QEMU를 기반으로 한 ERC32 프로세서 에뮬레이터 개발

최종욱*, 신현규**, 이재승***, 천이진****

Development ERC32 Processor Emulator based on QEMU

Jong-Wook Choi*, Hyun-Kyu Shin**, Jae-Seung Lee***, Yee-Jin Cheon****

Abstract

During the development of flight software, the processor emulator and satellite simulator are essential tools for software development and verification, which can be substituted for the actual hardware. LEO satellites being developed by KARI recently use the MCM-ERC32SC processor for on-board computer (OBC). For the flight software (FSW) development and testing, the software-based spacecraft simulator was developed using TSIM-ERC32 processor emulator from Aeroflex Gaisler. It is needed to get rid of the constraints and dependencies of TSIM-ERC32 processor emulator and to obtain high performance processor emulator to develop full satellite simulator. This paper presents the development of the ERC32 emulator based on open source dynamic translator, QEMU, as the first step. And it describes the software development and testing/debugging on the developed emulator.

초 록

위성 탑재소프트웨어를 개발하는 과정에서 프로세서 에뮬레이터와 위성 시뮬레이터는 핵심 툴로서, 소프트웨어 개발과 검증 단계에서 사용되며 실제 하드웨어를 대체할 수 있는 수준까지 활용이 가능하다. 현재 한국항공우주연구원에서 개발 중인 저궤도 위성의 탑재컴퓨터의 프로세서는 SPARC v7 기반의 MCM-ERC32SC 프로세서를 사용하며, 프로세서 에뮬레이터의 경우 Aeroflex Gaisler에서 판매되는 TSIM-ERC32 에뮬레이터를 사용한다. 국내 인공위성 개발 시 ERC32 프로세서를 계속 사용할 경우 TSIM-ERC32의 제한 조건과 종속성을 벗어날 필요가 있으며, 추후 위성 시뮬레이터 개발 시 고성능의 프로세서 에뮬레이터가 요구되기 때문에 새로운 프로세서 에뮬레이터 개발 필요성이 지속적으로 대두되었다. 본 논문에서는 첫 번째 단계로 공개형 프로세서 에뮬레이터인 QEMU를 기반으로 ERC32 프로세서 에뮬레이터 개발 방법에 대해서 기술하며 개발 된 에뮬레이터 상에서의 소프트웨어 개발 및 디버깅 방법에 대해서 설명한다.

키워드 : 에뮬레이터(Emulator), QEMU, ERC32, laysim-erc32, TSIM-ERC32

접수일(2011년 9월 8일), 수정일(1차 : 2011년 10월 14일, 게재 확정일 : 2011년 11월 1일)

* 위성비행소프트웨어팀/jwchoi@kari.re.kr
** 위성비행소프트웨어팀/hkshin@kari.re.kr
*** 위성비행소프트웨어팀/jslee@kari.re.kr
**** 위성비행소프트웨어팀/yjcheon@kari.re.kr

1. 서 론

위성 탑재소프트웨어를 개발하는 과정에서 프로세서 에뮬레이터와 위성 시뮬레이터는 핵심 개발 툴로서, 단위 테스트, 통합 테스트, 검증시험 및 시스템 테스트 등의 개발/검증 전반에 사용되고 있으며 실제 하드웨어를 대체할 수 있는 수준까지 활용이 가능하다. 현재 한국항공우주연구원에서 개발 중인 저궤도 위성의 탑재컴퓨터의 프로세서는 Astrium/ESA에서 개발한 SPARC v7 기반의 MCM-ERC32SC 프로세서를 사용하며, 프로세서 에뮬레이터의 경우 Aeroflex Gaisler에서 상용으로 판매하고 있는 TSIM-ERC32 프로세서 에뮬레이터를 확장하여 사용하고 있다.

ERC32 프로세서 에뮬레이터의 경우 대부분 ESA 관련 업체에서 개발된 자체적인 프로세서 에뮬레이터가 사용되며 이중 오직 Aeroflex Gaisler사에서 개발된 TSIM-ERC32만이 상용으로 판매되고 있다. 국내 인공위성 개발 시 ERC32 프로세서를 계속 사용할 경우 지속적으로 TSIM-ERC32를 구매해야 하고 TSIM-ERC32를 이용하여 위성 시뮬레이터 개발 시 연동되는 프로세서 에뮬레이터의 구조를 변경 및 수정할 수 없는 어려움이 발생한다.

이러한 문제들을 원천적으로 해결하기 위해서는 자체적으로 새로운 프로세서 에뮬레이터 개발이 반드시 필요하다. 하지만 새로운 프로세서 에뮬레이터의 개발에는 많은 시간과 노력이 필요하며 검증에 대한 문제가 항상 발생한다. 또 다른 방법으로 기존 프로세서 에뮬레이터를 확장 할 경우 상대적으로 구현이 간단하나 라이선스 문제와 SPARC 코어를 제공하는 프로세서 에뮬레이터를 쉽게 찾을 수 없는 문제가 존재한다[1][2].

본 논문에서는 기존 프로세서 에뮬레이터 중 오픈 소스형태로 배포되는 다수의 에뮬레이터에 대한 분석 이후 선택된 QEMU를 기반으로 하여 ERC32 프로세서 에뮬레이터 개발 방법에 대해서 기술하며 개발 된 에뮬레이터 상에서의 탑재소프트웨어 개발 및 디버깅 방법에 대해서 설명한다. 그리고 Dhrystone을 이용한 TSIM-ERC32와 개발된 QEMU for ERC32 프로세서 에뮬레이터의 성능을 비교 분석한다.

2. 공개형 동적 변환 방식의 에뮬레이터 - QEMU

ERC32 에뮬레이터인 TSIM-ERC32의 경우 인터프리팅 (interpreting) 방식으로 프로세서를 에뮬레이션 해주며, 실제 ERC32 프로세서보다 높은 실시간 성능을 제공한다. 하지만, LEON2/3 프로세서의 경우 ERC32 프로세서보다 3~5배 높은 성능을 제공하기 때문에 기존 인터프리팅 방식으로는 에뮬레이터가 실시간성을 만족하지 못하는 문제가 발생하였다. LEON2의 경우 실제 하드웨어 성능은 86DMIPS (100MHz 기준)가 나오는 반면 TSIM-LEON2의 경우 최대 30DMIPS 정도의 성능을 보여준다[3]. 이러한 문제를 해결하기 위해서 Aeroflex Gaisler에서는 LEON2 AT697F 프로세서를 하드웨어 가속화 방식으로 에뮬레이션 해주는 TSIM-HW를 개발 및 판매하고 있으나 LEON3를 지원 하지 못하는 문제를 가지고 있다. ESA에서는 Coimbra 대학을 통해 “LeonVM : Using Dynamic Translation for Development High-Speed Space Processor Emulators[4]” 기술을 개발하여 실시간성을 보장하는 소프트웨어 기반의 LEON2 프로세서 에뮬레이터를 개발하였다. 영국의 SciSys사에서는 오픈소스 형태의 프로세서 에뮬레이터인 QEMU을 기반으로 ERC32 및 LEON2 프로세서 에뮬레이터[5]를 개발하여 GALILEO 시뮬레이터로 사용하고 있다. Ada Core에서는 자체 Ravenscar run-time을 검증하기 위하여 QEMU 확장 형태의 Couverture[6] 에뮬레이터를 개발하여 ERC32/LEON2 소프트웨어 개발 툴로 사용하고 있다.

아래의 그림 1은 일반적인 프로세서 에뮬레이터의 구조를 보여주며, Target Processor Emulator에 구현된 QEMU for ERC32 에뮬레이터가 위치하게 되고, VxWorks/RTEMS 등의 실시간 운영체제가 상위에 놓이며, 중간에 Processor와 OS간의 저수준의 입출력을 담당하게 된다.

표 1. ESA Emulator Support

Software Emulator	Method	ERC32	LEON2	LEON3	LEON3 (MMU)	Supplier
TSIM	Instruction Level	√	√	√	√	Aeroflex-GR
TSIM-HW	H/W Accelerated		√			Aeroflex-GR
Leon-SVE	Instruction Level	√	√			Spacebel
SimERC32 /SimLEON	Instruction Level	√	√	√		Astrium/CNES
Sim-SCOC3	Instruction Level				√	Astrium (internal)
Sim-MPDA	Instruction Level		√			Astrium (internal)
QERx (LGPL)	Dynamic Translation	√	√			FFQTECH SciSys
Couverture	Dynamic Translation	√	√			AdaCore
ESOC Simulator	Instruction Level	√				ESOC

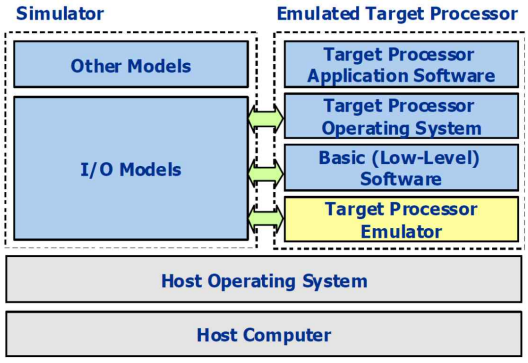

그림 1. Processor Emulator Architecture

기존 프로세서 에뮬레이터는 각 단계별로 타깃 프로세서의 명령어를 실시간으로 해석(인터프리팅)하는 방식으로 호스트 컴퓨터에서 에뮬레이션 하는 방법을 사용하였으며 실시간으로 타깃 프로세서를 구동하기 위해서는 타깃 프로세서의 클록 속도 보다 호스트 컴퓨터의 속도가 보통 50~100배 빨라야 한다. MIL-STD-1750, ERC32 프로세서의 경우 프로세서의 클록 속도가 일반 PC 보다 월등히 느리기 때문에 인터프리팅 방식의 에뮬레이션이 가능하였으나 LEON2/3의 경우 200MHz까지 클록 속도가 높아졌기 때문에 기존 방식의 인터프리팅 기법으로는 실시간성을 만족시킬 수가 없다.

QEMU의 경우 동적 변환 (dynamic translation)[7]이라는 새로운 방법을 사용한다. 이 방법은 타깃 명령어를 호스트 컴퓨터의 명령으로 실시간(on-the-fly) 컴파일 하고, 컴파일 된 명령어 블록을 메모리에 저장한다. 보통 블록은 branch 단위 혹은 프로세서의 상태정보가 바뀌는 명령들의 집합으로 구성되며 이는 보통 5~10개의 명령어로 구성된다. 호스트 컴퓨터에서 수행할 수 있도록 컴파일 된 타깃 명령어 블록이 재수행 될 경우 기 컴파일 된 명령 블록을 수행함으로서 에뮬레이터의 성능을 향상시키게 된다. 이러한 동적 변환방식을 이용함으로서 QEMU의 경우 기존 인터프리팅 방식의 에뮬레이션 기법보다 5~10배 이상 성능 향상되었으나, 실제 타깃 프로세서의 타이밍과 I/O 타이밍이 문제가 될 수 있는 단점을 가지고 있다.

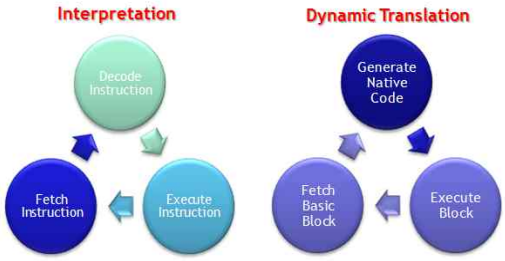

그림 2. Interpretation vs Dynamic Translation

QEMU은 타깃 프로세서로 x86, PowerPC, ARM, SPARC 등의 프로세서를 지원하며, Linux, Windows와 같은 운영체제를 수정 없이 구동할 수 있도록 시스템 에뮬레이션이 가능한 가상머신 기능까지 제공하고 있다. 하지만 공식적으로 QEMU에서 ERC32, LEON2/3 프로세서를 지원하지 않는다. 그러나 QEMU에서 타깃 프로세서로 SPARC을 지원하며 ERC32의 경우 SPARC v7, LEON2/3의 경우 SPARC v8 이고 기본적으로 IU (Integer Unit) 및 MEIKO FPU (Floating Point Unit)는 QEMU에서 지원되기 때문에 큰 어려움 없이 QEMU for ERC32 개발이 가능하다.

그러나 실제 QEMU을 기반으로 ERC32 프로세서 에뮬레이터를 개발 할 경우, 대부분의 공개형 소프트웨어의 가장 큰 취약점인 전체 모듈에 대한 기본/상세 문서 및 개발 방식에 대한 설명이 전혀 없으며 다수의 개발자들에 의한 서로 다른 표준 사용 등 초반 개발 과정에서 많은 어려움을 경험하게 된다. 항우연/위성비행소프트웨어 팀에서는 코드 레벨에서의 분석과 각 모듈간의 의존성에 대한 분석이 우선적으로 수행되었으며, 그 이후 단계별로 ERC32 프로세서의 기능을 구현하였다.

3. ERC32 프로세서 에뮬레이터 개발

현재 QEMU는 가장 최신 버전인 0.15.0을 제공(http://www.qemu.org)하고 있지만 QEMU for ERC32는 0.10.5를 기반으로 최초 개발/포팅 되었으며 공개 테스트 버전으로 QEMU laysim-erc32 v0.1가 배포 되었다. 이후 0.11.1을 기반으로 재포팅 완료 후 QEMU laysim-erc32 v0.3로 배포되어 현재 내부적으로 사용하고 있다. 앞에서 언급한 것처럼 QEMU에는 ERC32를 지원하지 않기 때문에 ERC32 포팅/개발을 위한 QEMU에 대한 전반적인 아키텍처 분석과 동적 변환기에 대한 연구가 선행되었다. 분석 과정에서 각 단계별로 아래 표 2와 같이 단계로 포팅이 이루어졌으며, 실제 하드웨어에서 테스트 되었던 테스트 소프트웨어를 이용하여 에뮬레이터의 기능 확인과 성능 분석이 수행되었다.

또한 현재 사용 중인 TSIM-ERC32와의 연계성 및 대체성을 위해 TSIM-ERC32와 유사한 인터페이스가 개발되었으며 추후 위성시뮬레이터의 핵심 코어로 사용되기 위해 다양한 위성 관련 하드웨어 에뮬레이션이 추가적으로 포팅 될 예정이다. 그리고 이는 지상국 및 탑재소프트웨어 개발 및 테스트를 위한 시뮬레이터로도 개발 될 예정이다.

표 2. Development Steps for laysim-erc32

분석	포팅 내용
분석 #001	MCMERC32 프로세서 등록
분석 #002	IBMU-PM32 보드 등록
분석 #003	QEMU MMU 문제 분석
분석 #004	CPU Init 과정 분석
분석 #005	RAM/ROM 할당
분석 #006	ROM file 로딩 및 수행
분석 #007	ERC32 MEC 기본 기능 구현
분석 #008	ERC32 UART 기능 구현
분석 #009	ERC32 Timer 기능 구현
분석 #010	ERC32 Interrupt 기능 구현
분석 #011	QEMU laysim-erc32 v0.1 배포 based on 0.10.5
분석 #012	QEMU laysim-erc32 v0.2 배포 based on 0.11.0
분석 #013	QEMU laysim-erc32 v0.3 배포 based on 0.11.1

3.1 QEMU for ERC32 개발

앞에서 얘기한 것처럼 QEMU은 다양한 프로세서와 시스템 시뮬레이션 까지 지원하기 때문에 오직 SPARC 코어만을 제공 할 수 있도록 의존성을 모두 제거 하였으며, GDB와 연동 될 수 있는 인터페이스를 수정 하였다. 그리고 ERC32 프로세서를 지원하기 위하여 시스템 레지스터와 메모리가 할당 되었으며, ERC32의 인터럽트 컨트롤러를 지원할 수 있도록 QEMU IRQ 관련 코드

표 3. QEMU for ERC32 Functions

기능	구현	구현 내용
ROM	Yes	64KB ROM 구현
NVMEM	Yes	2x4MB Flash & Flash Controller 구현
RAM	Yes	6MB RAM 구현
Int. Controller	Yes	ERC32 Interrupt 구현
System Control Registers	Yes	80% 구현 (Fault detection/injection 미구현)
UARTs	Yes	UART A/B 구현
Timers	Yes	RTCT, GTP, WDT 구현
GPIO	Yes	User-Extension에서 제어
DMA	No	저궤도위성에서 사용하지 않음
Virtual Eth.	Yes	VxWorks 개발 환경으로 사용

가 수정되었다. 또한 VxWorks 상에서의 탑재소프트웨어를 개발 할 수 있도록 virtual ethernet 기능이 추가 되었다. 아래 표3은 QEMU for ERC32에서 실제 구현된 기능을 정리한 것이다.

3.2 ERC32 프로세서 초기화 과정

QEMU laysim-erc32가 최초 수행 되면 아래 그림 3과 같이 *cpu_init()*함수가 호출 되어 MCM ERC32SC 프로세서의 초기화를 수행하고 SPARC 레지스터를 할당하게 된다.

프로세서 초기화 과정 이후 그림 4와 같이 RAM(6MB), ROM(1MB)를 *qeum_ram_alloc()* 함수를 이용하여 virtual memory를 할당 받고, 할당 받은 memory를 *cpu_register_physical_memory()* 함수를 이용하여 mapping하게 된다. Mapping할 때 해당 Memory가 RAM인지 ROM인지를 구분해야 하며, I/O 영역은 *cpu_register_io_memory()* 함수와 *cpu_register_physical_memory()* 함수를 함께 사용해서 할당한다. 할당 된 메모리에 elf, a.out 및 binary 포맷 형태로 ERC32 실행파일 로딩이 가능하다.

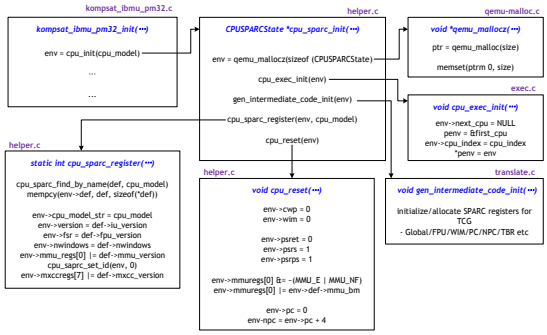

그림 3. Processor Initialization Process

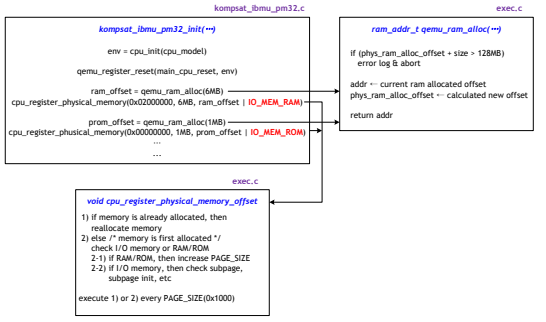

그림 4. Initialization for RAM and ROM

3.3 ERC32 MEC 기능 구현

ERC32 프로세서는 MEC (Memory Controller Unit)을 가지고 있으며, MEC 유닛은 3개의 타이머, 인터럽트 컨트롤러, 2개의 UART, 메모리 제어기, I/O 컨트롤러, fault detection/injection, 그리고 소프트웨어가 엑세스할 수 있는 레지스터들을 제공한다. QEMU laysim-erc32에서 가장 핵심이 되는 MEC을 구현하기 위해서는 ERC32 프로세서 자체의 기능 이해가 가장 중요하며 이를 기반으로 MEC의 기능을 구현해야 한다. 기존 RAM/ROM 메모리 할당과 다르게 I/O를 할당하기 위해서는 I/O memory structure가 선언되어야 하며, I/O access를 하는 callback 함수를 등록하게 된다. 즉 MEC을 할당하기 위해서는 그림 5와 같이 *cpu_register_io_memory()* 함수를 이용하여 *erc32_mec_read()*, *erc32_mec_write()* callback 함수를 등록하게 되고, 각 callback함수는 해당 주소를 액세스 할 때마다 각 기능별로 구현 된 하드웨어 에뮬레이션 함수가 수행되어야 한다.

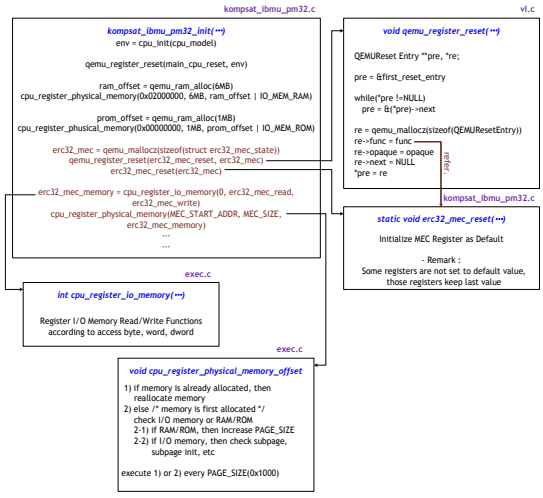

그림 5. Allocation of MEC

ERC32는 그림 6과 같이 두개의 UART를 제공하고 있으며 QEMU laysim-erc32에서는 MEC 레지스터와 연동하여 이벤트 기반의 시리얼 터미널을 구성하였으며 ERC32 프로세서에서는 인터럽트 기반으로 동작하게 된다. QEMU laysim-erc32 초기화 과정에서 *qemu_chr_add_handlers()*를 통해 ERC32 UART handler를 등록하게 되고, 각 핸들러에서 UART를 통한 데이터 입출력에 맞게 데이터를 처리하고 해당 인터럽트를 발생시키게 된다. ERC32 UART의 경우 QEMU의 virtual console에도 연동 될 수 있으며 QEMU 외부 (예, linux console)에서도 연동 할 수 있다.

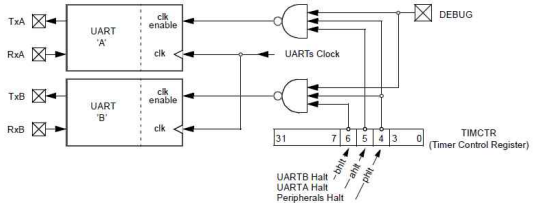

그림 6. ERC32 MEC UART Operation

ERC32 timer의 경우 그림 7과 같이 RTC (Real-Time Clock Timer), GPT (General-purpose Timer), WDT (Watchdog Timer) 3개의 Timer를 제공하고 있으며 모두 실제 하드웨어와 동일한 방식으로 컨트롤 된다. RTC의 경우 RTEMS/VxWorks의 시스템 클록으로 사용되며 GPT의 경우 다양한 목적으로 소프트웨어에서 사용된다. 실제 VxWorks/Tornado의 WindView를 통해서 시스템 시간을 측정할 경우 그림 10과 같이 정확히 60Hz 단위(16.67msec)로 동작하는 것을 확인할 수 있다. QEMU laysim-erc32 초기화 과정에서 *qemu_new_timer()*를 통해 ERC32 Timer handler를 등록하게 되고, 타이머 핸들러에서 Timer의 설정에 맞게 time tick이 발생할 수 있도록 동작하며, Timer timeout이 발생할 경우 Timer 인터럽트를 발생시키게 된다.

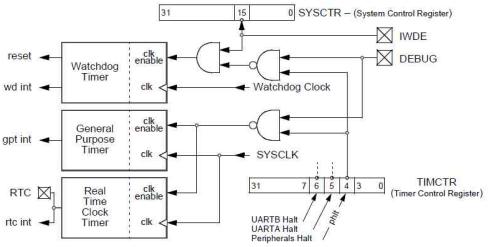

그림 7. ERC32 MEC Timer Operation

ERC32는 그림 8과 같은 인터럽트 컨트롤러 시스템을 사용하며, 15개의 외부/내부 인터럽트를 제공한다. QEMU laysim-erc32에서는 ERC32의 auto-vectored interrupt 방식을 지원하기 위해 QEMU 인터럽트 코어를 직접 수정하여 ERC32 인터럽트 컨트롤러를 구현하였으며 추후 LEON2/3로 포팅 할 경우에도 동일한 로직을 사용할 수 있을 것 판단된다. QEMU laysim-erc32 초기화 과정에서 *qemu_allocate_irqs()*를 통해 ERC32 Interrupt handler를 등록하게 되고, 각 핸들러에서 mask, pending, clear, force 등의 인터럽트 레지스터를 통해 인터럽트 발생 여부를 판단 한 뒤 *cpu_interrupt()* 함수를 통해 인터럽트가 발생했음을 전달하게 된다. 표 4는 QEMU laysim-erc32에서 구현된 인터럽트를 정리한 것으로 DMA 기능은 지원하지 않으며, VASI와 관련된 인터럽트의 경우 추후 VASI 코어가 개발될 경우 함께 구현 될 예정이다.

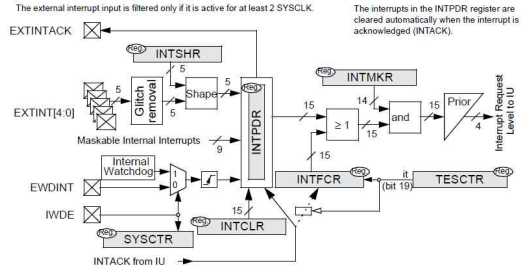

그림 8. ERC32 MEC Interrupt Operation

표 5는 QEMU laysim-erc32에 구현된 MEC 레지스터의 목록과 액세스 방식에 대해서 정리한 것이다. MEC 기능 중 현재 구현되지 않은 부분

은 fault detection/injection 관련 부분이며 해당 기능은 추후 시뮬레이터 개발 시 함께 구현될 예정이다.

표 4. ERC32 MEC Interrupts

Interrupt	tt	Comments
Watchdog timeout	0x1F	level=15 WDT는 Interrupt Mask Register를 통해서 masking할 수 없으며, IWDE가 1(active)인 경우에만 동작하게 된다. WDTTimeout이 발생하면 IRQ 15가 발생하게 되고, WDTResetTimeout안에 WDOGTR를 재설정 하지 않을 경우 WDTreset이 발생하여 ERC32/QEMU를 리셋 시킨다.
External INT 4	0x1E	level=14 현재 IBMU에서는 VASI RTC Cycle/Slot이 Ext. INT 4에 할당되어 있으며 추후 VASI를 포팅 할 경우 사용할 예정이다.
Real time clock timer	0x1D	level=13 ERC32 RTCTimeout이 발생하면 IRQ 13을 발생시킨다.
General-purpose timer	0x1C	level=12 ERC32 GPTTimeout이 발생하면 IRQ 12를 발생시킨다.
External INT 3	0x1B	level=11 현재 IBMU에서는 VASI Error Interrupt가 Ext. INT 3에 할당되어 있으며 추후 VASI를 포팅 할 경우 사용할 예정이다.
External INT 2	0x1A	level=10 현재 IBMU에서는 VASI Nominal Interrupt가 Ext. INT 2에 할당되어 있으며 추후 VASI를 포팅 할 경우 사용할 예정이다.
DMA timeout	0x19	level=9 DMA session exceeds permitted time
DMA access error	0x18	level=8 DMA performs an access error, access violation or illegal access
UART Error	0x17	level=7 ERC32 UART 통신 중 overrun/parity/frame 에러가 발생할 경우 IRQ 7이 발생한다.
Correctable memory error	0x16	level=6 SRAM Single Bit EDAC Error가 발생하면 IRQ 6이 발생하게 된다.
UART A Rx/Tx	0x15	level=5 ERC32 UART A 통신은 Byte단위로 Rx/Tx 인터럽트가 발생한다.
UART B Rx/Tx	0x14	level=4 ERC32 UART B 통신은 Byte단위로 Rx/Tx 인터럽트가 발생한다.
External INT 1	0x13	level=3 외부 인터럽트 1으로 사용하지 않음
External INT 0	0x12	level=2 외부 인터럽트 0으로 사용하지 않음
Masked hardware errors	0x11	level=1 ERC32의 H/W error와 발생했을 때 mask되어 있으면 IRQ 1 발생한다.

3.4 laysim-erc32 활용한 소프트웨어 개발 및 디버깅

현재 ERC32와 관련한 소프트웨어 개발 환경은 크게 리눅스 기반의 RTEMS와 윈도우 기반의 VxWorks/Tornado로 구분할 수 있다.

그림 9는 RTEMS 기반의 소프트웨어 개발 환경을 보여준다. RCC (RTEMS LEON/ERC32 Cross-Compiler System) 로 컴파일 된 ERC32 소프트웨어를 QEMU laysim-erc32에 로딩하여 정상적으로 멀티태스킹 소프트웨어가 동작하는 것을 확인할 수 있으며, GDB/DDD를 통하여 GUI 기반의 디버깅을 수행할 수 있다. 또한 구현된 UART를 통하여 결과 값을 출력할 수 있으며 TSIM과 유사한 인터페이스를 통하여 현재 프로세서의 IU/FPU 레지스터 정보, MEC 레지스터 정보 및 메모리 정보 등을 확인 할 수 있다.

표 5. ERC32 MEC System Registers

System Register Name		Address	Read/Write Access		Default
System Control Register	SYSCTR	0x01F80000	All R	Supervisor W	0x01350014
Software Reset	SWRST	0x01F80004	Supervisor W		-
Power-down	PDOWN	0x01F80008	Supervisor W		-
System Fault Status Register	SYSFSR	0x01F800A0	All R	Supervisor W	0x00000078
Failing Address Register	FAILAR	0x01F800A4	All R		0x00000000
Error and Reset Status Register	ERRRSR	0x01F800B0	All R	Supervisor W	0x00000000
Test Control Register	TESCTR	0x01F800D0	All R	Supervisor W	0x00000000
Memory Configuration Register	MCNFR	0x01F80010	All R	Supervisor W	0x00030000
I/O Configuration Register	IOCNFR	0x01F80014	All R	Supervisor W	0x00000000
Waitstate Configuration Register	WSCNFR	0x01F80018	All R	Supervisor W	0xFFFFFFFF
Access Protection Segment 1 Base Reg.	APS1BR	0x01F80020	All R	Supervisor W	0x00000000
Access Protection Segment 1 End Reg.	APS1ER	0x01F80024	All R	Supervisor W	0x00000000
Access Protection Segment 2 Base Reg.	APS2BR	0x01F80028	All R	Supervisor W	0x00000000
Access Protection Segment 2 End Reg.	APS2ER	0x01F8002C	All R	Supervisor W	0x00000000
Interrupt Shape Register	INTSHR	0x01F80044	All R	Supervisor W	0x00000000
Interrupt Pending Register	INTPDR	0x01F80048	All R		0x00000000
Interrupt Mask Register	INTMKR	0x01F8004C	All R	Supervisor W	0x00007FFE
Interrupt Clear Register	INTCLR	0x01F80050	Supervisor W		-
Interrupt Force Register	INTFCR	0x01F80054	All R	Supervisor W	0x00000000
Watchdog Timer Register	WDOGTR	0x01F80060	All R	Supervisor W	0xFFFFFFFF
Watchdog Timer Trap Door Set	WDOGST	0x01F80064	Supervisor W		-
Real-time Clock Timer <Counter> Register	RTCCR	0x01F80080	All R	Supervisor W	0xFFFFFFFF
Real-time Clock Timer <Scaler> Register	RTCSR	0x01F80084	All R	Supervisor W	0x000000FF
General-purpose Timer <Counter> Register	GPTCR	0x01F80088	All R	Supervisor W	0xFFFFFFFF
General-purpose Timer <Scaler> Register	GPTSR	0x01F8008C	All R	Supervisor W	0x0000FFFF
Timer Control Register	TIMCTR	0x01F80098	All R	Supervisor W	0x00000000
General-purpose Interface Configuration Reg.	GPICNFR	0x01F800A8	All R	Supervisor W	0x00000000
General-purpose Interface Data Register	GPIDATR	0x01F800AC	All R	Supervisor W	0x00000000
UART 'A' Rx and Tx Register	UARTAR	0x01F800E0	Supervisor R/W		0x00000000
UART 'B' Rx and Tx Register	UARTBR	0x01F800E4	Supervisor R/W		0x00000000
UART Status Register	UARTSR	0x01F800E8	All R	Supervisor W	0x00060006

그림 10은 Windows기반의 VxWorks/Tornado에서의 소프트웨어 개발 환경을 보여준다. QEMU laysim-erc32에서 virtual ethernet을 통한 Tornado의 Target Server 기능을 이용하여 에뮬레이터가 연동되고 소프트웨어 모듈 다운로딩 및 디버깅 등을 수행하게 된다. 또한 WindView를 통하여 각 태스크의 TimeStamp를 확인할 수 있으며 앞에서 언급한 것처럼 정확히 RTC 인터럽트가 동작하는 것을 확인 할 수 있다.

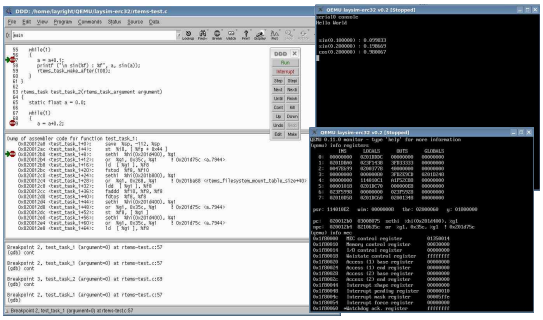

그림 9. S/W Development base on RTEMS

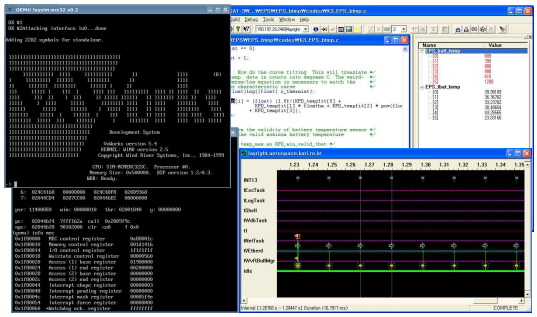

그림 10. S/W Development base on VxWorks/Tornado

동적 변환기의 가장 큰 장점인 고성능의 에뮬레이션의 경우 Dhrystone을 이용하여 성능을 측정한 결과 그림 11에서처럼 TSIM-ERC32에서는 10.1 DMIPS가 측정된 반면, QEMU laysim-erc32에서는 213.4 DMIPS가 측정되었다. 즉 프로세서 에뮬레이터의 시스템 클록에 대한 정수 연산의 성능을 보여주는 벤치마킹 프로그램인 Dhrystone을 통해 확인한 결과 높은 정수연산 성능을 보여주었다. 그리고 실제 시뮬레이션에서 의미가 있는 Dhrystone의 수행 시간을 절대시간을 측정 한 결과 QEMU laysim-erc32가 TSIM-ERC32 보다 7배가 빠르게 측정되었다. 이러한 고성능의 경우 실제 하드웨어와 동일한 성능을 요구하는 경우 문제가 될 수 있지만 추후 위성 시뮬레이터와 연동될 경우에도 다양한 시뮬레이션이 수행 되는 경우에도 프로세서의 성능을 보장할 수 있는 큰 장점이 될 수 있다.

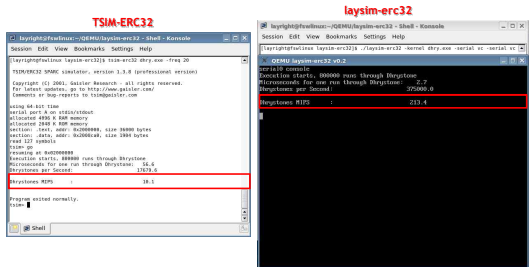

그림 11. Performance Comparison

4. 결 론

본 논문에서는 기존 ERC32 프로세서 에뮬레이터인 TSIM-ERC32의 제한 조건과 종속성을 벗어나기 위해 오픈 소스 기반의 동적 변환기 기반의 에뮬레이터인 QEMU를 기반으로 한 ERC32 프로세서 에뮬레이터 개발 방법에 대해서 알아보았으며, 개발 된 QEMU laysim-erc32상에서의 VxWorks 및 RTEMS 운영체제에서의 소프트웨어 개발 환경 및 Dhrystone을 통한 성능 측정을 수행하였다. 현재 개발된 QEMU laysim-er32는 TSIM-ERC32를 사용할 수 없는 외부 소프트웨어 개발섹션에서 사용되고 있으며 대학에서 ERC32 프로세서기 기반의 소프트웨어 개발을 위해 연구목적으로 사용되고 있다. 현재 항우연에서 개발중인 차세대 위성에서는 LEON2/3 프로세서가 사용될 계획이며 이를 위해 내부적으로 QEMU for LEON2/3가 연구 중에 있다.

또한 기존 프로세서 에뮬레이터를 기반으로 하지 않고 새로운 프로세서 에뮬레이터인 laysim-erc32[8][9], laysim-leon3[10]가 개발되었으며, 추후 항우연에서 개발하는 에뮬레이터 및 시뮬레이터의 코어로 사용될 예정이다.

참 고 문 헌

1. 최종욱, 이재승, 이상곤, “동적 변환기 기반의 고성능 ERC32 프로세서 에뮬레이터 개발”, 한국항공우주학회 춘계 학술대회, 2010, pp. 539~542.

2. 최종욱, 이재승, 양승은, 천이진, “Development of High performance ERC32 processor emulator based on Dynamic Translation”, Joint Conference on Satellite Communication 2010, pp. 91~96.
3. Sandi Habinc, Jiri Gaisler, “AT697F/LEON2-FT Processor Software Tools" Proceedings of DASIA 2007, Naples, Italy, 29 May 2007.
4. Marques, P. , Feiteirinha, J. , Pureza, L. , Lindman, N., “LeonVM: Using Dynamic Translation For Developing High-Speed Space Processor Emulators”, Proc. of the 9th International Workshop on Simulation for European Space Programmes (SESP’2006), Noordwijk, The Netherlands, November 2006.
5. Alastair Pidgeon, Paul Robinson, Sean McClellan, “QERx : A High Performance Emulator for Software Validation and Simulations”, Proceedings of DASIA 2009, Istanbul, Turkey, May 2009.
6. Matteo Bordin, Cyrille Comar, Tristan Gingold, “Couverture : an Innovative Open Framework for Coverage Analysis of Safety Critical Applications”, 14th International Conference on Reliable Software Technologies, Ada Europe 2009.
7. Fabrice Bellard, “QEMU, a Fast and Portable Dynamic Translator”, Proceedings of the 2005 USENIX Annual Technical Conference, April 10-15, 2005, Anaheim. CA. USA.
8. 최종욱, 이재승, 천이진, “GUI 기반의 Cycle-True ERC32 프로세서 에뮬레이터 개발”, 한국항공우주학회 추계 학술대회, 2010, pp. 951~954.
9. 최종욱, 신현규, 이재승, 천이진, “위성 시뮬레이터 개발을 위한 ERC32 프로세서 기반의 가상화 시스템 개발”, 통신위성우주산업연구회 논문지, Vol. 6, No. 1, 2011, pp. 50~56.
10. 최종욱, 이재승, 신현규, 천이진, “Virtualized System Development based on LEON3-FT processor for Satellite Simulator”, 한국항공우주학회 춘계 학술대회, 2011, pp. 969~972.