시스템수준의 하드웨어 기능 검증 시스템
System-level Hardware Function Verification System

유명근* , 오영진** , 송기용***
Myoung-Keun You*, Young-Jin Oh**, Gi-Yong Song***
College of Electrical and Computer Engineering, Chungbuk National University, 361-763, Korea

요 약

시스템수준 설계방법론에서 널리 사용하고 있는 설계흐름도는 시스템명세, 시스템수준의 하드웨어/소프트웨어 분할, 하드웨어/소프트웨어 통합설계, 가상 또는 물리적 프로토타입을 이용한 통합검증, 시스템통합으로 구성된다. 시스템의 하드웨어 구성요소를 개발하는 과정에서 이전까지는 디자인단계가 많은 시간 및 노력을 요구하는 단계였지만, 현재에는 설계한 디자인의 기능적 검증단계가 중요 요소로 간주되고 있다. 본 논문에서는 시스템수준 설계언어인 SystemC 기반의 테스트벤치 구조를 이용하여 Verilog HDL로 설계된 하드웨어 구성요소의 올바른 동작여부를 판별하는 기능검증시스템을 설계하였다. 설계된 기능검증시스템에서 SystemC 모듈의 멤버 변수와 Verilog 모듈의 와이어 및 레지스터 변수간의 데이터 전달은 본 논문에서 정의되는 SystemC 사용자 정의 통신채널을 통하여 이루어진다. 제안된 기능검증시스템을 UART에 적용하여 올바른 동작여부를 판별하였다. 본 논문의 기능검증시스템 설계에 사용된 SystemC는 C++기반의 하드웨어 모델링용 클래스 라이브러리를 제공하므로 RT 수준보다 높은 추상화수준에서 소프트웨어와 하드웨어 또는 이 둘을 결합한 시스템수준의 모델링을 단일 언어와 환경에서 설계할 수 있는 이점이 있다. 또한 기능검증시스템 설계에 작성된 SystemC 모듈 코드들은 부분적인 코드 수정 후 다른 하드웨어 구성요소의 기능을 검증하는데 재사용할 수 있는 이점이 있다.

ABSTRACT

The flow of a universal system-level design methodology consists of system specification, system-level hardware/software partitioning, co-design, co-verification using virtual or physical prototype, and system integration. In the developing process of a hardware component in system, the design phase has been regarded as a phase consuming lots of time and cost. However, the verification phase in which functionality of the designed component is verified has recently been considered as a much important phase. In this paper, the implementation of a verification environment which is based on SystemC infrastructure and verifies the functionality of a hardware component is described. The proposed verification system uses SystemC user-defined channel as communication interface between variables of SystemC module and registers of Verilog module. The functional verification of an UART is performed on the proposed verification system. SystemC provides class library for hardware modeling and has an advantage of being able to design a system consisting hardware and software in higher abstraction level than register transfer level. Source codes of SystemC modules are reusable with a minor adaptation on verifying functionality of another hardware component.

Keywords : SystemC infrastructure, verification system, communication interface, higher abstraction level

I. 서 론

최근 시스템의 설계가 대형화되고 복잡해지면서 상위준수 추상화에 기반한 설계범위탐색이 보다 중요시되고 있다. 이에 따라 상위수준 추상화를 이용하여 시스템을 설계하는 SystemC와 같은 시스템수준 설계언어가 주목을 받고 있다[1-4]. SystemC는 하드웨어 모델링용 라이브러리를 포함시킨 C++ 클래스 라이브러리와 사건구동 방식의 시뮬레이션 커널로 구성된 시스템수준 설계언어로, 하드웨어 구성요소, 소프트웨어 프로그램, 그리고 이 둘이 결합된 시스템을 통합설계할 수 있는 설계환경을 제공한다. 또한 SystemC는 RT(Register Transfer) 수준보다 높은 추상화수준에서 소프트웨어와 하드웨어 또는 이 둘을 결합한 시스템수준의

* 충북대학교 컴퓨터공학과 ** 충북대학교 반도체공학과
*** 충북대학교 전자정보대학
투고 일자 : 2009. 12. 14 수정완료일자 : 2010. 4. 15
게재확정일자 : 2010. 4. 29
* 이 논문은 2008년도 충북대학교 학술연구지원사업의 연구비 지원에 의하여 연구되었음(This work was supported by the research grant of the Chungbuk National University in 2008).

모델링을 가능케 하며, 타겟시스템을 시스템수준에서부터 설계하기 시작하여 점진적인 설계구체화를 통해 RT 수준까지 설계할 수 있는 단일 언어 및 환경을 제공한다.

시스템수준 설계방법론에서 널리 사용하고 있는 설계흐름도는 시스템명세, 시스템수준의 하드웨어/소프트웨어 분할, 하드웨어/소프트웨어 통합설계, 가상 또는 물리적 프로토타입을 이용한 통합검증, 시스템통합으로 구성된다. 시스템의 하드웨어 구성요소를 개발하는 과정에서 이전까지는 디자인단계가 많은 시간 및 노력을 요구하는 단계였지만, 현재에는 설계한 디자인의 기능적 검증단계가 중요 요소로 간주되고 있다. 기능적 검증의 포괄적인 의미는 DUT(Device Under Test)의 동작을 확인하기 위하여 소프트웨어를 작성 및 실행하는 것이다 [5-7].

최근 하드웨어 구성요소의 기능검증에 프로세싱 코어를 대신하는 BFM(Bus Functional Model)을 이용하는 방법이 주로 사용되고 있다. 또한 최근 발표된 Verilog HDL(Hardware Description Language)의 확장으로 시스템 설계 및 검증언어인 SystemVerilog를 이용한 방법론이 대두되고 있다. 그러나 기존의 HDL은 RT 수준의 하드웨어 구성요소 설계에는 매우 적합하지만, 객체지향프로그래밍 기법의 특징인 상속, 참조, 스레드, 그리고 형변환과 같은 개념을 제공하지 못하기 때문에 상위수준 추상화개념을 이용한 프로그래밍 기법을 처리하기에는 부적합하다.

이에 본 논문에서는 SystemC 구조를 기반으로 한 SystemC 시뮬레이션 커널과 HDL 시뮬레이터로 구성되는 하드웨어 기능검증시스템을 설계 및 구현한다. SystemC 모듈과 Verilog 모듈 간의 통신은 VPI(Verilog Procedural Interface) [8]를 이용하여 본 논문에서 정의하는 SystemC 사용자 정의 통신채널을 통해 이루어진다.

본 논문의 구성은 다음과 같다. 2장에서는 SystemC와 VPI에 대하여 간단히 기술하고, 3장에서는 SystemC 기반의 기능검증시스템 설계에 대하여 기술한다. 4장에서는 설계한 시스템을 UART(Universal Asynchronous Receiver and Transmitter)에 적용하여 기능검증을 수행한 결과를 보이며, 마지막으로 5장에서 결론을 맺는다.

II. Preliminary

2.1 시스템수준 설계언어, SystemC

하드웨어 구성요소와 소프트웨어 프로그램으로 구성되는 시스템의 전형적인 설계방법론에서 하드웨어 구성요소는 HDL로 설계되는 반면에, 소프트웨어 프로그램은 C 또는 C++와 같은 프로그래밍 언어를 이용하여 작성된다. 이와 같이 상이한 언어를 사용하는 heterogeneous 설계환경에서는 시스템설계의 초기단계에서 설계범위탐색을 통한 하드웨어/소프트웨어 분할을 수행하기 어렵다는 문제점이 있다. 하드웨어/소프트웨어 분할은 최종적인 시스템의 구조 및 성능에 영향을 미치는 중요한 단계로, 타겟시스템 설계과정의 초기단계에서 분명하지 않은 하드웨어/소프트웨어 분할은 추후 발행할 수 있는 설계변경에 따른 재설계를 야기한다. 일반적으로 재설계 과정은 비용 및 개발시간 측면에서 많은 양을 소비하게 된다.

시스템수준 설계언어는 하드웨어 구성요소와 소프트웨어 프로그램의 분할에 유연성을 제공한다. 더욱이 시스템은 다양한 기능을 수행하는 하드웨어 구성요소들과 소프트웨어 프로그램이 결합하여 상호동작하기 때문에, 단독으로 기능을 검증할 경우 제한적인 검증만 가능하게 된다 [5-6].

시스템수준 설계언어인 SystemC [1-4]는 시간, 하드웨어 데이터 타입, 동시성, 그리고 계층적 구조 등의 하드웨어적인 개념을 C++ 클래스 라이브러리 형태로 지원하며, 타겟시스템을 다양한 추상화수준에서 설계할 수 있도록 SystemC 구성자를 제공한다. 시스템모델의 다양한 추상화수준을 그림 1에 보인다.

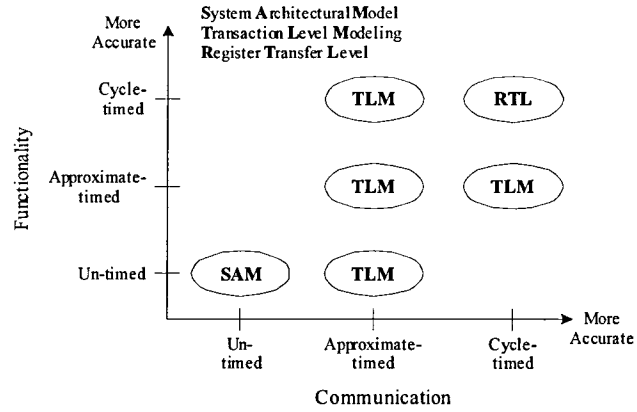

그림 1. 시스템모델의 추상화수준

Fig. 1. Abstraction level of system models

SystemC를 이용한 시스템 설계흐름은 상위수준 추상화 모델에서 하위수준 추상화모델로 해당 구성자를 사용하여 점진적인 설계구체화 과정을 거치게 된다. 또한 시스템설계의 최종 단계가 결국 단위모듈의 통합이라고 볼 때, 시스템 모델링과 사양설정의 초기단계에서부터 구현까지 단일 언어의 설계환경을 갖는다는 것은 SystemC의 큰 장점이라 할 수 있다.

2.2 VPI (Verilog Procedural Interface)

하드웨어 IP(Intellectual Property) 설계흐름은 크게 설계명세, HDL 모델링, 검증, 그리고 시스템통합 및 테스트 단계로 구성되며, 검증단계는 크게 가상 프로토타입을 이용하는 시뮬레이션과 물리적 프로토타입을 이용하는 에뮬레이션으로 구분된다. VPI[8]는 C 프로시저 인터페이스를 통하여 검증단계에서 이용되는 HDL 시뮬레이터와 통신할 수 있는 시스템함수 및 태스크를 정의하는 메커니즘을 제공한다. 시뮬레이션 인터페이스인 Verilog PLI(Programming Language Interface)는 사용자 지정 C 함수가 시뮬레이션 데이터 구조에 실행시(run-time) 접근하여 값을 읽거나 변경할 수 있는 방법을 제공한다. 즉 Verilog PLI는 Verilog 소스 코드와 상호작용하지 않고 다만 시뮬레이션 데이터

구조에만 영향을 끼친다. Verilog PLI는 설계자에게 시뮬레이터의 시스템함수 및 태스크를 정의한 후 사용할 수 있도록 하여 Verilog의 기능을 확장할 수 있는 방법을 제공한다. 사용자 정의 시스템함수 및 태스크의 이름은 '$'로 시작하며, 시스템함수 및 태스크가 호출될 때 시뮬레이터의 실행흐름은 Verilog 서브루틴의 실행흐름과 유사하다. 먼저 VPI로 정의된 시스템함수 및 태스크로 분기하여 기술된 명령어들을 실행한 후, 함수 및 태스크를 호출한 곳으로 되돌아간다.

III. 시스템 수준의 기능 검증시스템 설계

3.1 기능 검증시스템의 구조

본 논문에서 설계하는 SystemC 모듈과 HDL 모듈로 구성된 시스템수준의 하드웨어 기능검증시스템의 구조를 그림 2에 보인다. 순수가상함수를 포함하고 있는 추상화 기초클래스인 generator_ABC, driver_ABC, checker_ABC, 그리고 monitor_ABC를 이용하여 generator, driver, checker, 그리고 monitor 클래스를 각각 유도한 후 해당 인터페이스를 재정의한다. 추상화 기초클래스는 유도클래스의 인터페이스를 명시하며, 추상화 기초클래스의 포인터는 유도클래스의 homogeneous 객체들을 관리하는데 이용된다. 이는 하나의 추상화 기초클래스에서 유도된 서로 다른 클래스들의 객체들을 추상화 기초클래스의 포인터로 가리킬 수 있기 때문이다. 각각의 SystemC 모듈에서 수행하는 동작은 다음과 같다.

- Generator 모듈
  - 임의의 테스트벡터를 생성한 후, 시스템의 기본채널을 통해 Driver 모듈로 전달한다.
- Driver 모듈
  - SystemC 모듈과 HDL 시뮬레이터 간의 인터페이스를 제공하는 통신채널을 통해 Generator 모듈로부터 받은 테스트벡터를 HDL 시뮬레이터로 전달한다.
- Monitor 모듈
  - HDL 시뮬레이터의 출력을 받아 Checker 모듈로 전달한다.
- Checker 모듈
  - DUT의 참조모델인 C 함수의 리턴값과 Monitor 모듈을 통해 전달받은 HDL 시뮬레이터의 출력값을 비교하여 결과를 출력한다.

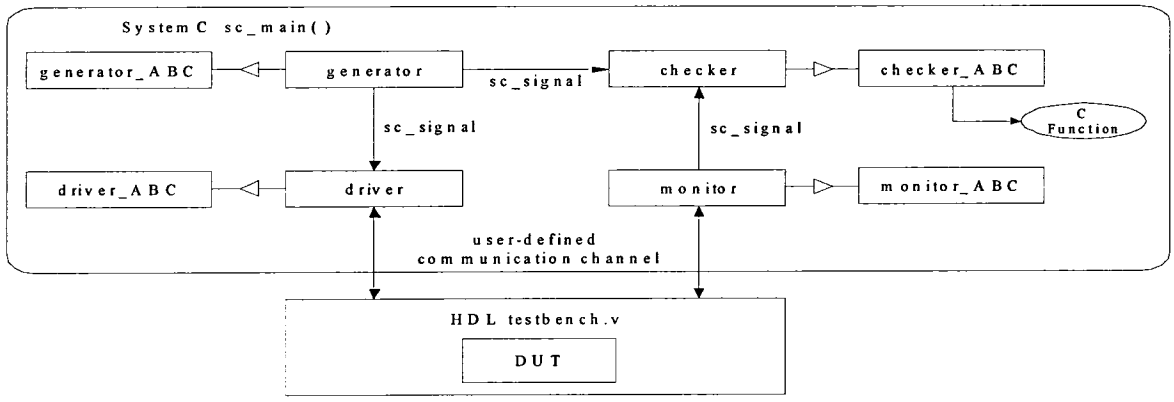

그림 2. 검증시스템의 구조
Fig. 2. Structure of the verification system

Driver 모듈과 Monitor 모듈은 본 논문에서 정의하고 있는 사용자 정의 통신채널의 메소드를 호출하기 위하여 해당 입·출력 인터페이스를 파라미터로 가지는 포트를 포함한다. Driver 모듈은 HDL 모듈과 SystemC 모듈 사이의 연결을 제공하는 사용자 정의 통신채널을 통해 HDL 모듈 내부의 와이어 및 레지스터 변수에 값을 기록하여 DUT의 시뮬레이션을 진행한다. 또한 Monitor 모듈은 사용자 정의 통신채널을 통해 HDL 모듈의 출력값을 읽는다. Checker 모듈은 웨이브 파형을 분석하여 올바른 기능여부 판별에 소비되는 시간을 줄이기 위하여 참조모델인 C 함수를 사용한다.

객체지향프로그래밍 기법의 특징 중에 하나인 상속은 재사용 가능한 코드를 작성할 수 있게 한다. 즉, 본 논문에서 설계하는 SystemC 모듈은 부분적인 코드 수정을 통하여 다른 하드웨어 구성요소를 검증하는데 재사용 가능하다는 이점을 가진다.

3.2 SystemC 함수와 Verilog 시스템함수

VPI는 Verilog 모듈 시뮬레이션에 SystemC 모듈을 결합하기 위하여 사용된다. SystemC 모듈의 최상위 함수인 sc_main()는 VPI를 통하여 HDL 시뮬레이터의 시스템함수로 등록된 후, Verilog의 최상위 모듈인 테스트벤치의 initial 블록 안에서 호출된다. SystemC 시뮬레이션 커널은 시스템함수 호출의 결과로서 동작을 시작한다.

그림 3에 보이는 바와 같이 Verilog 파서가 $로 시작되는 구문을 만나는 경우, Verilog 시뮬레이터는 외부 심볼테이블에서 시스템함수의 이름을 찾아 대응하는 함수를 호출한다. s_vpi_systf_data 구조체 타입의 tfname 멤버변수는

Verilog 파서에 의해 인식되는 토큰을 포함하며, calltf 멤버는 사용자 정의 시스템함수의 함수 포인터를 포함한다. vpi_user.h에 정의된 vpi_register_systf() 함수가 s_vpi_systf_data 타입의 포인터를 전달인자로 가지기 때문에 해당 시스템함수는 Verilog 시뮬레이션 과정에서 호출된다.

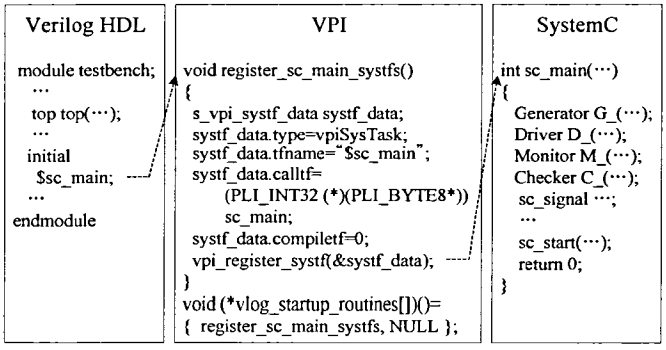

그림 3. 시스템함수의 등록

Fig. 3. System function registration

3.3 통신 채널

시스템설계의 초기단계에서 수행되는 구조적 탐색은 타겟시스템의 기능을 분할한 후, 분할된 모듈의 기능을 구체화하는 과정 및 통신 프로토콜을 구체화하는 과정을 포함한다. 또한 SystemC는 인터페이스, 포트, 그리고 채널과 같은 구성들을 제공하기 때문에 통신 프로토콜의 구체화는 모듈의 기능 구체화와 병렬적으로 이루어진다. 즉, 구조적 탐색에서는 다양한 추상화수준으로 설계된 모듈을 통합하여 하나의 타겟시스템을 구성하는 과정이 수행된다.

Teal [9-10]는 공개된 검증 유틸리티로서 최하위 추상화 수준에서 이용되는 연결 라이브러리이다. Teal 라이브러리 가운데 vreg 클래스는 HDL 시뮬레이터와 C++ 함수 간의 VPI를 통한 연결을 제공하며, 이를 통해 C++ 함수는 Verilog 모듈의 와이어 및 레지스터 변수와 간접적으로 데이터를 전달한다. 그리고 Teal의 at() 함수는 HDL 모듈의 내부 신호가 변할 때까지 스레드로 등록한 C++ 함수의 수행을 중지하는 기능을 한다.

이절에서 설계하는 통신채널은 SystemC 모듈과 HDL 모듈 간의 연결을 제공하는 SystemC 사용자 정의 채널로서 sc_signal<> 및 sc_buffer<>와 같은 기본채널의 평가-갱신 매커니즘을 사용한다. 이 채널은 vreg 클래스의 객체를 멤버변수로 포함하기 때문에 그림 2와 같이 HDL 테스트벤치와 Driver 모듈 및 Monitor 모듈 사이의 연결에 사용된다.

통신채널의 입·출력 인터페이스는 sc_interface 클래스로부터 상속받은 추상화 기초클래스로써 통신채널을 접근하기 위한 메소드를 순수가상함수로 정의하며, 메소드의 구현은 통신채널에서 이루어진다.

통신채널의 인터페이스는 value_changed_event(), default_event(), event(), read(), write(), 그리고 wait_() 등을 포함한다. default_event()는 정적인 센서티브 목록의 사용을 단순화하기 위하여 사용되며, read()와 write()는 멤버변수인 vreg 객체에 대하여 값을 읽고 쓰는 동작으로 HDL 모듈과의 데이터 전달을 위하여 사용된다. write()와 wait_()는 평가-갱신 매커니즘의 평가단계를 포함하며 내부적으로 request_update()를 호출한다. default_event()와 value_changed_event()는 채널을 해당 타입의 센서티브 목록에 등록할 때 사용된다. wait_()는 at()와 request_update()를 차례로 호출하여 vreg 타입의 멤버변수에 연결된 HDL 신호의 변화를 SystemC 통신채널에 동기화시키며, default_event()를 호출하여 반환되는 이벤트 객체를 반환한다.

본 논문에서 정의한 통신채널의 인터페이스를 그림 4에 보인다.

```
#include "systemc.h"
template <class T>
class com_channel_in_if : virtual public sc_interface
{public:
  virtual const sc_event& value_changed_event() const = 0;
  virtual const sc_event& default_event() const = 0;
  virtual bool event() const = 0;
  virtual const T& read() const = 0;
  virtual void wait_() =0;
 protected:
  com_channel_in_if ( ) { }
 private:
  com_channel_in_if( const com_channel_in_if<T>& );
  com_channel_in_if<T>& operator =
         ( const com_channel_in_if<T>& );
}
template <>
class com_channel_in_if<bool> : virtual public sc_interface
{public:
  virtual const sc_event& value_changed_event() const = 0;
  virtual const sc_event& posedge_event() const = 0;
  virtual const sc_event& negedge_event() const = 0;
  virtual const bool& read() const = 0;
  virtual void wait_() =0;
  virtual bool event() const = 0;
  virtual bool posedge() const = 0;
  virtual bool negedge() const = 0;
 protected:
  com_channel_in_if ( )  { }
 private:
  com_channel_in_if( const com_channel_in_if<bool>& );
  com_channel_in_if<bool>& operator =
         ( const  com_channel_in_if<bool>& );
}
template <class T>
class com_channel_inout_if : public com_channel_in_if<T>
{ public:
   virtual void write( const T& ) = 0;
 protected:
  com_channel_inout_if()  {  }
 private:
  com_channel_inout_if( const com_channel_inout_if<T>& );
  com_channel_inout_if<T>& operator =
         ( const com_channel_inout_if<T>& );
}
#define com_channel_out_if com_channel_inout_if
```

그림 4. 통신채널의 인터페이스

Fig. 4. Interface of the communication channel

Ⅳ. UART의 기능검증

이절에서는 설계한 시스템을 DUT인 UART에 적용하여 기능검증을 수행한다. UART는 컴퓨터의 직렬통신을 위한 서브시스템으로 바이트 크기의 데이터를 연속적인 비트로 전송하는 기능을 수행한다 [11]. UART와 기능검증시스템의 연결 구성을 그림 5에 보인다.

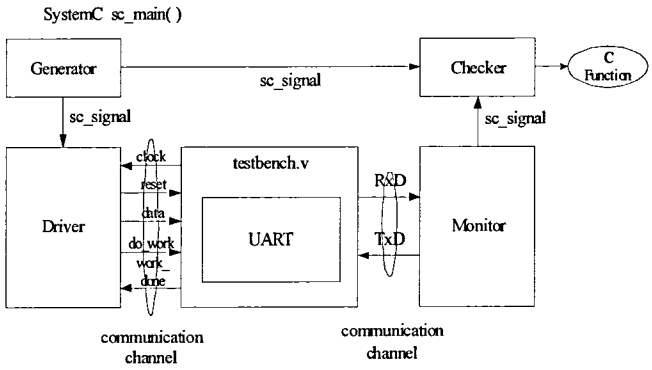

그림 5. 검증시스템의 모듈 연결

Fig. 5. Module connection of the verification system

Generator 모듈은 임의의 테스트벡터를 생성한 후 Driver 모듈과 Checker 모듈로 sc_signal 채널을 통해 전송한다. 만약 테스터벡터가 사용자 데이터 타입인 경우 할당연산자인 operator=()와 비교연산자인 operator==()를 타입의 메소드로 정의한 후 사용하여야 한다. Driver 모듈과 Monitor 모듈은 사용자 정의 통신 채널을 통해 HDL 모듈의 변수에 값을 쓰고 읽으며, Checker 모듈은 참조모델인 C 함수의 리턴값을 DUT의 결과값과 비교하여 올바른 동작여부를 판별한다. UART의 기능검증을 수행한 결과를 그림 6에 보인다.

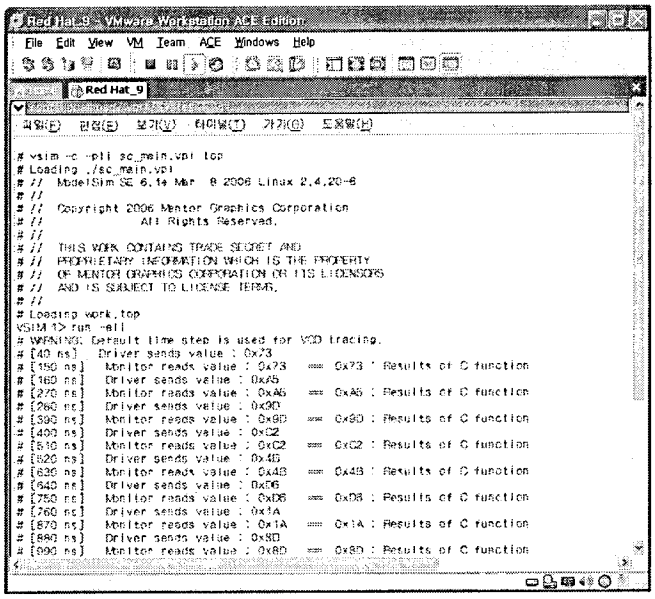

그림 6. 기능검증의 결과

Fig. 6. Results of the functional verification

Ⅴ. 결 론

본 논문에서는 시스템수준 설계언어인 SystemC를 이용한 하드웨어 기능검증시스템을 설계하였다. SystemC 코드가 VPI를 통하여 HDL 시뮬레이터의 시스템함수로 등록된 후 하드웨어 모듈의 내부 신호에 접근할 수 있도록 하드웨어 모듈을 Verilog HDL로 설계한다. SystemC 시뮬레이션 커널과 HDL 시뮬레이터 사이의 연결은 SystemC의 사용자 정의 통신채널 및 HDL 시뮬레이터의 사용자 정의 시스템함수를 통해 이루어진다.

본 논문에서는 다양한 추상화수준에서 모델링할 수 있는 SystemC를 이용하여 설계흐름의 초기단계에서 설계범위탐색을 통한 시스템수준의 하드웨어/소프트웨어 분할을 수행한 후, 점진적인 설계구체화 과정을 수행한다. 이는 시스템 모델링과 사양설정의 초기단계에서부터 구현까지 단일 언어의 설계환경을 갖는다는 이점이 있다. 또한 기능검증시스템 설계에 작성된 SystemC 모듈 코드들은 부분적인 코드 수정 후 다른 하드웨어 구성요소의 기능을 검증하는데 재사용할 수 있는 이점이 있다.

참 고 문 헌

[1] David C. Black, Jack Donovan, *SystemC:From the Ground Up*, pp.12-23, Eklectic Ally, Inc., 2004.

[2] Thorsten Grotker, Stan Liao, Grant Martin, Stuart Swan, *System Design with SystemC*, pp.51-57, Kluwer Academic Publishers, 2002.

[3] 유명근, 송기용, "C-to-SystemC 합성기의 설계 및 구현," 신호처리·시스템 학회 논문지, 제10권, 2호, pp.141-145, 2009.

[4] *SystemC Language Reference Manual*, http://www.systemc.org

[5] Jason R. Andrews, *Co-Verification of Hardware and Software for ARM SoC Design*, pp.119-129. Elsevier Inc., 2005.

[6] Ando Ki, *SoC Design and Verification: Methodologies and Environments*, pp.2-8, Hongreung Science, 2008.

[7] 유명근, 송기용, "SystemVerilog와 SystemC 기반의 통합검증환경 설계 및 구현",신호처리·시스템 학회 논문지, 제10권, 4호, pp.274-279, 2009.

[8] Stuart Sutherland, *The Verilog PLI Handbook : A Tutorial and Reference Manual on the Verilog Programming Language Interface*, pp.27-54, Kluwer Academic Publishers, 2002.

[9] Mike Mintz, Robert Ekendahl, *Hardware Verification with C++:A Practitioner's Handbook*, pp.67-88, Springer, 2006

[10] *The Teal User's Manual*, http://www.trusster.com

[11] Wikipedia, *Universal Asynchronous Receiver Transmitter*, http://en.wikipedia.org/wiki/UART.

유 명 근(Myoung-keun You)

2003년 충북대 컴퓨터공학과 학사
2006년 충북대 대학원 컴퓨터공학과 석사
2010년 충북대 대학원 컴퓨터공학과 박사
관심분야 : 임베디드시스템 설계 · 검증, 상위수준 설계 · 검증

오 영 진(Young-jin Oh)

2005년 충북대 컴퓨터공학과 학사
2007년 충북대 대학원 컴퓨터공학과 석사
2008년~현재 충북대 대학원 반도체공학과 박사과정
관심분야 : SoC 설계 · 검증, 상위수준 설계 · 검증

송 기 용(Gi-yong Song)

1978년 서울대 공교과 학사
1980년 서울대 대학원 전자과 석사
1995년 미 루이지애나대 박사
1983 ~ 현재 충북대 전자정보대학 교수
관심분야 : SoC 설계 · 검증, 상위수준설계, C++ 기반 하드웨어 검증