 스케쥴링 기법을 구사할 수 있어 높은 실시간 응답성을 요구하는 응용에 활용될 수 있다.

III. 결 론

본 논문에서는 실시간 임베디드 시스템의 문맥전환에서 발생하는 지연과 임계영역으로 인해 응답성을 보장하지 못하는 문제를 해소하기 위해 '레지스터 팜' 구조를 제안하였다.

제안된 시스템은 비용과 복잡도가 증가하는 단점을 가지고 있지만 문맥전환이 1 사이클에 동작하므로 인터럽트를 놓치지 않고 빠른 문맥전환으로 높은 응답성을 제공하기 때문에 임베디드 시스템의 응용범위를 크게 확대할 수 있다.

참고문헌

[1] K. Tanaka, "Fast context switching by hierarchical task allocation and reconfigurable cache", IWIA'03, pp. 20-29, 2003.

[2] J. Kreuzinger, A. Schulz, M. Pfeffer, Th. Ungerer, "Real-time scheduling on multithreaded processors", RTCSA 2000, pp. 155-159, 2000.

[3] C. Huang, K. Hsieh, J. Li, J. K. Lee, "Support of paged register files for improving context switching on embedded processors", IEEE, pp. 352-357, 2009.

[4] W. Grunewald and T. Ungerer, "Toward extremely fast context switching in a block-multithreaded processors", CSE'09, pp. 592-599, 1996.

[5] P. R. Nuth, "Parallel processor architecture: a thesis proposal", MIT VLSI Memo, 1990.

[6] P. Jaaskelainen, P. Kellomaki, J. Takala and H. Kultala, "Reducing context switch overhead with compiler assisted threading", EUC'08, pp. 461-466, 2008.

[7] W. Qin, "Double context microprocessor architecture", HPC 2000, pp. 370-371, 2000.

저 자 소 개

손 정 호

2000년 : 경주대학교 컴퓨터공학과 학사.
2002년 : 경북대학교 정보통신학과 석사.
2007년 : 경북대학교 컴퓨터공학과 박사.
현재, 한국전자통신연구원 선임연구원.
관심분야 : 무선네트워크, RTOS, 센서네트워크.
Email : phdson@etri.re.kr