FPGA 컴파일 회피에 의한 효과적인 시뮬레이션 가속

심 규 호[†] · 박 창 호[††] · 양 세 양[†††]

요 약

본 논문에서는 FPGA 기반의 시뮬레이션가속을 통한 함수적 검증에서 매 설계오류의 수정 과정에서 필수적으로 진행되어야 하였던 긴 FPGA 컴파일 시간에 의한 오랜 디버깅턴어라운드시간을 단축할 수 있는 FPGA 컴파일 회피를 통한 효과적인 시뮬레이션가속 방법을 제시하였다. 마이크로컨트롤러 설계의 함수적 검증에 제안된 방법을 적용한 결과, 본 논문에서 제안된 방법이 시뮬레이션가속의 높은 시뮬레이션 수행 속도를 유지하는 동시에 디버깅턴어라운드시간도 크게 단축할 수 있음을 확인할 수 있었다.

키워드 : 검증, 시뮬레이션, 시뮬레이션가속, 현장프로그램가능게이트어레이

Efficient Simulation Acceleration by FPGA Compilation Avoidance

Kyuho Shim[†] · Changho Park[††] · Seiyang Yang[†††]

ABSTRACT

In this paper, we proposed an efficient FPGA-based simulation acceleration method based on FPGA compilation avoidance, which can effectively decrease the long debugging turnaround time incurred from the every debugging process in the functional verification. The proposed method had been experimentally applied to the functional verification for a microcontroller design. It had clearly shown that the debugging turnaround time was greatly reduced while the high simulation speed of the simulation acceleration was still maintained.

Key Words : Verification, Simulation, Simulation acceleration, FPGA

1. 서 론

현재 반도체 칩의 집적도 증가에 따라서 설계복잡도도 크게 증가하고 있다. 특히 사용자 특정 하드웨어뿐만 아니라 범용 프로세서 코어, 메모리, DSP 코어들이 동시에 존재하는 SoC와 같은 시스템급 설계가 일반화되고 있는 현실에서 함수적 검증 대상의 크기와 복잡도는 지속적으로 증가하고 있다. 더욱이 시스템 설계 회사들의 극심한 경쟁과 시장의 요구에 의해 설계/제조 비용을 최대한으로 낮추면서 해당 SoC 칩의 신속한 출시 요구는 함수적 검증의 중요성을 아주 중요한 문제로 대두시키고 있다. 현재와 같은 설계 복잡도는 이미 전체 설계과정에서 함수적 검증을 위하여 전체 설계시간의 50%-70%를 투입하여야만 하는 검증위기 상황을 초래하고 있다. 따라서 검증 생산성을 높일 수 있는 효과적인 방법론이 제시되지 않는다면 가까운 미래에 검증을 위하여 투입되는 비용과 시간은 더욱 크게 증가할 것으로 예상되고 있다[1][15].

기존의 함수적 검증 방법들로는 시뮬레이션, 프로토타이핑, 에뮬레이션, 시뮬레이션 가속 등이 있다. 특히 과거에서부터 함수적 검증에 보편적으로 제일 많이 사용되고 있는 방법은 시뮬레이션이었다. 시뮬레이션은 사용의 편이성, 유연성, 강력한 디버깅 지원, 타이밍 검증 능력 등의 매우 중요한 장점들을 지니고 있다. 하지만 최근의 수백만 내지는 수천만 게이트급 이상의 시스템 수준 함수적 검증에서는 시뮬레이션의 속도가 크게 떨어짐으로 인하여 SoC와 같은 대규모 설계의 함수적 검증을 시뮬레이션에만 의존하는 것은 현실적으로 불가능하다[2]. 이런 시뮬레이션의 단점인 느린 속도를 보완하고저 제안된 방법이 시뮬레이션가속이다[3]. 특히 최근들어서 집적도가 매우 높은 FPGA의 출현으로 인하여 FPGA 기반의 시뮬레이션가속기들이 함수적 검증에 많이 사용되고 있다. 그러나, 이와 같은 FPGA 기반의 시뮬레이션가속기들은 논리합성 시간 및 배치/배선의 FPGA 컴파일에 많은 시간을 요하게 됨으로서 시뮬레이션가속을 통한 함수적 검증에서 일어나게 되는 많은 횟수의 디버깅에서 많은 시간을 FPGA 컴파일에 소비하게 되는 문제점이 있다. 본 논문에서는 이와 같은 FPGA 기반의 시뮬레이션가속을 이용한 함수적 검증에서 디버깅 시간을 단축할 수 있는 기법을 제안하고, 이의 유용성을 실험을 통하여 확인한다.

※ 본 논문은 부산대학교 자유과제 학술연구비(2년)에 의하여 연구되었음
†준 회 원 : 부산대학교 대학원 컴퓨터공학과 석박사통합과정 수료
††정 회 원 : 동양시스템즈 IT서비스운영팀
†††정 회 원 : 부산대학교 컴퓨터공학과 교수
논문접수 : 2007년 1월 17일, 심사완료 : 2007년 4월 5일

시뮬레이션가속은 설계 대상의 복잡도가 수백만 게이트에 이르면서 시뮬레이션의 속도를 높이기 위해 소프트웨어 기반의 시뮬레이션의 다음 단계로 개발된 검증기법으로, 합성 가능한 DUT(Design Under Test)을 별도의 하드웨어검증플랫폼(예로, FPGA 어레이)에 구현하여 최대한 병렬적으로 동작될 수 있도록 하고, TB(Testbench)는 HDL(Hardware Description Language) 시뮬레이터에서 실행하여 DUT와 TB를 연동시켜서 동작시킴으로 시뮬레이션의 속도를 높이는 것이다[3]. 이러한 시뮬레이션가속 기법은 HDL 시뮬레이터만을 이용하는 순수 시뮬레이션에 비해 트란젝션 수준의 가속에서는 최대 1,000-10,000배까지의 실행속도 향상을 보여주며, 순수 시뮬레이션과 매우 유사한 환경에서 함수적 검증을 수행할 수 있는 장점이 있다[3, 4, 9].

특히 최근에 들어서는 집적도가 매우 높은 FPGA들의 등장으로 인하여 1 이상의 FPGA를 하드웨어검증플랫폼으로 사용하여서 시뮬레이션가속을 수행할 수 있는 다양한 하드웨어 에뮬레이터들이 SOC 내지는 ASIC 설계에 사용되고 있다. 그러나, 시뮬레이션가속을 위하여 FPGA에 DUT를 구현하기 위하여 FPGA 논리합성, FPGA 배치 및 배선, FPGA 비트파일 생성, 다운로딩(이와 같은 전체 과정을 본 논문에서는 FPGA 컴파일이라고 정의함)의 과정을 수행하여야만 한다. 특히, 이 중에서 FPGA 배치 및 배선과 FPGA 논리합성은 매우 시간이 오래 걸리는 과정이다.

현재 레지스터전송수준(RTL)에서 이루어지는 함수적 검증에서 시뮬레이션가속 기법을 적용하는 것이 매우 일반적인 상황인데, 함수적 검증의 초기 단계에서는 매우 많은 수의 설계오류들이 존재함으로 인하여 시뮬레이션가속 기법을 통한 함수적 검증 과정은 시뮬레이션가속 수행을 통한 설계오류의 발견과 이를 수정하는 디버깅에 의한 설계의 수정, 그리고 수정된 설계를 대상으로 하는 또 다른 시뮬레이션가속 수행(본 논문에서는 하나의 설계오류를 수정하기 위하여 소비된 시뮬레이션 실행시간과 컴파일 시간의 합을 디버깅턴어라운드시간이라 정의하기로 하는데, 이 디버깅턴어라운드시간은 1번 이상의 시뮬레이션가속 실행시간과 1번 이상의 FPGA 컴파일 시간을 합친것임)이 최소한 수십번 반복적으로 진행되게 된다. 이와 같은 상황에서 수정된 설계를 대상으로 하는 또 다른 시뮬레이션가속 수행을 위해서는 수정된 설계를 FPGA에 구현하기 위하여 FPGA 컴파일이 필요하게 되며(이와 같은 설계 수정 이후에 수행되는 FPGA 컴파일을 본 논문에서는 FPGA 재컴파일이라고 칭함), 이와 같은 FPGA 재컴파일은 디버깅을 통한 설계 수정이 이루어질 때마다 반복적으로 수행되어져야 함으로 시뮬레이션가속의 경우에는 디버깅턴어라운드시간에서 FPGA 컴파일 시간이 차지하는 비중이 상대적으로 매우 높아지게 되고, 디버깅턴어라운드시간의 단축을 크게 제약하게 된다.

특히, 설계 오류들의 숫자가 상대적으로 많은 이와 같은 함수적 검증의 초기 단계에서부터 시뮬레이션가속 기법을 적용하게 되면 경우에 매 설계 오류의 발견과 수정때마다 장시간의 FPGA 컴파일을 필요하게 됨으로서 시뮬레이션가속을 이용한 검증 생산성 향상에 매우 큰 제약 요소가 됨을 알 수 있다.

2. 컴파일 회피를 통한 효율적 시뮬레이션가속

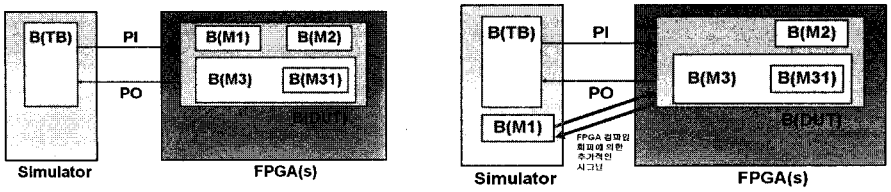
본 논문에서는 대규모 회로에서 검증 속도가 느린 순수 HDL 시뮬레이션의 단점과 FPGA 기반 시뮬레이션가속에서의 긴 FPGA 컴파일 시간의 단점을 보완하여 설계 오류의 수정 후에 변경된 DUT 내지는 DUT 내의 일부분을 재구현하기 위한 FPGA 재컴파일을 일정 횟수만큼 회피함으로서 높은 검증 속도와 짧은 디버깅턴어라운드시간을 동시에 얻을 수 있는 FPGA 컴파일 회피(compilation avoidance)가 가능한 시뮬레이션가속 방법을 제안한다.

2.1. 시뮬레이션가속에서의 FPGA 컴파일 회피

시뮬레이션 내지는 시뮬레이션가속을 통한 함수적 검증에 소요되는 전체 시간을 줄이기 위해서는 디버깅턴어라운드시간을 단축시키는 것이 매우 중요하다. 새로운 방법을 통하여 디버깅턴어라운드시간을 단축시키기 위해서는 실행시간을 줄이거나, 컴파일시간을 줄이는 것이 필요하다.

HDL 시뮬레이터만을 이용하는 순수 시뮬레이션이 디버깅턴어라운드시간을 단축시키는 것에 결정적으로 어려운 이유는 HDL 시뮬레이터의 컴파일시간은 매우 빠르지만 실행시간이 매우 오래 걸리기 때문이다. 반면에, FPGA 기반의 시뮬레이션가속에서는 시뮬레이션가속의 실행시간은 짧지만 FPGA 컴파일시간이 매우 오래 걸리게 된다. 본 논문에서 제안하는 FPGA 컴파일 회피 방법은 FPGA를 이용하는 시뮬레이션가속을 통하여 실행시간을 단축시키는 동시에, 디버깅을 통한 설계 변경 이후에 시뮬레이션가속의 재실행 단계에서 FPGA 재컴파일을 수행하지 않는 대신에, 설계가 변경된 DUT 내의 일부분만을 TB와 함께 HDL 시뮬레이터 상에서 수행시킴으로서(따라서, FPGA 컴파일 회피에서는 특정 설계오류의 수정 후에 오랜시간이 소요되는 FPGA 재컴파일 대신에 매우 짧은 시간만이 필요한 HDL 시뮬레이터 재컴파일 과정을 수행하게 됨) 시뮬레이션가속을 통한 실행시간 단축과 FPGA 컴파일 시간 단축을 동시에 가능하도록 하여서 디버깅턴어라운드시간을 단축시킬 수 있게 한다.

(그림 1)은 이와 같은 본 논문에서의 FPGA 컴파일 회피를 이용하는 시뮬레이션가속의 상황을 개략적으로 보여주고 있는 예이다. (그림 1(a))는 DUT 전체가 하드웨어검증플랫폼인 FPGA에 구현되어서 통상적인 방식의 시뮬레이션가속으로 수행되어지는 상황을 보여주고 있다. 그림 1(b)는 그림 1(a)의 통상적인 방식의 시뮬레이션가속 실행의 결과로 설계오류가 존재하는 DUT 내의 한 특정 설계블럭 B(M1)이 디버깅 과정을 통하여 설계오류가 수정 되어진 후에 FPGA 재컴파일 과정 대신에 HDL 시뮬레이션 재컴파일 과정을 통하여 시뮬레이터 영역으로 이동하여서 TB와 같이 수행되어지는 FPGA 컴파일 회피 과정을 통한 시뮬레이션가속 상황을 보여주고 있다. FPGA 상에서 DUT의 다른 설계블록들과 함께 수행되던 특정 설계블럭이 설계 오류가 수정되어 HDL 시뮬레이터로 이전되어서도 DUT의 다른 설계블럭들과 같이 수행되어지기 위해서는 이들 다른 설계블럭들과 이



(a) FPGA 컴파일 회피 (b) FPGA 컴파일 회피 후

(그림 1) FPGA 컴파일 회피의 개략적 개념

특정 설계블럭 간의 포트연결이 FPGA 재컴파일 없이도 동적으로 변할 수 있어야만 한다. 이와 같은 상황은 (그림 1(a))에서 다른 설계블럭들과 특정 설계블럭 B(M1)의 FPGA 내부에서의 포트연결이 (그림 1(b))에서는 특정 설계블럭 B(M1)과 다른 설계블럭들간의 FPGA와 시뮬레이션 사이에 새롭게 존재하는 연결(그림 1(b))에서 시뮬레이터와 FPGA 간에 추가적인 사선)로 변화되어져 보여지고 있다. 여기서 다시 한번 강조할 것은 FPGA 컴파일 회피가 수행되면 FPGA에 대한 재컴파일은 수행되지 않는 대신에 시뮬레이션 재컴파일이 수행된다는 것인데, 이와 같은 것을 가능하게 하는 것이 BSC(Block-boundary Scan Cell)(BSC에 대한 자세한 설명은 3.3.2를 참조) 구조이다.

이와 같은 특정 설계블럭이 FPGA 컴파일 회피를 통하여 FPGA에서 HDL 시뮬레이터로 동적으로 이동되더라도 DUT 내의 다른 설계블럭들과의 포트연결이 가능하도록 하기 위하여 DUT 내의 모든 설계블럭들의 입출력 포트에는 BSC(3.3.2를 참조)을 추가시키는 것이 필요하다. 본 논문에서 FPGA 컴파일 회피를 가능하게 하기 위하여 사용된 BSC 구조는 IEEE-1149.1의 바운더리스캔 구조[10]와 매우 유사하다. 단, 바운더리스캔 구조는 바운더리스캔을 지원하기 위한 핀수를 최소화시키는 것이 매우 중요함으로 데이터입력(TDI)과 데이터출력(TDO)으로 각각 1비트만을 사용하는데 반하여서, BSC는 FPGA 컴파일 회피 과정 중에서 진행되는 시뮬레이션가속의 속도를 최대한도록 높게 유지시키기 위해서 데이터입력과 데이터출력을 각각 16비트로 할당하여서 스캔인과 스캔아웃이 빠른 속도에서 이루어질 수 있도록 고려하였다. 또한 DUT의 계층구조에서 DUT의 바로 하위계층에 존재하는 설계블럭들만을 대상으로 BSC를 부가(instrument)함으로 BSC에 의한 면적 오버헤드가 커지지 않도록 하는 동시에, 임의의 특정 설계블럭에 설계오류가 존재하더라도 항상 FPGA 컴파일회피가 가능하도록 고려 하였다.

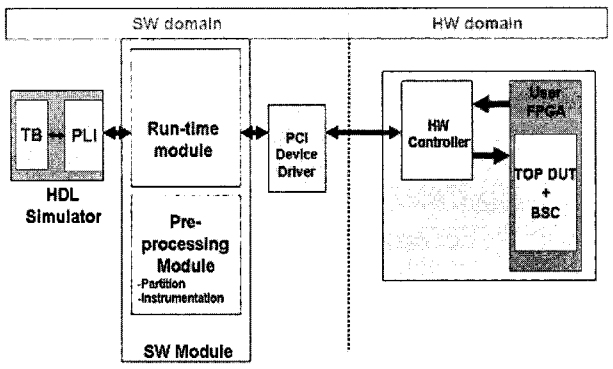
(그림 2)는 본 논문에서 제안하는 시뮬레이션가속을 위한 시스템 전체 구성도이다. 전체 시스템은 소프트웨어 도메인과 하드웨어 도메인으로 나누어진다. 소프트웨어 도메인의 HDL 시뮬레이터에서 실행되는 테스트벤치(TB)와 하드웨어 도메인의 사용자 FPGA에서 실행되는 DUT와의 연동을 위하여 소프트웨어 도메인에는 S/W 모듈, 하드웨어 도메인에는 H/W 콘트롤러를 두며, PCI 디바이스 드라이버[6, 7]를 통해 하드웨어 도메인과 소프트웨어 도메인 사이의 데이터 교환이 이루어진다. 구성요소들을 좀 더 자세히 설명하면 다음과 같다.

1. HDL 시뮬레이터: PLI(Programmable Language Interface)[8, 9]를 통해 테스트벤치로부터 DUT로 보내어질 DUT 입력값을 얻어오고, DUT 출력값을 받는 테스트벤치와 수정될 Verilog DUT 하위 모듈이 실행되는 블록이다.

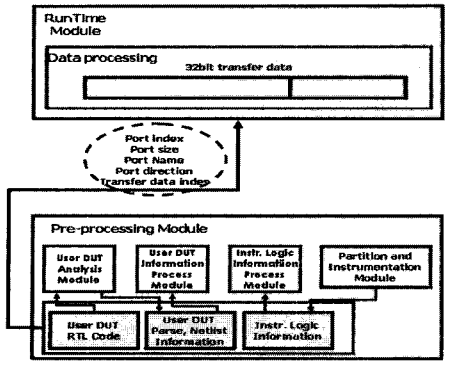
2. S/W 모듈: HDL 시뮬레이터에서 PLI를 통해 읽어온 Verilog 테스트벤치와 수정된 Verilog DUT 하위블록의 출력, 입출력 포트 값을 32 비트로 조합하여 PCI 디바이스 드라이버로 넘겨주고, FPGA로부터 나온 DUT의 출력값을 PCI 디바이스 드라이버를 통해 읽어와서 이를 분해한 후 테스트벤치와 수정된 Verilog DUT 하위 모듈에 값을 할당하는 블록이다.

3. PCI 디바이스 드라이버: S/W 모듈과 FPGA 사이의 데이터를 읽고 쓰는 블록이다.

4. FPGA 보드: FPGA 보드는 하드웨어검증플랫폼의 역할을 수행한다. 이 FPGA 보드에는 시뮬레이션가속에 필요한 HW 컨트롤러가 구현되고, 함수적 검증 대상이 되는 DUT와 더불어 FPGA 컴파일 회피를 가능하게 하는 BSC도 같이 구현되어진다. FPGA 컴파일 회피 하에서의 동작은 PCI 디바이스 드라이버를 통해 소프트웨서 도메인에서 넘어온 데이터를 BSC를 이용하여 FPGA에 있는 DUT에 전송하고, 다시 이 BSC를 통해 FPGA의 DUT의 출력 데이터를 S/W 모듈로 보내기 위해 PCI 디바이스 드라이버로 전송한다. 이와 같은 동작을 통하여 설계오류 수정에 의하여 HDL 시뮬레이션에서 수행되어지는 설계오류가 수정된 DUT 내의 특정 설계블럭의 입출력 포트와 계속적으로 FPGA에 남아있는 DUT 내의 나머지 설계블럭들의 입출력 포트들 상에서의 논리값 교환이 FPGA 컴파일 회피 이후에도 올바르게 이루어지게 된다.



(그림 2) 시뮬레이션가속 시스템 전체 구성도



(그림 3) 런타임 모듈과 전처리 모듈

2.2. HDL 시뮬레이터와 S/W 모듈 사이의 인터페이스

HDL 시뮬레이터와 S/W 모듈 사이의 인터페이스는 PLI (Programming Language Interface)를 통하여 이루어진다. PLI는 IEEE 1364-1995에 정의되어 있는 C 프로시져 인터페이스 표준으로 Verilog HDL 디자인의 계층적인 구조를 탐색할 수 있고, Verilog HDL 디자인 내에 있는 객체(object)들의 정보를 동적으로 접근하고 수정 할 수 있다. 또한 시뮬레이션 환경에 대한 정보도 가져올 수 있는 C 언어 함수 라이브러리를 제공한다[13].

2.3. S/W 모듈과 FPGA와의 인터페이스

본 절에서는 S/W 모듈에 있는 런타임 모듈(런타임 모듈)과 전처리 모듈(Pre-Processing Module)에 대해서 설명하고, 이 S/W 모듈과 FPGA와의 인터페이스에 대하여 논한다.

2.3.1. 런타임 모듈과 전처리 모듈

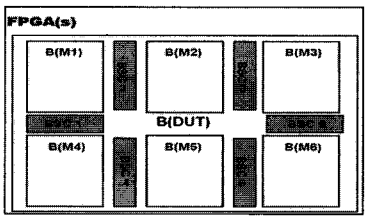
런타임 모듈은 시뮬레이션가속 실행에서 TB가 수행되는 HDL 시뮬레이터와 DUT가 수행되는 FPGA의 연동을 제어하는 기능을 수행하게 된다. 전처리 모듈은 실제 시뮬레이션가속 실행에 앞서서 수행되어야 하는 모든 과정들을 처리하게 되는데, 여기에는 일반적인 시뮬레이션가속에서도 필요한 DUT Verilog 코드와 TB Verilog 코드를 읽어들여서 파싱하고 DUT를 구현하는 FPGA가 2 이상의 경우에는 분할을 수행하는 것 이외에 추가적으로 본 논문에서 디버깅턴어라운드시간을 단축시키기 위하여 제안한 FPGA 컴파일 회피를 가능하도록 DUT의 하위계층으로 존재하는 각 블록별로 BSC를 추가(BSC 로직 인스트루멘테이션)하는 과정을 포함하게 된다.

런타임 모듈은 소프트웨어 도메인과 하드웨어 도메인 사이에서 전송되는 데이터 구성을 위해 Verilog HDL 디자인의 포트정보 및 전송데이터의 정보들이 필요하게 되는데 이러한 정보들을 전처리 모듈로부터 제공받는다. 전처리 모듈은 Verilog 파서(parser)를 이용해서 Verilog HDL 디자인(DUT와 TB)의 정보들을 얻으며, FPGA 상에 DUT와 같이 다운로드 되는 BSC 구성을 위한 정보를 얻는다. 위의 (그림 3)과 같이 소프트웨어 도메인에서 하드웨어 도메인으로 데이터가 전송될 때 런타임 모듈은 전처리 모듈이 제공하는 정보들을 이용하여 32 비트 전송데이터를 구성하며, 그 반대의 경우인 하드웨어 도메인으로부터 넘어온 32 비트 전송데이터를 소프트웨어 도메인에 있는 HDL 시뮬레이터에 할당 할 때 Verilog HDL 디자인의 포트 정보에 맞게 재구성 한다.

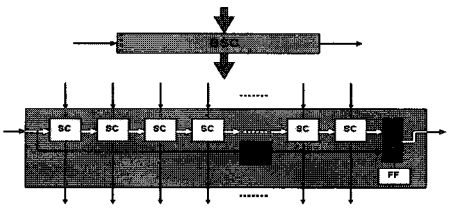
2.3.2. FPGA 컴파일 회피를 위한 BSC 구조

본 논문에서 제안하는 FPGA 컴파일 회피를 통하여 DUT내에 존재하는 설계오류를 수정한 이후에 시뮬레이션 가속의 재실행을 위하여 FPGA 재컴파일을 수행하지 않고서도 올바르게 시뮬레이션가속이 실행되기 위해서는 FPGA에서 계속 실행되는 DUT내의 설계변경이 이루어지지 않은 설계블록들과 설계오류가 존재하여 수정이 이루어진 후에 시뮬레이션으로 수행되는 설계오류가 존재하였던 설계블록 사이에 올바른 신호값들의 주고 받음이 시뮬레이션가속의 재실행 과정에서 이루어질 수 있도록 DUT 내의 각 설계블럭들의 입출력에 BSC 로직을 부가시키는 것이 필요하다.

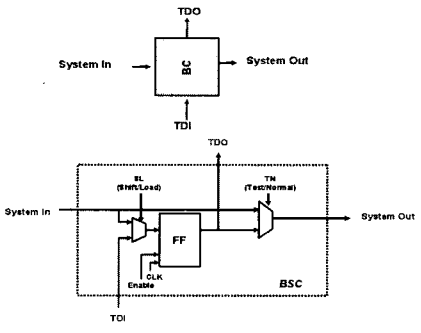
(그림 4)는 이와 같은 역할을 수행하는 BSC 로직의 구현을 보여주고 있다. 그림 4에서부터도 알수있는 것과 같이 BSC 로직의 구조는 IEEE-1149.1의 바운더리스캔 방식과 유사하다. 이와 같은 BSC 로직을 부가함으로 인하여 해당 설계블럭이 FPGA 내부에 존재하더라도 이 설계블럭의 입력에 대한 제어도(controllability)와 출력에 대한 관측도(observability)가 확보되어짐으로서 FPGA 컴파일 회피를 통하여 FPGA에 구현되어져 존재하다가 HDL 시뮬레이터 상에서 수행되어지는 특정 설계블럭과의 연결이 계속적으로 가능하게 된다.



(a) DUT에 BSC가 부가된 상황

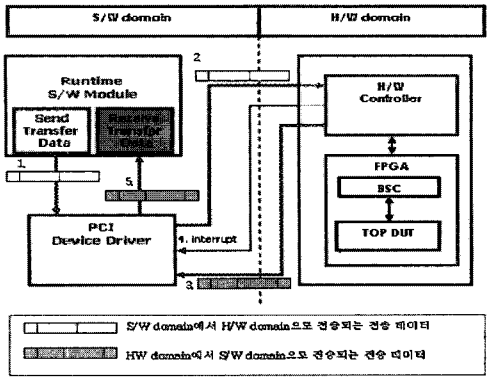


(b) BSC의 내부구조



(C) SC의 내부구조

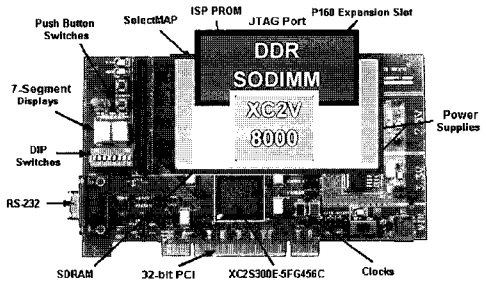
(그림 4) BSC의 구조



(그림 5) 전송데이터 전송과정

2.3.3. 런타임 모듈과 H/W 콘트롤러의 인터페이스

런타임 모듈과 H/W 콘트롤러와의 연동은 PCI 디바이스 드라이버를 통해서 이루어진다. 런타임 모듈은 하드웨어 도메인으로 보낼 전송데이터를 생성하고 이를 PCI 디바이스 드라이버를 통해서 H/W 콘트롤러로 전송하고, FPGA에서 실행되는 DUT로부터 출력값이 얻어질 때까지 기다리게 된다. 뿐만 아니라 FPGA 컴파일 회피가 이루어진 경우에는 H/W 콘트롤러는 BSC를 이용하여 DUT로 추가적인 입력 데이터를 인가하게 되고 추가적인 출력 데이터도 얻어오게 된다.



(그림 6) 실험에 사용된 PCI 기반의 FPGA 보드

3. 실험 및 분석

FPGA 기반의 시뮬레이션가속을 통한 함수적 검증에서 DUT에 대한 설계오류의 수정 후에도 본 논문에서 제시한 FPGA 컴파일 회피를 통한 시뮬레이션가속을 FPGA 재컴파일없이 진행하는 실험을 수행하였다. 실험 환경에서 사용된 환경 구성은 다음과 같다. 운영 체제는 리눅스(Linux Redhat9 커널 버전 2.4.20-8), 시뮬레이터는 Model Technology사의 Modelsim 5.7 Linux 버전이다.[11], 그리고 하드웨어 도메인과 소프트웨어 도메인 사이의 인터페이스를 위한 PCI 디바이스 드라이버는 리눅스용 디바이스 드라이버를 자체 구현하여 환경을 구성하였고 시뮬레이션 가속을 위한 PCI 기반의 FPGA 보드(board)[12](그림 6)를 사용하였으며, 소프트웨어 도메인과 하드웨어 도메인에서 교환되는 데이터 대역폭은 32 비트이다. FPGA에 Verilog HDL 디자인을 다운로드 하기 위하여 합성(synthesis)에 사용된 툴은 Synplify Pro 7.2[13]가 사용되었으며, 배치 및 배선과정(Place & Route)에 사용된 툴은 Xilinx ISE 6[14]이 사용되었다. 마지막으로 검증 대상 디자인으로는 8비트 마이크로 컨트롤러인 PIC 마이크로컨트롤러를 기반으로 하는 벤치마크 회로를 자체적으로 설계하여서 사용하였다.

실험은 벤치마크 회로를 사용하여서 순수한 시뮬레이션, FPGA 컴파일 회피가 없는 전통적인 시뮬레이션가속, FPGA 컴파일 회피를 이용한 시뮬레이션가속의 세가지 시뮬레이션 상황에 대하여 시뮬레이션 수행시간(S), 컴파일 시간(C), 디버깅턴어라운드시간(T) (T = S + C)을 측정하였다.

<표 1>에 실험결과가 보여지는 것과 같이, 전통적인 시뮬레이션가속의 경우에 시뮬레이션의 속도는 순수한 시뮬레이션에 비하여 649% 향상되었으나, 전통적인 시뮬레이션가속의 경우에 설계오류가 수정되어진 후에 재실행의 경우에 오랜시간의 FPGA 컴파일 시간이 필요함으로 인하여서 전통적인 시뮬레이션가속의 디버깅턴어라운드 시간은 순수한 시뮬레이션에 비하여서 오직 20%만이 단축됨을 알 수 있다. 이에 비하여 FPGA 컴파일 회피를 이용한 시뮬레이션가속의 경우에는 설계오류가 수정된 설계블럭은 HDL 시뮬레이터에서 수행되어짐으로 인하여 오랜 시간의 FPGA 컴파일 대신에 매우 빠른 시간에 수행된(이 실험의 경우에 9초) HDL 시뮬레이션 컴파일 시간만이 필요함으로 디버깅턴어라운드 시간이 순수한 시뮬레이션에 비하여서는 610%가, 전통적인 시뮬레이션가속에 비하여서는 493%가 단축됨을 알 수 있다. 물론, FPGA 컴파일 회피를 수행하는 경우에 HDL 시뮬레이터에 의하여 수행되어지는 부분이 증가하고 시뮬레이션가속의 하드웨어검증플랫폼과 HDL 시뮬레이터 간의 통신 대역폭도 증가함으로 인하여 시뮬레이션의 속도가 전통적인 시뮬레이션가속에 비하여 6% 감소(그러나, 순수한 시뮬레이션에 비해서는 FPGA 컴파일 회피를 수행한 경우에도 610% 성능향상을 계속 유지함)됨을 알 수 있으나, 이와 같은 시뮬레이션 속도의 미세한 저하는 디버깅턴어라운드 시간의 대폭적인 단축에 비해서 매우 미미함으로 본 실험을 통하여 FPGA 컴파일 회피의 유용성을 충분히 확인할 수 있다. 또한 BSC를 DUT 계층구조에서 DUT 바로 하위계층에 존재하는 설계블럭들에만 부가함으로 늘어나는 면적 오버헤드는 약 10%로 측정되었는데, 이와 같은 10% 정도의 면적 증가는 우려할만한 수준이 아니다.

〈표 1〉 실험결과

시뮬레이션 기술	시뮬레이션 수행시간		컴파일 시간 (초)	디버깅 턴어라운드 시간(초)
	1,000,000 시뮬레이션 사이클 수행시간(초)	시뮬레이션 성능 (cycles/sec)		
순수 시뮬레이션	4,090	244	53	4,143
FPGA 컴파일 회피가 없는 전통적인 시뮬레이션가속	630	1,587	2,720	3,350
FPGA 컴파일 회피를 이용한 시뮬레이션가속	670	1,493	9	679

물론, FPGA 컴파일 회피 기법에 의하여 어느 정도만큼 디버깅턴어라운드 시간을 단축할 수 있을지는 설계 사례들마다 다를 수 있다. 그러나 일반적으로 최근의 FPGA 집적도의 향상에 의하여 백만게이트급의 설계를 단일 FPGA에 구현가능하게 됨으로 인해서 FPGA 컴파일 시간은 이에 비례하여 계속적으로 증가하는 상황과 최근들어서 시뮬레이션가속의 속도가 트란젝션기반의 시뮬레이션가속 방식을 통하여 순수 시뮬레이션에 비하여 100-1,000배이상 빨라지는 추세 때문에 FPGA 기반의 시뮬레이션가속의 경우에서 설계 오류의 수정에 의한 FPGA 재컴파일로 초래되는 긴 디버깅턴어라운드 시간은 앞으로도 계속적으로 문제가 될 것임을 알 수 있다. 따라서 이와 같은 상황에서 본 논문에서 제안되는 FPGA 컴파일 회피를 통한 디버깅턴어라운드 시간의 단축은 매우 유용함을 알 수 있다.

4. 결론 및 향후 과제

본 논문에서는 FPGA 기반의 시뮬레이션가속을 통한 함수적 검증에서 매 설계오류의 수정 과정에서 필수적으로 진행되어야 하였던 긴 FPGA 컴파일 시간에 의한 오랜 디버깅턴어라운드시간의 단축을 가능하게 하는 FPGA 컴파일 회피를 통한 효과적인 시뮬레이션가속 방법을 제시하였다.

이와 같은 함수적 검증 방법은 소프트웨어 기반의 검증 방법과 하드웨어 기반의 검증 방법의 장점들을 취하게 되어 HDL 시뮬레이터만을 사용하는 것에 비하여 고속의 검증 실행속도와 FPGA 기반의 시뮬레이션가속에서의 오랜 컴파일 시간도 함께 단축시킬 수 있는 장점들을 동시에 가질 수 있는데, 이와 같은 사실은 마이크로콘트롤러 설계에서의 함수적 검증에 본 논문에서 제안된 방법을 적용하여서 실험적으로도 확인할 수 있었다.

앞으로 계속 연구해야할 연구과제로는 HDL 시뮬레이터의 포트 정보를 얻기 위한 PLI 루틴의 오버헤드와 소프트웨어 도메인과 하드웨어 도메인의 연동을 위한 PCI 디바이스 드라이버에서의 전송데이터 교환에 소요되는 오버헤드를 최소화하여 본 논문에서 제안한 시스템의 전체 성능을 더욱 향상시킬 수 있는 연구가 필요할 것이다.

참 고 문 헌

[1] Richard Foster, "A Design Style to Simplify IP integration and Verification," White paper, VLSI Technology, Inc. (http://www.vlsi.com), 1999.

[2] N. Kim, H. Choi, S. Lee, S. Lee, I. Park, and C. Kyung, "Virtual Chip: Making Functional Models Work on Real Target Systems," in Proc. of 35th DAC, pp.170-173, June 1998.

[3] "Cadence Emulation Simulation Acceleration", White paper, Cadence Design Systems Inc. (http://www.cadence.com), 2002.

[4] Murali Kudlugi, Soha Hassoun, Charles Selvidge, Duaine Pryor, "A Transaction-Based Unified Simulation/ Emulation Architecture for Functional Verification", in Proc. of 38th DAC, pp.623-628, June 2001.

[5] Stuart Swan, "SystemC Transaction Level Models and RTL Verification", in Proc. of 43rd DAC, pp.90-92, July 2006.

[6] Daniel P. Bovet, Marco Cesati, Understanding the Linux Kernel, O'REILLY Publishers, December 2002

[7] Alessandro Rubini, Jonathan Corbet, Linux Device Drivers, O'REILLY Publishers, June 2001.

[8] IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language, IEEE Std. 1364-1995.

[9] Swapnajit Mittra, Principles of Verilog PLI, Kluwer Academic Publishers, 2000.

[10] JTAG Boundary Scan, IEEE Std. 1149.1.

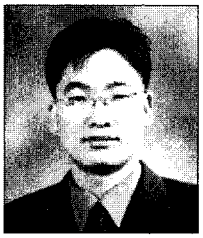
[11] ModelSim SE User's Manual, Mentor Graphics (http://www.model.com), 2006.

[12] Spartan-II 200™ PCI Development Board User's Guide V2.0, Memec (http://www.memec.com), 2002.

[13] Synplify Pro User Guide, Synplicity Inc. (http://www.synplicity.com), 2005.

[14] Xilinx ISE 6 Software Manuals, Xilinx Inc. (http://www.xilinx.com), 2004.

[15] 김남도, 양세양, "초고속 시스템 에뮬레이터의 구조와 이를 위한 소프트웨어", 한국정보처리학회논문지 A, 1598-2831, 제8A권4호, pp.479-488, 2001.



심 규 호

e-mail : capnemo@pnu.kr

2003년 부산대학교 정보컴퓨터공학부 학사

2006년 부산대학교 대학원 컴퓨터공학과 석박사통합과정 수료

관심분야 : SoC 검증

박 창 호

e-mail : zest7196@nate.com

2004년 부산대학교 정보컴퓨터공학부 학사

2006년 부산대학교 대학원 컴퓨터공학과 공학석사

2007년 현 재 동양시스템즈 IT서비스운영팀

관심분야 : SoC 검증

양 세 양

e-mail : syyang@pusan.ac.kr

1981년 고려대학교 전자공학과 학사

1985년 고려대학교 전자공학과 석사

1990년 University of Massachusetts, 전기컴퓨터공학과 박사

1991년~현 재 부산대학교 컴퓨터공학과 재직

관심분야 : SOC 검증