ARMulator를 이용한 시스템 프로그래밍 관점의 가상 프로토타입 설계 및 구현

최혁상°, 조상영 ,이정배
한국외국어대학교 컴퓨터공학과°,
{tecumseh°, sycho}@hufs.ac.kr, jblee@sunmoon.ac.kr

Virtual Prototype design and Implementation from a system-programming point of view using ARMulator

Hyuk-Sang Choi°, Sang-Young Cho, Jung-bae Lee
Computer Engineering Department,
Hankuk University of Foreign Studies, Sunmoon University

요 약

소프트웨어 개발에 있어 가상 프로토타입의 활용은 하드웨어와의 병렬적인 개발 진행, 하드웨어 변경에 따른 신속한 대처, 확장된 디버깅과 벤치마킹 정보 등을 통해 개발 효율을 증대시킨다. 본 논문은 ARM을 기반하는 시스템의 소프트웨어 개발을 위한 가상 프로토타입 구현에 대해 다룬다. ARM사의 ADS1.2에서 제공하는 ARMulator의 Instruction Set Simulator를 기반하여 소프트웨어 개발자 관점의 추상화 수준으로 System-on-chip인 삼성 S3C2400의 축소된 형태를 가정하여 가상 프로토타입을 설계 및 구현하였다.

1. 서 론

가상 프로토타입은 컴퓨터상에서 가상적으로 구현된 하드웨어 시스템이다. 이것은 다양한 활용과 장점을 지니는데
첫째, 실제 하드웨어 개발 전에 설계를 검증하고 성능을 평가하여 실제 구현상의 시행착오를 줄이고 설계 변화에 유연하게 대처 할 수 있다 .
둘째, 소프트웨어 개발을 하드웨어와 병렬적으로 진행하여 전체 개발 시간을 단축시킬 수 있다 .
셋째, 보다 다양하고 확장된 디버깅 정보들을 제공해 주고 물리적 오류에서 벗어나게 함으로써 소프트웨어 개발 자체의 효율을 증대시킬 수 있다.

본 논문은 앞서 설명한 활용과 장점에서 두 번째와 세 번째에 해당할 수 있는, 프로그램 개발 목적의 가상 프로토타입(Virtual Prototype, 이하 VP)을 설계 및 구현한다. 구현 대상으로, System-on-chip인 삼성 S3C2400에서 기반 구조는 유지한 체 IP모듈들의 수를 줄인 하드웨어 시스템을 가정하였다.

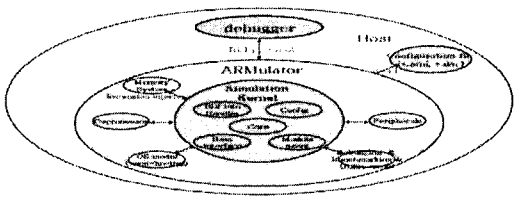
프로그래머는 시스템에 대해 프로그래머 모델에서 제시하는 보다 추상화된 기능과 성능을 인터페이스로 한다. 따라서 구현 수준은 S3C2400 user' s manual[7]의 프로그래머 모델을 준하였다.

개발 환경으로 ARMulator를 사용하였다. ARMulator는 ARM사의 ADS1.2에서 제공하는 ARM기반 시스템 시뮬레이션 환경으로 다양한 ARM 프로세서들에 대한 시뮬레이터를 제공해주며 ARM 디버거에 기본적으로 연동이 되어 소프트웨어 개발용 폐를 구성하는데 필요한 많은 기능을 제공하고 있다.

본 논문의 구성은 2장에서 VP를 개발하는 입장에서의 ARMulator 구조와 모듈들간의 연결 그리고 기본 동작에 대해 알아보고 3장에서는 확장 모듈 들을 설계하고 구현하는 데 있어 전형적인 형태를 모듈 기능 별로 살펴본다. 4장에서는 목표 하드웨어에 대한 설계 및 구현에 대해 기술한다: 5장에서는 구현된 시스템의 동작을 확인하며 마지막으로 6장에서는 결론과 향후 연구과제를 논한다.[1]

2. ARMulator

2.1 구조



[그림 1] ARMulator 구조

ARMulator는 [그림1]과 같이 시뮬레이션 커널을 기본으로 다양한 주변장치 모듈들이 제작 추가되어, ARM 프로세서를 기반으로 하는 임의의 가상 하드웨어 시스템을 구성 할 수 있다.

시뮬레이션 커널은 ARM Instruction Set Simulator와 cache, MMU시뮬레이터를 Core에 포함하고 있어 설정에 따라 다양한 ARM 프로세서가 구성된다. 추가된 주변장치 모듈들을 동적으로 연결하는 요소(Module Agent), Debugger의 요구에 따라 시뮬레이션을 운영하는 요소(Core), 디버거로부터의 시뮬레이션 관련 디버깅 정보 요청을 처리하는 요소(RDI info Handler), 설정파일들을 관리하는 요소(Config), 모듈 디버깅을 위해 모듈에서 외부로 문자열 메시지를 보내는 요소(Host interface)로 구성된다.

[1] 본 연구는 대학 ITRC 육성지원사업의 연구 결과로 수행되었음

주변 장치 모듈들은 ARMulator Extension Kit이라는 라이브러리와 헤더파일을 이용하여 일정한 형식의 DLL파일로 제작 된다. 모듈들은 크게 그 기능에 따라 버스 역할을 하는 Memory System 모델, Coprocessor 모델, 타이머 등과 같은 일반적인 주변장치 모델, OS의 기능을 부분 지원하는 OS모델, 확장된 디버깅 및 벤치 마킹을 위한 모델 등으로 구분되어진다.

2.2 모듈 연결

Configuration file의 설정에 따라 ARMulator 초기화 될 때 확장 모듈 중 Memory System 모델이 다른 주변장치 모델보다 먼저 초기화 되는데, 이 때 자신의 접근 관련 함수와 데이터에 대한 포인터를 core에 등록하고 다른 확장모듈들을 load하여 초기화 시키면서 해당 접근 함수와 데이터에 대한 포인터를 자신의 자료구조에 등록한다.

2.3 동작

프로세서는 디버거의 명령에 따라 내부적으로 processor cycle 단위로 시뮬레이션을 실행하고 그 결과를 유지하며 해당 요청이 있을 때 이에 대한 처리를 한다. 프로세서 시뮬레이션 실행 시 매 bus cycle 마다, 초기화 때 등록된 memory system의 access함수가 호출되며 매개변수로 버스 시스템을 컨트롤하는 정보와 전송 데이터에 대한 정보가 전달된다. 이 함수는 내부적으로 컨트롤 정보를 분석하여 내용에 따라 메모리를 포함한 주변장치에 접근하여 버스가 한 싸이클 동안 할 수 있는 일을 한다. 주변 장치에 대한 접근은 메모리 시스템에 등록된 각 장치 access 함수에 대한 호출로써 이루어진다. 메모리 시스템의 access함수는 다양한 반환 값을 취할 수 있는데, BUSY의 경우 한 버스 싸이클 만큼의 실행 지연이 발생하게 된다.

3. 확장 모듈 제작의 전형적인 형태

3.1 메모리 시스템 모델

확장 모듈 중 가장 중심적인 모듈로서 하드웨어의 버스 시스템을 추상화한 모델이다. 모듈 초기화 시 프로세서 모델 마다 차이가 있으나 920T 프로세서의 경우 프로세서 메모리 인터페이스에 다음과 같은 형식의 callback 함수를 등록해야 한다.

```
int MemAccessCached(
                    void *handle, ARMword address,
                    ARMword   *data,      ARMul_acc
```

이 함수는 메모리 접근 싸이클과 휴지 싸이클을 포함하여 매 버스 싸이클에서 호출되어 진다. 이 때 address로는 접근 주소를 보내고 data를 통해서는 R/W 데이터를 유지하며 acc로는 버스 컨트롤 정보(r/w, idle, sequential, non-sequential,…)를 보낸다. 이 함수 안에서 메모리를 포함한 모든 장치에 대한 접근 동작이 정의된다. 반환 값으로 DONE/ BUSY/ERROR등의 형식이 있다. BUSY를 반환할 경우 프로세서 시뮬레이션에 wait상태가 반영된다. 이를 통해 시뮬레이션에 적절한 시간지연을 발생 시킬 수 있다.

메모리 시스템 모듈은 여러 개가 계층적으로 구성될 수 있는데 이를 통해 정진적인 모델 상세구현을 효과적으로 할 수 있다. 시뮬레이터에서 기본 제공하는 메모리 시스템 모델로서 mapfile과 flatmem을 제공하는데 이 둘은 설정에 따라 계층적으로 메모리 시스템을 구축하게 된다. flatmem은 가장 기본적인 버스 기능과 메모리 공간을 제공하며 mapfile은 접근 형식에 따른 메모리에 대한 시간 지연을 반영시킨다. 추가 기능을 위해 새로운 메모리 시스템 모델을 정의하고 이를 기존의 것과 계층적으로 연결시키면 된다.

3.2 Coprocessor 모델

모듈 초기화 시 프로세서의 coprocessor 인터페이스에 coprocessor 번호와 함께 각 coprocessor 명령에 해당하는 특정 형식의 callback 함수들을 정의하고 등록해 주어 구현한다. 등록된 함수는 해당 coprocessor명령이 실행 될 때 마다 호출된다. 함수는 상태에 따라 적절한 값을 반환하여 프로세서 동작에 반영된다.

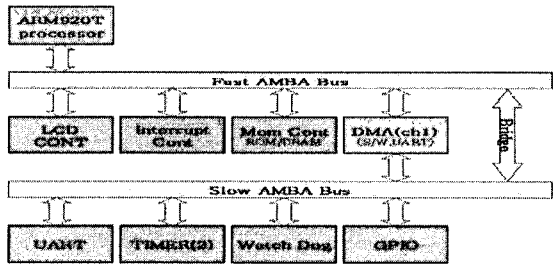
3.3 Peripheral 모델

메모리 시스템 모델에 의해 load되어 초기화 된다. 초기화 시 해당 모듈의 접근을 위해 정의된 함수와 관련 내부 변수에 대한 포인터가 메모리 시스템 모델의 bus관련 자료구조에 등록된다. 메모리 시스템이 접근 요청에 대하여 주소를 분석하고 해당 장치의 등록된 함수를 호출함으로써 동작이 이루어진다. 기타 시간단위 동작을 하는 타이머나 인터럽트 시그널을 보내는 인터럽트 컨트롤러를 위해 ARMulator Extension Kit 라이브러리 관련 인터페이스에서는 시간 스케줄링 함수와 인터럽트 시그널 발생 함수 등을 제공해 준다.

3.4 기타 모델

ARMulator Extension Kit은 시뮬레이터 요소와 내부 데이터 그리고 이벤트를 접근하기 위한 다양한 인터페이스들을 제공하고 있다. 이를 응용하여 OS지원 기능 모델이나 디버깅용 모델 또는 편의성을 제공하는 기타의 모델 들을 구현 할 수 있다. ARMulator에서 기본 제공하는 semihost, tracer, pageTab등이 여기에 해당한다.

4.설계 및 구현



[그림2] Target System 구조

구현 대상으로 [그림2]와 같은 ARM 920T 프로세서 기반 하드웨어를 가정하였다. 기본 구조와 각 IP들은 S3C2400 user' s manual[7]을 따랐다.

4.1 Processor 모델

프로세서는 설정파일과 디버거의 옵션 선택으로 간단히 구성된다. ARMulator에서 제공하는 920T 프로세서는 캐쉬와 관련해서 싸이클 정확도가 실재 하드웨어와 다소 차이가 있으나, 동작 확인과 근사한 성능 측정에는 문제가 없다.

4.2 AMBA bus 모델

ARM 프로세서 기반의 SoC 시스템은 빠른 장치를 위한 AHB/ASB 버스와 느린 장치를 위한 APB버스로 주로 구성된다. 시스템 프로그래머 입장에서 대개, 구체적인 버스 프로토콜은 관심 밖이다. 단지 정확한 데이터 전송과 적절한 시간 지연 계산만이 고려 대상이다. 따라서 AHB/ASB 와 APB를 프로토콜에 상관없이 단지 시간 지연의 차이로만 구별되도록 하였다. 아뮬레이터에서 기본 제공하는 mapfile과 flatmem 모델은 메모리와 메모리 접근 종류에 따른 시간지연 그리고 기본적인 버스 기능을 제공한다. 이를 그대로 사용하고 프로세서와 DMA간의 중재 기능을 하는 메모리 시스템을 새롭게 구현하여 이들을 계층적으로 구성하였다. 버스 마스터로 프로세서와 DMA가 있는데 DMA가 우선순위가 높도록 하였다.

4.3 메모리 컨트롤러

메모리 컨트롤러는 메모리 접근 컨트롤과 시간 지연을 관장한다. 이는 mapfile과 flatmem을 통해 지원됨으로 이를 그대로 사용하였고 단지 코드를 위해 컨트롤 레지스터를 위한 공간만을 확보하였다.

4.3 LCD 컨트롤러

TFT 기능에 한정하여 기능을 모두 구현하였으나 내부 FIFO와 DMA 구성을 생략하였다. 이를 통해 FRAME의 주소 변화 시 발생할 수 있는 상황이 실재와 다르게 되었고 시간 지연에 대한 실재 하드웨어와의 차가 더해지게 되었지만 많은 시뮬레이션 실행 속도 향상을 얻을 수 있게 되었다. 실질적인 LCD 프레임 디스플레이를 위해 xTarget이라는 GUI 프로그램을 구현하였다. 이 프로그램은 ARMulator 초기화 시 자동으로 실행되어 WIN32 IPC인 메모리 맵을 통하여 해당 Frame 데이터를 자신의 영역에서 디스플레이 한다.

4.4 Interrupt controller/Timer/Watch Dog Timer

ARMulator에서 기본 제공하는 각 모델을 기본으로 하여 구조와 기능을 더하여 구현하였다. 인터럽트 컨트롤러의 경우 설정에 따른 인터럽트 우선순위 기능을 더하였으며, 타이머의 경우 PWM(Pulse Width Modulation) 출력을 더하였다. PWM은 GTKwave프로그램에서 시그널 출력을 확인해 볼 수 있도록 해당 형식의 log파일이 만들어 지도록 하였다.

4.5 DMA

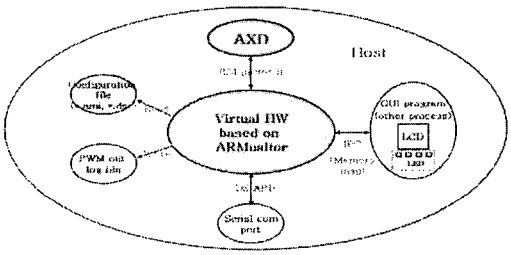
여러 채널 중 하나의 채널만 구현하였고 request source 를 S/W와 Uart로 한정하였다. 구체적인 프로토콜에 상관없이 전송 수준에만 관점을 두어 데이터의 이동과 시간 지연만을 고려하여 구현하였다.

4.7 GPIO

S3C2400은 여러 GPIO 그룹을 갖는데 포트 그룹 C와 D의 일부만을 구현하였다. 이중 포트 그룹 C의 12~15번에 LED를 연결하였고 해당 LED는 LCD와 같은 형식으로 xTarget프로그램의 영역에 표현되도록 하였다.

4.8 UART

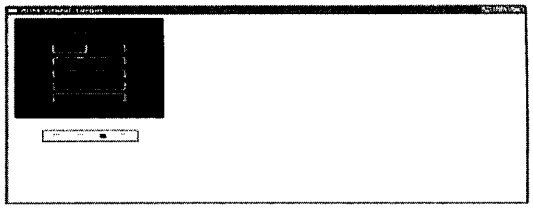
WIN32의 Uart관련 API를 이용하여 host 컴퓨터의 시리얼 포트를 통해 외부 장치와 통신할 수 있도록 구현하였다.



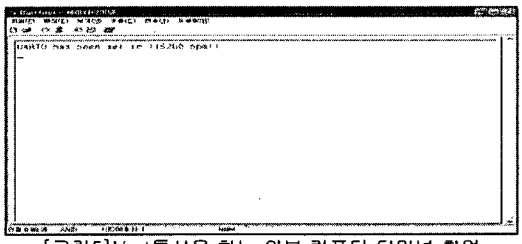
[그림3] 구현한 가상 프로토타입 전체 구성도

5.동작 확인

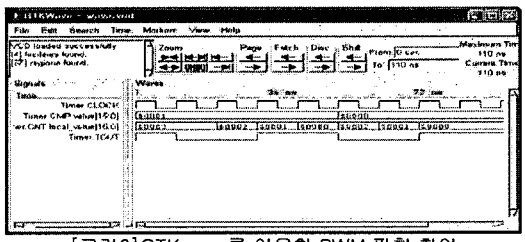
구현한 가상 프로토타입의 동작 확인을 위해 각 장치 관련 S3C2400 Test 프로그램을 응용하여 디지털 시계와 Uart통신 프로그램 그리고 PWM 파형을 변화 시키는 Timer 응용프로그램을 작성하여 실행시켜 보았다. 디지털 시계는 타이머, LCD, 인터럽트 장치,GPIO를 이용하여 시간 진행에 따른 시/분/초의 변화와 LED를 켜고 끄는 프로그램이며 Uart 통신 프로그램은 host의 COM포트에 연결된 외부 컴퓨터와 간단히 문자열을 주고 받는 프로그램이다. 실행 결과 아래 그림들에서처럼 각 장치들이 설계대로 동작하였으며 디버거를 통한 내부 동작확인에서도 정상적인 작동을 확인할 수 있었다.



[그림4]디지털 시계의 LED와 LCD Display



[그림5]Uart통신을 하는 외부 컴퓨터 터미널 화면



[그림6]GTKwave를 이용한 PWM 파형 확인

6.결론 및 향후 계획

본 논문에서는 ARMulator를 이용하여 간단한 SoC 시스템에 대한 가상프로토 타입을 설계 구현하여 보았다. 프로그래머 관점의 동작 구현과 프로그램 개발에 필요한 다양한 정보를 확장된 인터페이스를 통해 제공함으로써 프로그램 개발의 편의를 도모하였지만 정확한 계산 모델을 통한 성능 측정에 대한 부분이 충실하지 못하였다. 향후 시스템 각 구성 장치에 대한 명확한 계산 요소를 추상화해 시간 정확도를 보완하고 ARMulator에 종속되어 있는 ISS를 독자적인 것으로 개발하도록 할 것이다.

참 고 문 헌

[1]ARM Cop. " ARM DDI0100E ARM Architecture Reference Manual" .

[2]ARM Cop. " ARM IHI 0011A AMBA Specification(Rev 2.0)" .

[3]ARM Cop. " ADS1.2 / Develper Suite / Debug Target Guide" .

[4]ARM Cop. " ARM DDI015C ARM920T Bus Interface Unit"

[5]Steve Fuber," Arm System on chip Architecture" , Addison Wesley, 2000.

[6]ARM Cop. " ARM DAI0032E Application Note32 The ARMulator" .

[7]Samsung Cop. " S3C2400X User' s Manual"