12

웹기반 SystemC 통합 설계/디버깅 CAD 시스템, Scenian

(주)시스템 센트로이드 ▌박인학
한국항공대학교 ▌나종화

1. 서 론

시스템칩(Soc, System-On-a-Chip)의 설계 방법론은 두 방향으로 진화하고 있다. 하나는 반도체 IP 재사용에 의한 플랫폼 기반 설계방법론이고, 다른 하나는 SystemC 언어를 이용한 ESL(Electronic System Level) HW/SW 통합 설계방법론이다. 전자는 이미 여러 기업체에서 구축한 플랫폼이 성공사례로 제시될 정도로 보편화 되고 있다. 후자의 경우는 ESL에서 설계할 수 있는 표준언어로 SystemC가 2005년 12월에 IEEE1666 표준으로 선택되었고 아직도 계속 진화하고 있다. 하지만 C/C++ 기반 하드웨어 모델링, TLM(Transaction Level Modeling), 임베디드 소프트웨어 통합 설계 등의 무한한 가능성을 갖고 있기 때문에 SystemC 언어의 중요성은 날로 높아질 것이다[1].

시스템칩 설계에 사용되는 여러가지 표준 언어들을 추상화 수준에 따라 구분하면 그림 1과 같다. 가로축은 설계 언어의 종류이고, 세로축은 설계 데이터의 추상화 수준이라 할 수 있다. UML(Unified Modeling Language)과 SDL(System Description Language)이 최상위 수준의 시스템 설계 언어라면, VHDL과 Verilog HDL 언어가 RTL(Register Transfer Level) 수준에서 가장 보편적으로 사용되는 언어이다. C/C++ 언어는 하드웨어 설계의 개념이 전혀 없는 순수한 프로그래밍 언어로 사용되고 있다. 현재 System Verilog와 SystemC 언어가 ESL 수준 설계에 새로운 언어로 대두되고 있다. 그러나 System Verilog 언어는 검증에 강점이 있는 반면 SystemC 언어는 HW/SW 혼합 설계에 강점이 있기 때문에 임베디드 소프트웨어의 비중이 점차 높아가는 시스템칩 설계에 중요한 위치를 차지하리라 예상된다[2,3].

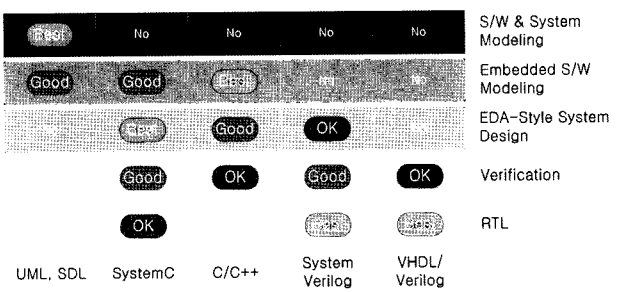


그림 1 설계언어로서 SystemC 언어의 포지셔닝

최근 미국 기업들이 주도하여 다양한 SystemC 언어 기반의 다양한 툴들이 EDA 시장에 출시되고 있다. 대표적으로 SystemC 코드에서 시스템칩 회로를 자동 생성하는 합성 툴로서는 Forte Design Systems사의 Cynthesizer와 Synopsys사의 CoCentric이 있다. SystemC 시뮬레이션을 지원하는 툴로는 Mentor Graphics사의 ModelSim, Veritools사의 SuperC, Cadence사의 SPW 등이 있다. 그밖에 SystemC 코드와 HDL 코드간의 자동 변환 툴을 지원하는 Adelante Technologies사의 AIRT 등이 있다. SystemC와 관련된 CAD 툴을 개발하는 국내 업체는 다이나릿스(주)와 (주)시스템 센트로이드가 유일하다. 다이나릿스(주)의 iPROVE는 SystemC 코드와 iPROVE에 탑재된 HDL 코드가 연동하여 시뮬레이션이 되는 검증 환경이다. 반면 (주)시스템 센트로이드의 Scenian은 SystemC 코드로 설계하고 디버깅 및 시뮬레이션 검증을 지원하는 순수한 소프트웨어 개발 환경이다.

2. 웹기반 CAD 시스템

Scenian은 웹기반 CAD 시스템으로서 인터넷을 통해 실시간으로 데이터를 주고 받으면서 서로 통신하는 서버와 클라이언트 툴로 구성된다(그림 2). 클라이언트 툴은 설계자가 직접 SystemC 코드를 작성하고 디버깅하고 시뮬레이션 결과를 검증할 수 있는 그래픽 환경을 제공한다. 반면 서버에는 고가의 설계자원이 설치되어 있고 클라이언트에서 작업하는 설계자들이 인터넷을 통하여 이를 공유하여 사용할 수 있는 기능을 제공한다[4]. 설계자원이란 시뮬레이션 툴과 IP 라이브러리와 고성능 컴퓨터 등을 의미한다. 클라이언트의 그래픽 환경은 궁극적으로 서버의 설계자원을 활용하기 위한 창구 역할을 한다.

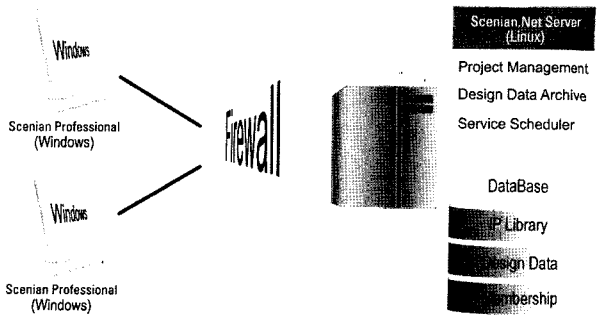


그림 2 웹기반 CAD 시스템의 구조

Scenian이 웹기반 동작으로 구성된 근본 원인은 설계자의 전문지식이 집약된 IP의 설계 데이터가 외부에 노출되지 않도록 보안하면서 동시에 여러 설계자들이 용이하게 공유할 수 있는 모델을 구현하기 위함이다. 웹기반 CAD 시스템은 모든 설계자원을 원격 서버에 설치하고 일반 설계자들은 인터넷에 연결된 자신의 컴퓨터에서 설계를 진행하면서 서버에 설치된 설계 자원을 자신의 것과 같이 활용할 수 있다. 즉, 하드웨어나 소프트웨어를 구매할 필요가 없으며 전산실 운영도 필요없고 시공간의 제약도 받지 않는다. 서버에서 지원하는 모든 서비스를 인터넷에 연결된 컴퓨터만 있으면 언제 어디서든지 즉시 공급받을 수 있다. 단 하나의 단점이라면 인터넷을 통한 통신 속도에 따라 대기 시간이 늘어나는 것인데 인터넷 통신 대역폭이 증가함에 따라 대기시간은 줄어들게 된다.

설계자가 자신의 컴퓨터에서 SystemC 언어로 설계를 한 후 원격 서버에서 서비스를 받아 결과를 얻기까지의 프로세스가 그림 3에 나타나있다. 번개 모양의 그림은 인터넷을 통한 통신을 의미한다. 가운데를 중심으로 좌측은 클라이언트에서 수행되는 작업이고 우측은 원격 서버에서 수행되는 작업이다.

IP의 실제 데이터(Real IP)는 원격 서버에 라이브러리로 구축되어 있다. 서버의 IP들을 재사용하기 위해 IP의 인터페이스 데이터(IP Shell)을 다운로드 받아 클라이언트에 설치한다. 설계자는 클라이언트 그래픽 툴들의 도움을 받아 원격 서버에 설치된 IP들을 재사용한 SystemC 코드를 설계한다. SystemC 설계가 완성되면 원격 서버에 시뮬레이션 검증 서비스를 요청한다. 서버는 여러 대의 클라이언트에서 요청되는 서비스를 모두 받아 작업큐에 등록시킨 후 차례대로 처리하게 된다. 시뮬레이션시 IP의 실제 데이터가 설계자의 설계에 링크되어 처리된다. 클라이언트에서 인터페이스 데이터만으로 설계된 IP의 내부 설계가 서버에서 검증될 때 완성됨을 의미한다. 시뮬레이션이 수행되면서 발생하는 로그는 실시간으로 클라이언트에 나타나며, 시뮬레이션 결과 파형은 클라이언트에 다운로드 받아 검증하게 된다.

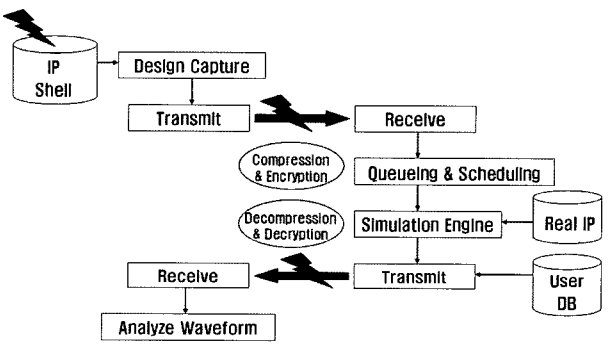


그림 3 원격 검증의 흐름

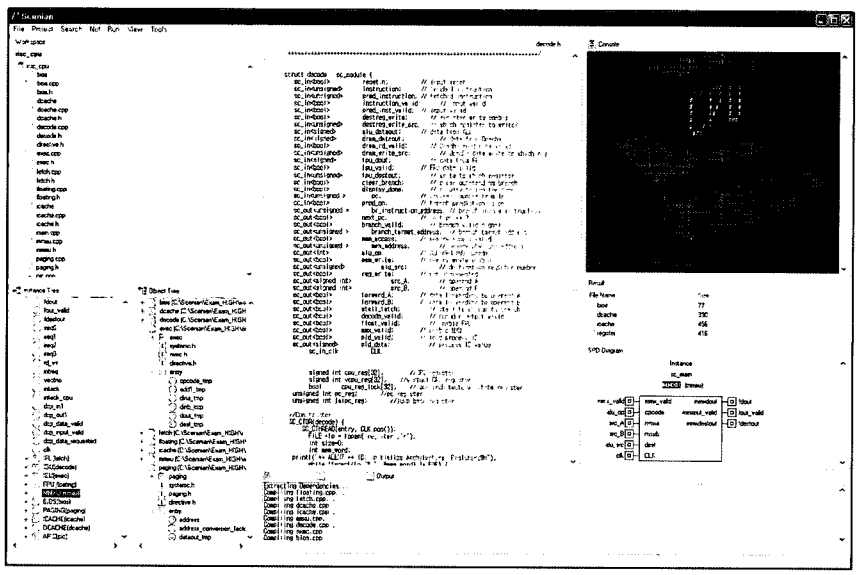


그림 4 클라이언트 화면

3. SystemC 코드 획득 및 디버깅

클라이언트는 SystemC 코드를 작성고 하드웨어와 소프트웨어 디버깅을 돕는 다양한 그래픽 툴들로 이루어진다(그림 4). 기본적으로 전체 설계환경은 Workspace로 정의되고, 회로를 구성하는 SystemC 파일들은 Project에 등록된다. 설계자는 SystemC 전용 텍스트 편집창에 코드를 수작업으로 입력하는데 코드에 정의된 하드웨어 회로를 분석할 수 있는 다양한 도우미 기능이 제공된다. 내장된 파서는 SystemC 파일들을 분석하여 추출한 하드웨어의 계층구조와 시그널 정보를 바탕으로 여러 가지 디버깅 기능들을 만들어낸다.

Scenian은 SystemC 코드의 하드웨어와 소프트웨어 디버깅을 구분하여 지원한다[5]. SystemC 언어가 하드웨어 구조도 정의할 수 있지만 회로의 알고리즘 동작도 동시에 기술할 수 있기 때문이다. 하드웨어 디버깅을 돕는 가능들을 요약하면 다음과 같다.

- 컴파일 오류 메시지 : SystemC 코드에 직접 관련된 컴파일 메시지만으로 간략화
- Instance Tree와 Object Tree : 회로의 구성 및 계층구조를 그래픽 트리로 표현
- SPD : 단자 및 시그널간의 상관관계를 그래픽으로 표현하고 순/역방향 추적 지원
- 아키텍쳐 생성 : 하드웨어 전체 구조를 논리회로도로 생성하며 계층구조 탐색 지원
- 파형 분석 : 시뮬레이션 파형값을 Instance Tree, SPD 및 SystemC 코드에 동시 표현

소프트웨어 디버깅은 일반적인 일반 프로그램 개발 환경과 동일한 기능을 제공한다. 실제로는 원격 서버에 설치된 GNU 디버거가 실행되지만 설계자는 클라이언트의 그래픽 환경에서 SystemC 코드의 디버깅을 실행한다. 소프트웨어 디버깅을 돕는 기능들은 다음과 같다.

```
13
14  and2 *DUT;
15  DUT = new and2("DUT");
16  DUT->a(a);
17  DUT->b(b);
18  DUT->z(z);
19
20  and2_tb *STM;
21  STM = new and2_tb("STM");
22  STM->a(a);
23  STM->b(b);
24  STM->stmstep(stmstep);
```

그림 6 소프트웨어 디버깅 화면

- Breakpoint 기능 : 실행을 정지시키고 프로그램을 한 라인씩 순차적으로 진행
- Watch 기능 : SystemC 코드에 선언된 변수의 현재값을 살필 수 있는 기능
- Call Back 기능 : 중단된 코드의 위치를 procedure의 계층구조로 표현

SystemC 코드를 편리하고 쉽게 획득할 수 있도록 돕는 다양한 그래픽 툴들이 내장되어 있다. 논리회로도 편집기는 아키텍쳐 설계를 돕는 그래픽 툴로서 IP 라이브러리나 설계자 모듈을 사용하여 계층구조의 논리회로도를 설계할 수 있다. 상태도 편집기는 Mealy나 Moore 타입의 상태도를 설계하는 툴이고, 테스트 파형 편집기는 DUT(Design Under Test) 모듈에 테스트 파형을 가하는 테스트벤치 모듈을 정의한다. 그래픽 설계를 그대로 모델링하는 SystemC 코드를 자동으로 생성될 수 있다.

4. 원격 서비스

SystemC 시뮬레이션은 기본적으로 C/C++ 프로그램의 컴파일, 빌드 및 실행이다. 이 기능은 IP 라이브러리와 연동이 되어야 하기 때문에 서버에 설치된 GNU Compiler가 서비스로 제공한다.

- SystemC 컴파일 서비스 : SystemC 코드를 컴파일하여 프로그램 오류를 보고
- SystemC 빌드 서비스 : IP 라이브러리와 접속하여 실행 파일을 생성

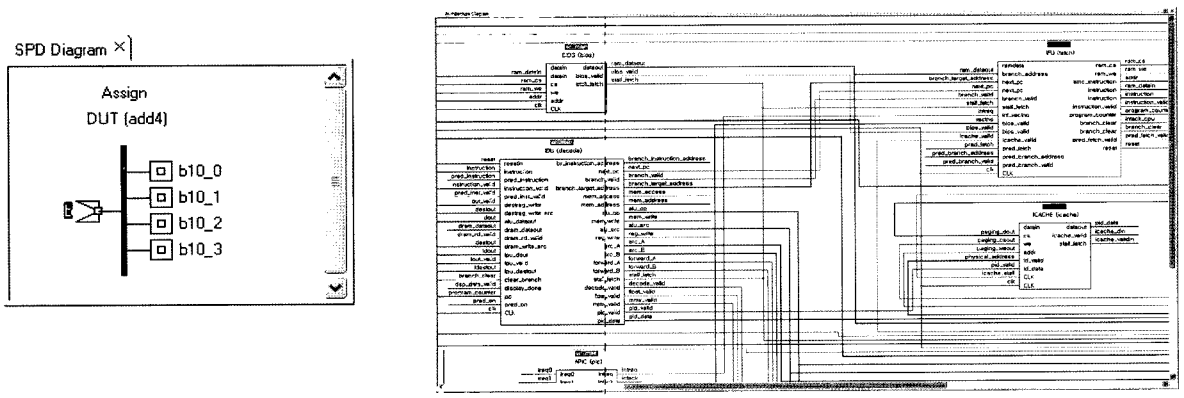
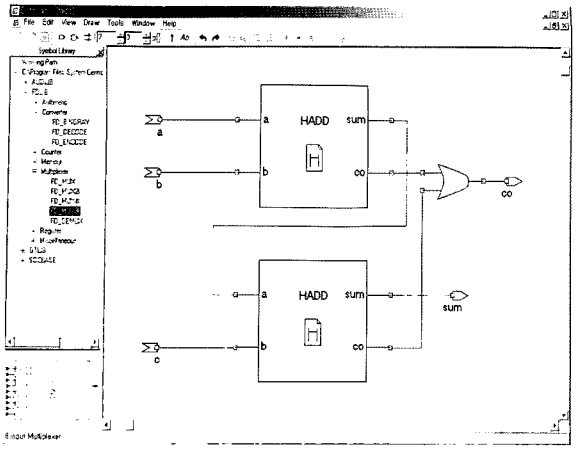
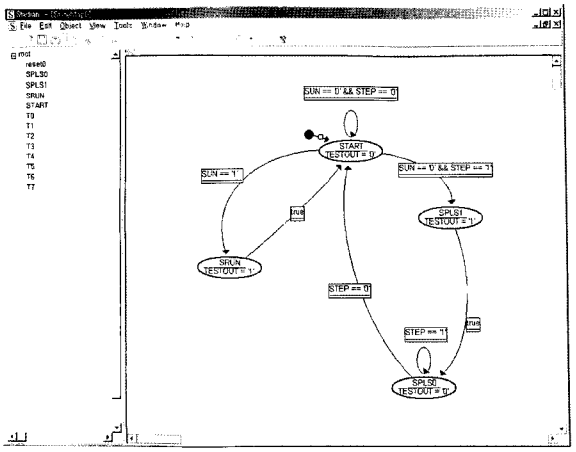


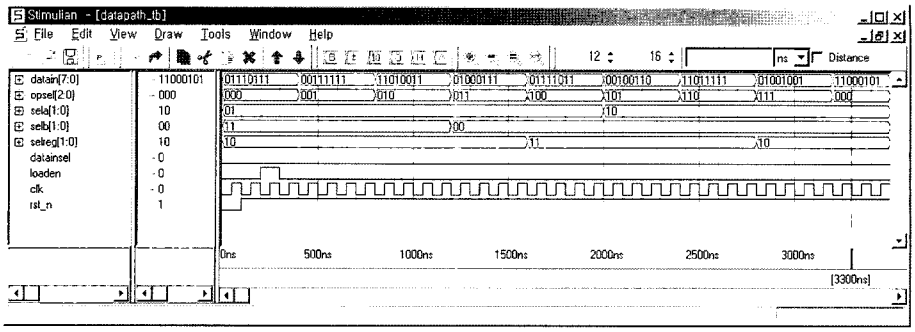
그림 5 하드웨어 디버깅을 위한 SPD와 아키텍처 화면



(a) 논리회로도 편집기



(b) 상태도 편집기



(c) 테스트 파형 편집기

그림 7 SystemC 코드 자동 생성그래픽 툴들

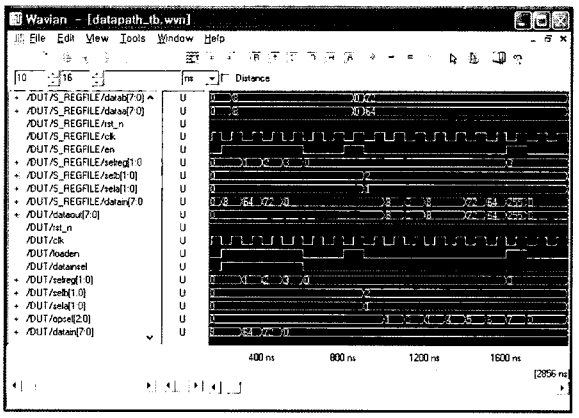
- SystemC 실행 서비스 : 실행 파일을 실행 (로그 관찰 및 출력 파일 획득)

클라이언트에서 설계된 SystemC 코드와 설계 데이터는 원격 서버로 전송되어 컴파일과 빌드와 실행이 수행되는데 이 과정에서 발생하는 로그는 실시간으로 클라이언트 로그창에 출력된다. 실행파일이 생성한 파일들은 클라이언트로 다운로드 받을 수 있으며 시뮬레이션 파형은 파형분석기 그래픽 툴에서 확인할 수 있다(그림 8).

5. Fault Tolerant System 설계 및 검증

최근 Deep Submicron 기술의 보편화 및 가속적인 직접화로 인하여 프로세서 및 SoC 시스템의 신뢰성 문제가 제기되고 있다. 이런 문제의 해결 방안으로서 Design for Testability, Design for Manufacturability에서 진일보한 Design for Reliability가 이슈화 되고 있다. 프로세서의 신뢰도(Dependability)는 하드웨어 수준에서는 HDL 기반 Fault injection 환경이 연구 개발되어 우주 및 항공전자 분야 등에서 많이 사용이 되

(a) 시뮬레이션 실행 로그 창



(b) 시뮬레이션 파형 분석기

그림 8 원격 서비스

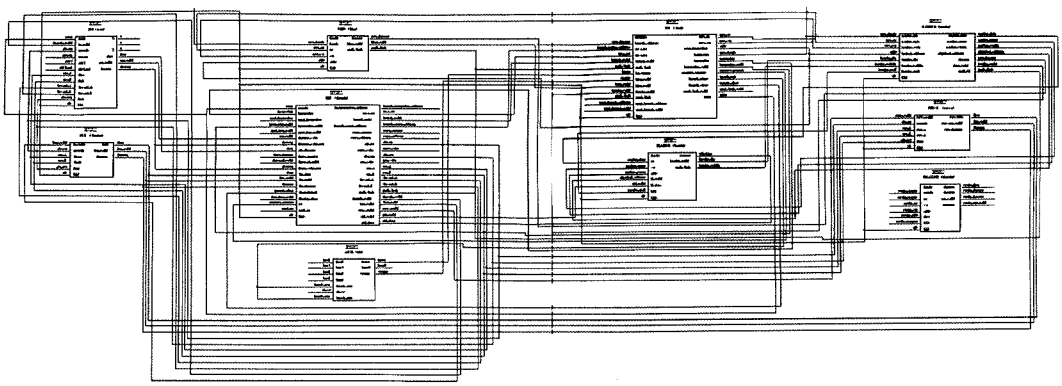


그림 9 신뢰성 검증을 위한 프로세서 모델

고 있다[6,7]. 그러나 SoC 및 MPSoC의 Reliability는 시스템의 복잡도의 증가로 인하여 전체적인 시스템의 Dependability의 검증이 용이하지 않다. 이런 경우에는 SystemC SoC모델을 개발하고, SystemC를 이용한 Fault Injection 기반 신뢰도 평가 환경을 개발하여 개발하려는 목표 시스템의 신뢰도를 평가할 수 있다. 평가의 내용은 시뮬레이션을 통해 시스템의 안정성에 문제가 되는 critical spot를 찾거나, 시스템의 run-time 성능과 신뢰도 사이의 연관관계를 분석하는 것이 가능하게 된다.

일반적으로 Fault Injection 시뮬레이션을 이용하여 목표 신뢰도를 평가하는 방법은 크게 3가지가 있다. 첫 번째는 Simulator commands 기반 실험기법으로서 시뮬레이션 실험 중 직접 특정 신호에 오류를 삽입하는 방법으로 모델을 수정하지 않고서도 간단하게 검증 할 수 있다는 장점이 있으나, simulator에서 해당되는 명령어를 사용해야 하므로 수많은 실험을 수행해야 하기는 적합하지 못하다. 두 번째는 Saboteurs 방법을 들 수 있다. 이것은 시뮬레이션 모델에 오류주입 실험을 위한 모듈을 부가적으로 추가하여 특정 신호 또는 변수 값을 원하는 시간에 수정하는 것이다. 마지막으로 Mutants 기법이 있다. 이것은 기존 모델의 각각의 컴포넌트 이외에 오류를 내포하고 있는 대체가능한 컴포넌트들을 준비하여, 오류주입실험을 실행할 때마다 오류를 포함하고 있는 모델을 이용하여 실험을 수행한다. 본 논문에서는 간단한 RISC 프로세서의 신뢰성을 SystemC Saboture 모듈과 Mutants 모델을 이용하는 오류주입실험환경을 개발하여 신뢰성을 평가하는 데모를 수행하였다. 위의 그림 9에는 Scenian의 Analyze 도구를 이용하여 평가대상 모델을 아키텍쳐를 그림으로 표현하였다.

그림 10은 simple cpu모델의 실행 후 특정시간(42ns)에 register의 값을 출력한 결과이다. 출력 결과에 왼쪽은 fault가 존재하지 않는 Golden Run의 실행 결과이며, 오른쪽은 RISC 프로세서의 레지스터 중의 하나인 R0에 fault가 발생되어 졌을 때의 결과이다. fault injection을 하기위해 fault_time, fault_location, fault_mask를 설정하고 saboture 모듈을 통해 오류를 삽입하였다. 여기서 fault time은 10ns이며, fault_location은 R0이고, fault_mask는 0xffffffff로 설정하였다. 이와 같이 오류가 삽입되었을 경우 2가지 경우를 예상할 수 있다. 먼저 프로그램이 R0을 읽어가는 경우 오동작 또는 잘못된 결과를 산출 할 것을 예상 할 수 있다. 반면 R0에 데이터를 쓸 경우 삽입되어진 오류는 무시될 수도 있다. 이처럼 오류삽입실험을 통해 예상되는 시스템의 변화를 관찰 할 수 있으며 이를 통해 설계된 시스템의 신뢰성을 평가 및 예측 할 수 있다. 이러한 결과를 바탕으로 Design for Reliability의 구

```
**********************************
ID: REGISTERS DUMP at CSIM 42 ns
**********************************
REG :=============================================================
 R 0(00000000)  R 1(00000001)  R 2(ff000e2)  R 3(ffffffff)
 R 4(00000004)  R 5(00000005)  R 6(0000000a)  R 7(fcf0fdef)
 R 8(00000008)  R 9(00000009)  R10(00000010)  R11(0000ff31)
 R12(0000ff12)  R13(00000013)  R14(00000014)  R15(00000015)
 R16(00000016)  R17(00fe0117)  R18(00fe0118)  R19(00fe0119)
 R20(00fe0220)  R21(00fe0321)  R22(00fe0322)  R23(00ff0423)
 R24(00ff0524)  R25(00ff0625)  R26(00ff0726)  R27(00ff0727)
 R28(00f70728)  R29(00000029)  R30(00000030)  R31(00000031)
=============================================================
```

Golden data

```
**********************************
ID: REGISTERS DUMP at CSIM 42 ns
**********************************
REG :=============================================================
 R 0(ffffffff)  R 1(00000001)  R 2(ff000e2)  R 3(ffffffff)
 R 4(00000004)  R 5(00000005)  R 6(0000000a)  R 7(fcf0fdef)
 R 8(00000008)  R 9(00000009)  R10(00000010)  R11(0000ff31)
 R12(0000ff12)  R13(00000013)  R14(00000014)  R15(00000015)
 R16(00000016)  R17(00fe0117)  R18(00fe0118)  R19(00fe0119)
 R20(00fe0220)  R21(00fe0321)  R22(00fe0322)  R23(00ff0423)
 R24(00ff0524)  R25(00ff0625)  R26(00ff0726)  R27(00ff0727)
 R28(00f70728)  R29(00000029)  R30(00000030)  R31(00000031)
=============================================================
```

Fault injection

그림 10 RISC 프로세서 오류삽입 실험 수행 결과

현 즉, 신뢰성 있는 시스템 설계가 가능할 수 있다.

5. 결 론

SystemC 언어는 오래전부터 설계자들이 알고리즘 검증을 위하여 사용해온 C/C++ 언어에 하드웨어 구조를 기술할 수 있는 라이브러리와 시뮬레이션 커널이 추가되어 탄생하였다. 그러므로 전통적인 하드웨어 설계자의 설계문화와 조화를 이루면서도 하드웨어와 소프트웨어의 통합 설계 및 검증 문제를 해결한다. 특히 시스템칩 설계에서 하드웨어 설계가 완료되어야 소프트웨어 검증을 시작할 수 있던 고전적인 설계방식을 탈피하여 하드웨어와 소프트웨어를 동시에 설계하고 검증하면서 최적화를 찾을 수 있는 설계방법론을 가능케 했다는 점에서 다른 설계언어와 차별되는 장점을 갖는다.

Scenian은 원격 서버에 감추어진 IP 라이브러리를 재사용하면서 SystemC 언어를 기반의 회로를 설계하고 디버깅하고 검증할 수 있는 웹기반 CAD 시스템이다. SystemC 코드를 파싱하여 여러 가지 하드웨어 및 소프트웨어 디버깅 도우미를 제공하는 것도 Scenian의 편리한 기능이다. 특히, 원격 서버에 설치된 고가의 설계자원을 다수의 설계자가 언제 어디서든지 공유할 수 설계 환경을 제공할 수 있는 것은 Scenian만의 독창적인 특징이다.

참고문헌

[1] W. Muller, W. Rosenstiel, J. Ruf, "SystemC Methodologies and Applications", KAP, 2003.

[2] 기안도, "SystemC 시스템 모델링 언어", 2005.

[3] 데이빗 C. 블랙, 잭 도노반, (국일호 역), "원리부터 배우는 SystemC", 2007.

[4] Inhag Park, Yongjoo Kim, "Flowrian, an Internet CAD System for Design Reuse of Silicon IP", APSOC 2002, November 2002.

[5] C. Genz, F. Rogin, R. Drechsler, S. Rulke, "Visualized SystemC Debugging".

[6] Boue, J., Petillon, P., Crouzet, Y., "MEFISTO-L: a VHDL-based fault injection tool for the experimental assessment of fault tolerance", Fault-Tolerant Computing, 1998. Digest of Papers. Twenty-Eighth Annual International Symposium on 23-25 June 1998 Page(s): 168 - 173

[7] Jenn, E., Arlat, J., Rimen, M., Ohlsson, J., Karlsson, J., "Fault injection into VHDL models: the MEFISTO tool", Fault-Tolerant Computing, 1994. FTCS-24. Digest of Papers., Twenty-Fourth International Symposium on 15-17 June 1994 Page(s):66 - 75

박 인 학

1980 고려대학교 전자공학과(학사)
1983 고려대학교 전자공학과(석사)
1992 프랑스 국립폴리테크닉연구소(INPG)(박사)
1982~2000 한국전자통신연구소(ETRI), 책임연구원
2000.현재 (주)시스템 센트로이드 대표이사
관심분야 : 반도체 CAD, 컴퓨터 그래픽스, 상위수준합성, 웹 CAD
E-mail: ihpark@systemcentroid.com

나 종 화

1981 서강대학교 전자공학과(학사)
1986 Wayne State University(석사)
1989 University of Arizona(박사)
2005~현재 한국항공대학교 전자공학전공
관심분야 : Optical Computing, System Level Design, Multimodal Mouse
E-mail : jwna@hau.ac.kr