J. Astron. Space Sci. 25(3), 321-328 (2008) [기술논문]

위성 온-보드 소프트웨어 통합 성능 검증을 위한 프로세서 에뮬레이터 기반 시뮬레이트 소프트웨어 테스트 벤치 개념 연구

구철회[1†], 양군호[1], 최성봉[2]
[1]한국항공우주연구원 통신해양기상위성사업단 통해기체계팀
[2]한국항공우주연구원 통신해양기상위성사업단

Conceptual Study of Simulated Software Test Bench Based On Processor Emulator for Integrated Performance Verification of Satellite On-board Software

Cheol-Hea Koo[1†], Koon-Ho Yang[1], and Seong-Bong Choi[2]
[1]Communication Satellites Dept., COMS Program Office, KARI, Daejeon 305-333, Korea
[2]Communication Satellites Dept., KARI, Daejeon 305-333, Korea
E-mail: chkoo@kari.re.kr

(Received June 16, 2008; Accepted August 19, 2008)

요 약

위성 온-보드 소프트웨어 중 새로 개발되거나 일부분이 변경된 소프트웨어의 성능을 검증하기 위해서 요구되는 소프트웨어 테스트 벤치는 실제 구동환경을 가능한 가깝게 묘사하여야 하며 특히 결함 검출 및 복구 성능 검증에 대한 시험 환경을 충분히 제공하여야 한다. 실시간적인 성능 시험을 제외하면 프로세서 에뮬레이터 기반의 시뮬레이션 소프트웨어 테스트 벤치는 매우 매력적인 소프트웨어 성능 검증 환경을 하드웨어 소프트웨어 테스트 벤치보다 앞서 제공하는 것이 가능하다. 이 논문에서는 시뮬레이션 소프트웨어 테스트 벤치를 개발하기 위한 개념 연구 결과를 제시하고 있다.

Abstract

Software Test Bench should be simulated with maximum quality to real execution environment in order to verify the performance of software which is changed or newly developed. And especially faults detection and recovery are crucial function of the performance test environment. Simulated Software Test Bench based on processor emulator is attractive and can be available prior to hardware Software Test Bench if real time performance aspect is ignored. In this paper, the results of conceptual study for developing the simulated Software Test Bench are presented.

Keywords: software test bench, STB, simulated STB, simSTB

1. 서 론

위성 개발이 진행되면 임무에 따라 위성 하드웨어와 위성 온-보드 소프트웨어의 일부분은 수정이 불가피하다. 새로 개발되거나 일부분이 변경된 위성 온-보드 소프트웨어는 그림 1과 같이 요

†corresponding author

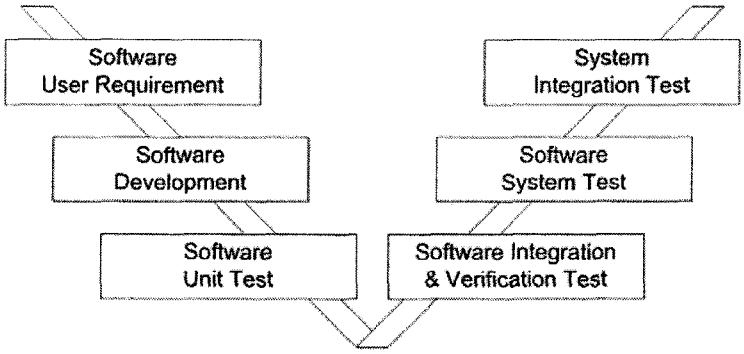

그림 1. E3000 비행 소프트웨어 개발 및 검증 V 다이어그램.

구사항 분석에서부터, 소프트웨어 구현 및 단위시험, 통합시험, 시스템 시험 단계를 거쳐 개발된다 (Astrium 2002, EUR3.PLN.07068.T.MMS). 이 논문에서는 소프트웨어 통합 및 검증 시험(Software Integration and Verification Test, SIVT)에 대한 부분을 중점으로 다루고자 한다. SIVT는 수정된 소프트웨어 모듈이 전체 소프트웨어 시스템에 탑재가 되었을 때 정확성뿐 아니라 호환성 및 안정성을 충분히 발휘하고 있는지 확인하는 것을 말한다. 따라서 SIVT를 수행하기 위해서는 소프트웨어 테스트 벤치(Software Test Bench, STB)가 가능한 소프트웨어의 구동 환경과 가깝게 구축되어 있어야 할 것이다. 위성 비행 소프트웨어(Flight Software)를 예로 든다면 위성 탑재 컴퓨터 및 주변 위성 제어 장치 등이 STB에 포함이 될 수 있다.

STB는 실제 하드웨어로 구축이 된 것과 하드웨어를 시뮬레이트한 에뮬레이터(Emulator)로 구성된 것, 두 부류로 구분할 수 있다. 하드웨어로 구축된 STB는 실제 위성 탑재 소프트웨어가 구동될 환경과 완전히 동일한 조건에서 시험을 수행할 수 있다는 장점이 있다. 하지만 버그를 발견하고 추적하는 것이 용이하지 않고 버그를 재현하는 것도 역시 쉽지 않다. 모든 것이 실시간으로 동작하고 중단점(Break point)을 잡는 것이 어렵기 때문이다. 하드웨어이기 때문에 취급에 소요되는 시간과 노력도 개발 일정 및 비용적인 측면에서 마이너스적인 요소이다. 결함 상황을 만드는 것 역시 용이하지 않고 효과적으로 사용하기가 힘들다. 예를 들어 소프트웨어 구동 중간에 메모리 에러가 발생하는 상황을 만들어야 한다면 실제 하드웨어에 어떻게 메모리 에러가 발생하도록 할 수 있는가?

또한 새로이 변경되는 하드웨어는 소프트웨어를 시험할 때에 아직 만들어지지 않았을 수도 있다. 이런 이유로 기술 및 비용 측면에서 값비싼 하드웨어를 소프트웨어로 대체하는 방안을 강구하는 것이 요구된다.

시뮬레이트 STB(simulated STB, simSTB)는 위에서 언급된 하드웨어 STB가 갖고 있는 단점들을 해결할 대안이다. 위성 컴퓨터 하드웨어 등을 묘사해주는 가상 구동환경에서 소프트웨어를 시험할 수 있도록 해주며 소프트웨어 디버그뿐만 아니라 결함을 인공적으로 주입하고 시스템의 복구 성능까지 테스트할 수 있게 해준다.

현재 위성 탑재 컴퓨터의 프로세서로 사용되고 있는 1750 프로세서와 ERC32 프로세서의 프로세서 에뮬레이터는 ESA 라이센스 또는 상용으로 개발되었으며 SIMSAT 등의 위성 시뮬레이터 및 비

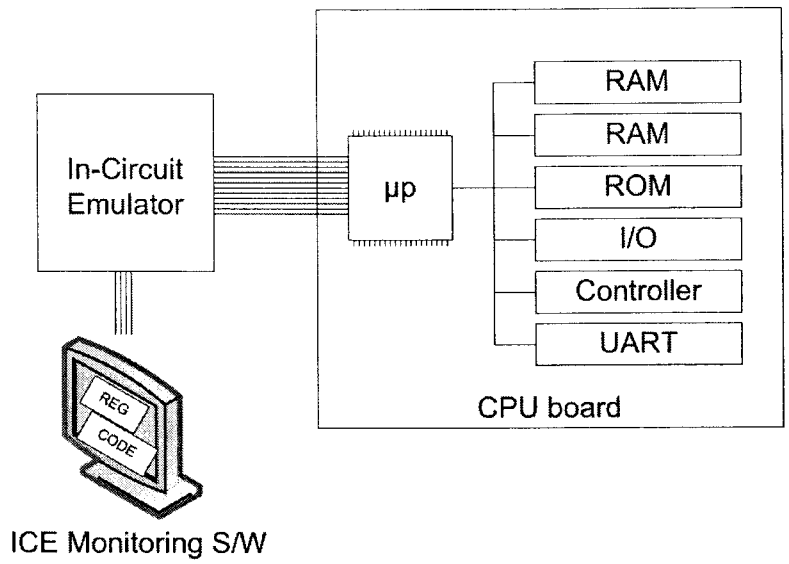

그림 2. 전형적인 프로세서 보드와 인-서킷 에뮬레이터 구성도.

행 소프트웨어 검증시 simSTB를 적극 활용하고 있으며 한국항공우주연구원에서도 저궤도 다목적 위성 3호, 5호 및 정지궤도 통신해양기상위성 1호 개발시에 simSTB를 활용하고 있으며 향후 국산화를 통해서 비행 소프트웨어 성능 검증의 기반기술로 발전시킬 전망이다(채동석 외 2006).

이 논문에서는 simSTB의 메커니즘과 개발 과정을 분석한 내용을 소개하고 향후 국내 기반 기술로 자리 잡기위한 단계적 개발 방안을 제시한다.

2. simSTB의 기술적 측면

simSTB는 시뮬레이트 소프트웨어 테스트 벤치(simulated Software Test Bench)이다. 그림 2와 같이 소프트웨어 성능 검증을 위해서 실제 프로세서가 탑재된 컴퓨터 보드와 인-서킷 에뮬레이터(In Circuit Emulator)로 구성된 소프트웨어 테스트 벤치가 고전적인 STB라고 한다면, 이러한 시스템을 단지 소프트웨어만으로 에뮬레이션(Emulation)한 것이 simSTB라고 정의내릴 수 있다.

simSTB는 크게 프로세서 에뮬레이터와 주변 I/O 시뮬레이터(Subsidiary I/O Simulator)로 구분할 수 있으며 그 전형적인 구성도는 그림 3에 나타내었다(구철회 외 2008). 에뮬레이션은 대상 하드웨어의 I/O 기능을 모사하기 위해서는 대상 하드웨어 내부의 기능 또한 정확히 모사되어야 할 경우 대상 하드웨어를 모사하는 방식이고, 시뮬레이션은 대상 하드웨어 I/O 기능을 모사하기 위해 대상 하드웨어 내부의 대략적인 모사로도 충분할 때 대상 하드웨어를 모사하는 방식으로 각각 구분지을 수 있다.

2.1 프로세서 에뮬레이터(Processor Emulator)

프로세서 에뮬레이터는 simSTB의 가장 중심에 있는 컴포넌트로 컴퓨터의 가장 중요한 성능 규격인 프로세서를 에뮬레이션한다. 예로 들어 SPARC 프로세서 에뮬레이터는 SPARC 프로세서의 연

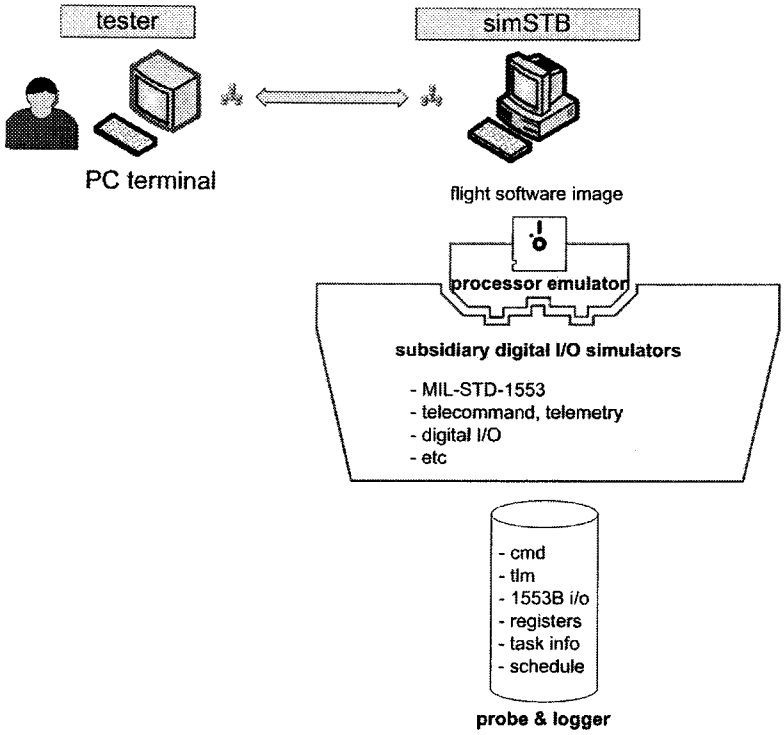

그림 3. 위성 simSTB 전형적 구성도.

산장치, 레지스터, 메모리, I/O(메모리 Mapping 등) 등을 그대로 에뮬레이션한다. 그래서 특별한 주변 장치를 필요로 하지 않는 SPARC 프로그램은 프로세서 에뮬레이터 상에서 바로 구동이 가능하다. 프로세서 에뮬레이터의 핵심 기능은 연산장치의 에뮬레이션이며 프로세서가 수행하는 수치 계산, 메모리 연산 등을 그대로 소프트웨어적으로 묘사한다.

프로세서의 기능은 모두 에뮬레이션하지만 소프트웨어적으로 하는 것이기 때문에 실제 수행속도와는 차이가 있다. 수행 시간을 측정하는 방법으로는 수행된 명령 개수를 세고 각각 명령이 수행되는데 걸리는 시간을 합산하는 방법을 사용한다. 최근의 하이 엔드 컴퓨터의 속도가 매우 빠르기 때문에 아주 빠른 프로세서를 에뮬레이션하지 않는다면 거의 실시간으로 프로세서를 에뮬레이션하는 것이 가능하다. 예로 위성 탑재 컴퓨터의 프로세서 구동 클럭을 30MHz 정도로 본다면 근래의 하이 엔드 컴퓨터 프로세서의 클럭은 2 ~ 3GHz에 이르기 때문에 실제와 큰 차이없이 구동하는 것이 가능하다. 프로세서 에뮬레이터는 상용과 오픈소스 모두 존재하지만 모두 프로세서가 에뮬레이터로 가용한 것은 아니니, 사용자가 사용하고자 하는 프로세서가 에뮬레이터로 존재하는지, 성능은 요구사항에 부합하는지 확인이 필요하다.

2.2 주변 I/O 시뮬레이터(Subsidiary I/O Simulator)

simSTB의 주변 I/O 시뮬레이터는 RS-232, RS-422 또는 MIL-STD-1553B와 같은 I/O를 시뮬레이션한다. 하지만 PCI 버스와 같은 소프트웨어적으로 의미없는 시스템 버스와 같은 것들은 시뮬레

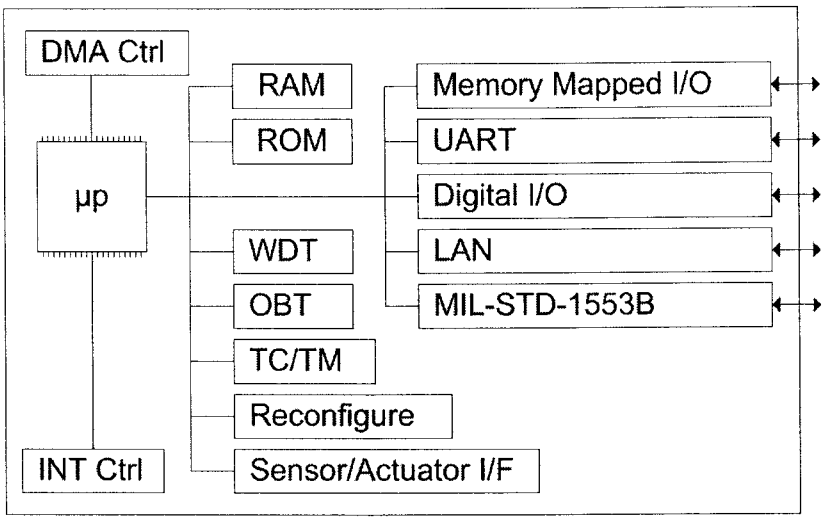

그림 4. 프로세서 주변 I/O.

이선하지 않는다. 소프트웨어 시험에서 하드웨어와 직접 관여하지 않는 부분의 구현은 고려되지 않는다. 예를 들어 PCI 버스에서 문제가 발견되었을 때 소프트웨어가 취할 수 있는 행동은 아무것도 없기 때문이다. simSTB는 가능성있는 하드웨어의 고장 또는 결함 모델이라 하더라도 고장 및 결함을 일부러 소프트웨어에 주입하는 용도가 아니면 시뮬레이션하지 않는다. 역시 결함 허용 로직에 특별한 요구사항이 있지 않는 한 모든 결함 상황도 고려할 필요가 없다. 그외에 순수 프로세서 외에 추가된 I/O 들은 모두 시뮬레이션 되어야 한다. 그림 4는 전형적인 위성 컴퓨터에서 사용 가능한 주변 I/O 들을 보여주고 있다(Edvin & Daniel 2002, Stuart 2000).

2.3 simSTB 인터페이스

사용자(테스터)와 simSTB 사이의 인터페이스는 simSTB의 활용성을 결정짓는 매우 중요한 규격이다. 사용자는 어떻게 simSTB를 사용할 지를 결정해야만 하며 개발된 소프트웨어를 simSTB를 통해서 전체적인 구동 흐름을 검증할 것인지, 아니면 디버거 수준의 상세 소프트웨어 구동 정보를 얻는 것이 목적인가에 대한 구분을 해야 한다. 그리고 한명이 사용하는지 다수가 동시에 사용하는지에 따라서 개발 규격도 달라진다. 다수의 사용자가 네트웍을 통해서 하나의 simSTB에 접속해서 사용하는 방식으로도 사용가능하고 simSTB를 각각의 사용자에게 단독으로 배포하여 사용하는 방식도 가능하다.

한국항공우주연구원의 정지궤도 통신해양기상위성 비행 소프트웨어 개발의 경우 소프트웨어 시험 절차 스크립트에 의해 다수의 사용자가 동시에 하나의 simSTB를 서로 다른 소프트웨어를 테스트하기 위해서 사용했다.

어느 정도의 디버거 수준의 정보를 얻기 위한 인터페이스는 simSTB와 사용자 사이에 반드시 요구된다. 이것이 시험 스크립트(Test Script)와 연동되면 인터페이스는 더욱 강력해진다. 사용자는 자신이 보고 싶은 디버그 정보에 대해서 시험 스크립트를 통해 simSTB에 요구할 수 있으며 따라서 사

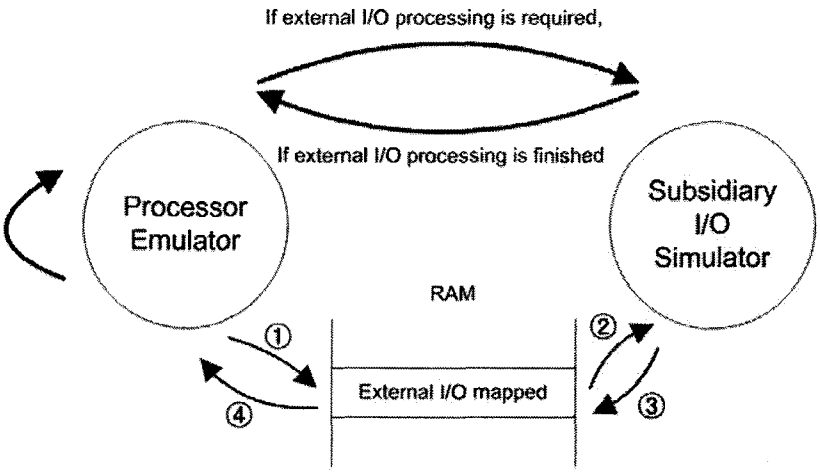

그림 5. 프로세서 에뮬레이터와 주변 I/O 시뮬레이터 동작도.

용자가 원하지 않는 정보의 처리량을 대폭 줄일 수 있을 것이기 때문이다(Jakob et al. 2006).

결함으로부터의 복구 능력을 시험하기 위한 결함 주입도 시험 스크립트로 가능하다. 사용자는 자기가 원하는 시점에 결함을 주입하고 그것이 전체 시스템에서 어떻게 처리가 되는지 확인할 수 있다. 사용자가 원하는 시점에, 또한 항상 반복해서 결함을 주입할 수 있기 때문에 simSTB는 하드웨어 STB 보다 디버그 관점에서 유리하다.

3. simSTB의 개발 개념

3.1 simSTB의 목적

그림 1에서 볼 수 있듯이 수정된 소프트웨어의 로직의 옳고 그름을 판단하는 단위 시험은 SIVT 단계 이전에 완료되며, simSTB에서 수행되는 SIVT는 수정된 소프트웨어가 전체 소프트웨어 시스템과 통합(Integration)이 됐을 때 호환성(Compatibility)의 만족 여부를 판단하는 목적을 가지고 있다.

simSTB는 하드웨어적인 실시간성을 완벽히 재현할 수 없기 때문에 SIVT의 최종 시험은 하드웨어 STB에서 수행되어져야 한다. 그리고 simSTB는 로직을 검증하기 위한 폐루프 시험(Closed-loop Test)을 수행하지 않는다. 요약하면 simSTB는 실시간 수행환경을 제공하지 않으며 로직 검증을 제외한 소프트웨어 호환성 시험에 그 목적을 두고 있다.

시험 로그(Test Log)를 통해서 시험의 수행 과정 및 결과를 해석할 수 있으며 사용자가 소프트웨어를 디버그하고 에러를 추적하는데 필요한 정보를 포함하도록 비침습적(Non-intrusive)으로 시험 과정에 개입할 수 있다.

3.2 simSTB의 구성

simSTB는 그림 3과 같이 프로세서 에뮬레이터와 주변 I/O 시뮬레이터로 구성된다. 프로세서 에뮬레이터는 프로세서의 동작만을 에뮬레이션하며 대부분의 경우 변경할 필요가 없는 부분이다. 따

라서 대부분의 simSTB 개발 작업은 주변 I/O 시뮬레이터의 개발에 대해서 이루어지게 된다.

프로세서 에뮬레이터는 수행도중 외부 I/O의 수행이 발생되거나 인터럽트가 발생하면 제어권을 주변 I/O 시뮬레이터에 돌려주게 된다. 위성 컴퓨터의 프로세서 외부에 달려있는 주변 I/O를 시뮬레이션해주는 주변 I/O 시뮬레이터는 프로세서 에뮬레이터로부터 제어권을 넘겨 받아 상황에 따라 적절한 연산 및 동작을 수행하게 되고, 그림 5에서 볼 수 있는 것과 같이 주변 I/O 시뮬레이터의 작업이 모두 종료되면 제어권은 다시 프로세서 에뮬레이터로 반환되어 나머지 처리를 계속하게 된다. 이런식으로 프로세서 에뮬레이터와 사용자가 구현한 주변 I/O 시뮬레이터가 번갈아 simSTB를 제어하며 동작하게 된다(Edvin & Daniel 2002).

시험 스크립트는 시험의 자동화(Automatical Testing)를 목적으로 사용될 것이다. 시험 자동화는 반복 시험이 필요할 때 매우 유용하다. 사용자는 시험이 종료된 후 로그 파일을 보면서 시험 결과를 확인하는 것으로 시험시 에러가 발생했는지의 여부를 파악할 수 있다.

3.3 simSTB의 확장

simSTB는 단적으로 stand-alone 환경의 소프트웨어 검증을 목적으로 하는 컴퓨터 시뮬레이터로 볼 수 있다. 프로세서 에뮬레이터와 주변 I/O 시뮬레이터는 정적으로 결합되고 단순히 소프트웨어 호환성 시험을 목적으로 사용되기 때문이다. 하지만 simSTB는 기술적으로 자세제어와 같은 위성 동역학 모델, 지상국 모델, 궤도 모델 등과 통합되어 실시간 시뮬레이션 환경(Real-time simulation environment)으로 확장될 수 있다. 물론 두 분야는 처음부터 목적을 달리해서 개발이 되어왔기 때문에 두 환경이 각각 최적화된 구조로 분리가 되었으나 기반 기술은 매우 유사하다고 할 수 있다. 따라서 앞으로 stand-alone 환경에서 실시간 시뮬레이션 환경으로 확장도 고려할 수 있다. 또한 simSTB 내부 모듈의 스케줄링 수정으로 정확한 실시간 응답 특성을 보장하는 방안도 향후 연구해야할 필요성이 있다.

4. 결 론

지금까지 이 논문에서는 simSTB의 개발에 있어서 기술적인 측면과 simSTB가 가져올 효과를 극대화하는 방안에 대해서 기술하였다. 소프트웨어 시험 및 검증을 위한 STB 사용시 simSTB를 도입하는 것은 시험의 효율을 증대시키고 소프트웨어 개발의 비용적인 측면과 위험관리 측면에서 모두 이익을 가져온다. simSTB가 비록 하드웨어를 완전히 대체할 수는 없지만 하드웨어의 대체재(Substitute)로서의 구현이 매우 용이하고 앞으로 실시간적인 동작도 구현 가능하게 될 것으로 전망되어 simSTB의 활용도를 앞으로 크게 기대한다.

특히 simSTB는 결함 주입 및 탁월한 디버그 성능으로 매우 실질적인 시험이 가능하다. 그리고 simSTB는 여러 소프트웨어 모듈에 대한 시험을 동시에 진행할 수 있다는 장점이 있다. 하드웨어 STB 만을 사용한다면 동시에 한 개 이상의 소프트웨어를 테스트하는 것은 현실적으로 어렵다. 그러므로 simSTB를 SIVT 단계의 주 시험 도구로 사용함으로서 기술 및 비용적인 측면에서 효과를 얻을 수 있다.

감사의 글: 이 논문은 교육과학기술부 "통신해양기상위성 시스템 및 본체 개발사업"의 일부임을 밝히며 연구지원에 감사를 드립니다.

참고문헌

구철회, 최재동, 박수현, 최소영, 강수연, 양군호 2008, 한국우주과학회보, 17(1), 42

채동석, 최종욱, 이재승 2006, 한국항공우주연구원 Technical Memo KARI-SWT-TM-2006-002

Edvin, C. & Daniel, H. 2002, High Level Description of an ASIC Implementing CPU Support and I/O Control (Goteborg: Chalmers University of Technology)

Jakob, E., Guillaume, G., & Bengt, W. 2006, Proc. ERTS 2006: Embedded Real-Time Software, p.1

Stuart, R. 2000, Embedded Microprocessor Systems, 2nd Ed (Massachusetts: Newnes), pp.53-79