韓國 컴퓨터情報學會 論文誌
第13卷 第3號, 2008. 5.

2008-13-3-1-6

메모리 카드 호환성 테스트를 위한 통합 검증 환경

성 민 영*

Co-Validation Environment for Memory Card Compatibility Test

Sung Min Young*

요 약

디지털 카메라, MP3 플레이어 등과 같은 가전 기기에서 낸드 플래시 메모리에 기반한 다양한 메모리 카드가 인기를 얻게 됨에 따라 기존 호스트 시스템과 새로 개발된 메모리 카드 간의 호환성 문제가 제품의 시장 진입에 큰 장애가 되고 있다. 메모리 카드 호환성 테스트를 위한 일반적인 방법은 실제 호스트 시스템을 테스트베드로 사용하는 것이다. 이를 개선하는 방법으로서 FPGA 기반의 프로토타입 보드를 이용하여 호스트 시스템을 에뮬레이션하는 것을 고려할 수 있다. 그러나 이 방법은 긴 셋업 시간을 필요로 하며, 다양한 호스트 및 장치 시스템을 표현하는데 제약이 있다. 본 논문에서는 Esterel 언어와 통합 시뮬레이션 기법에 기반한 모델을 이용하여 메모리 카드와 호스트 시스템간의 호환성 테스트를 위한 통합 검증 환경을 제안한다. 또한, 실제 메모리 카드 개발에 대한 사례 연구를 통해 제안된 기법의 유용성을 증명한다.

Abstract

As diverse memory cards based on NAND flash memory are getting popularity with consumer electronics such as digital camera, camcorder and MP3 player, the compatibility problems between a newly developed memory card and existent host systems have become a main obstacle to time-to-market delivery of product. The common practice for memory card compatibility test is to use a real host system as a test bed. As an improved solution, an FPGA-based prototyping board can be used for emulating host systems. However, the above approaches require a long set-up time and have limitations in representing various host and device systems. In this paper, we propose a co-validation environment for compatibility test between memory card and host system using formal modeling based on Esterel language and co-simulation methodology. Finally, we demonstrate the usefulness of the proposed environment with a case study of real memory card development.

▸ Keyword : 메모리 카드, 낸드 플래시 메모리 (NAND flash memory), 통합 검증 (co-validation), Esterel

• 제1저자 : 성민영
• 접수일 : 2008. 4. 14, 심사일 : 2008. 4. 18, 심사완료일 : 2008. 5. 24.
* 상명대학교 컴퓨터소프트웨어공학과 교수

I. 서 론

플래시 메모리는 비휘발성 (non-volatility), 외부 충격에의 안전성, 저전력 소모 등의 장점으로 인해 다양한 모바일 시스템에서 데이터 저장 및 코드 저장의 용도로 널리 사용되고 있다. 대표적인 플래시 메모리종류로는 낸드 (NAND)와 노어 (NOR)를 들 수 있다. 노어 플래시는 고속의 임의 접근을 요구하는 코드 저장 용도에 특히 적합하다. 한편, 낸드 플래시는 고밀도 저비용의 데이터 저장을 지원하므로 SD (Secure Digital), CF (Compact Flash), MMC (Multimedia Card), 메모리 스틱 등의 이동형 메모리 카드에 적합하다.

최근 들어, 다양한 메모리 카드와 함께 디지털 카메라, 캠코더 등과 같은 다양한 호스트 시스템이 인기를 얻게 됨에 따라, 기존 호스트 시스템과 새로 개발된 메모리 카드 간의 호환성 문제가 제품의 시장 진입에 큰 장애가 되고 있다. 호환성 문제는 호스트 시스템과 메모리 카드가 주로 영어로 기술되어 있는 산업 표준에 기반해 독립적으로 개발되고 있다는 점에 기인한다. 이는 시스템 설계자에 따라 시스템 동작 양식의 모호한 해석을 가능하게 하여 결국, 메모리 카드 시스템의 하드웨어/소프트웨어 개발 중 설계 오류를 유발하게 할 수 있다. 메모리 카드 시스템의 호환성 문제에 대한 다른 이유는, Bluetooth나 USB와 같은 통신 프로토콜에서와는 달리, 호스트 시스템과 메모리 카드 장치간의 정형화된 검증 절차가 없다는 점이다. 따라서, 새로 개발된 메모리 카드는 기존 호스트 시스템에서 많은 시간을 요하는 임의적인 테스트를 통해 호환성이 검증될 수 있다. 그러나, 개발 후반부에서의 테스트 실패는 큰 턴어라운드 시간을 요구하며 특히, 개발된 장치의 내부 상태가 추적불가능한 경우 디버깅을 매우 어렵게 한다. 따라서, 호환성 문제를 효과적으로 해결하기 위해서는 개발 초기 단계에서 시스템 규정에서 허용하는 테스트 케이스들을 미리 실행할 수 있는 효율적이고 신속한 통합 검증 (co-validation) 시스템이 절실히 요구되고 있는 실정이다.

본 논문에서는 통합 시뮬레이션 (co- simulation)을 이용한 호스트와 메모리 카드 기기 간의 호환성 테스트를 위한 통합 검증 환경을 제안한다. 호스트와 기기에 대한 정형적이고 유연한 모델링 방법을 위해서는 Esterel 언어를 활용한다 [1,12].

논문의 구성은 다음과 같다. 제 2절에서는 MMC를 예를 들어 메모리 카드 시스템에 대해 기술한다. 제 3절에서는 관련 연구를 기술한다. 제 4절에서는 MMC 카드 호환성 테스트를 위한 통합 검증 환경을 제안한다. 마지막으로 제 5절, 제 6절에서는 실험 결과와 결론을 기술한다.

II. 메모리카드 시스템 개념 및 MMC 표준

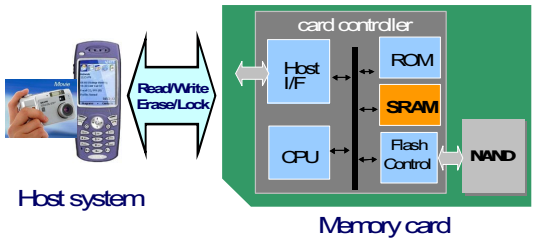

그림 1. 메모리 카드 시스템
Fig. 1. Memory card system

본 논문에서 제안한 방법은 일반적인 카드 시스템에 모두 적용할 수 있지만, 본 논문에서는 실제 사례에의 적용성을 보이기 위해 MMC에 중점을 두어 기술하도록 한다. 그림 1에 보인 바와 같이 MMC 시스템은 MMC 호스트와 MMC 카드로 구성된다. 호스트, 즉, 디지털 카메라, PDA, 무선 전화 등은 외부 메모리 카드를 이용하는 기기를 지칭한다. 호스트와 카드는 클럭, 명령, 데이터 라인을 위한 세 개의 직렬 데이터 버스로 연결된다. 이들 간의 통신은 다음 세 종류의 메시지 토큰을 통해 이루어진다.

명령 (command)은 동작을 시작하는 토큰이다. 각 명령은 호스트에서 MMC 카드로 전송되며 명령 라인을 통해 직렬 전송된다.

응답 (response)은 카드에서 호스트로 전송되며, 이전에 수신된 명령에 대한 응답이다. 응답은 명령 라인을 통해 직렬 전송된다.

데이터 (data)는 호스트에서 카드로 혹은 반대 방향으로 전송된다. 데이터는 데이터 라인을 통해 전송된다.

다른 대부분의 산업 표준과 마찬가지로, MMC 규정은 대부분 영어로 기술되어 있다. 이 규정은 요구 조건에 대한 이해를 돕기 위해 다이어그램과 표를 포함하고 있다.

그림 2와 그림 3은 규정에서 발췌한 다이어그램 예이다. 그림 2와 같은 상태 다이어그램은 MMC의 상태 전이를 설명하는데 사용되며, 그림 3과 같은 타이밍 다이어그램은 시간 제약 조건을 기술하는데 사용된다. 하지만, 이들은 요구조건

을 정의하기 위한 것이 아니라 이해를 돕기 위한 것이므로 모호성이나 불완전성을 해결하기에 충분하지 않다. 예를 들어, 직렬 버스 상의 토큰들 간의 시간 관계를 보이는 다이어그램들이 다수 존재하나, 모든 가능한 시나리오 중 일부만이 기술되고 있다.

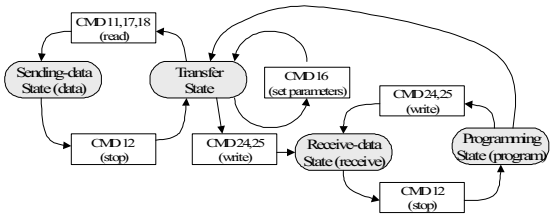

그림 2. 데이터 전송모드를 위한 상태 다이어그램
Fig. 2. State diagram for data transfer mode

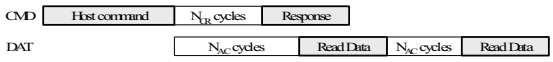

그림 3. 다중 블록 읽기를 위한 타이밍 다이어그램
Fig. 3. Timing diagram for multiple block read

MMC 표준에서는 카드가 동작 중 머무를 수 있는 11개의 상태를 정의하고 있다. 각 상태에서 카드는 호스트로부터 명령 토큰을 수신하고 필요한 경우 응답 토큰을 전송한 후, 요구되는 작업을 수행하고 내부 상태를 변경한다. 예를 들어, programming state에서 만약 호스트가 read 명령을 보내면 잘못된 동작으로 인식된다. 잘못된 명령을 수신하면, 카드는 이를 무시하고 응답을 보내지 말아야 한다. 본 본문의 주 대상은 카드 기기가 호스트 시스템으로부터 전송되는 다양한 명령 순서에 대해 올바르게 응답하는지를 검증하는 것이다.

III. 관련 연구

호환성 테스트를 위한 가장 신뢰할 수 있는 방법은 개발된 카드 기기를 실제 호스트 시스템 상에서 운영하는 것이다. 하지만, 이러한 테스트는 현재 가용한 호스트 세트에 국한된다는 제약이 있다. 미래에 출시될 제품에 대한 잠재적 문제는 해결되지 않는다. 또한, 문제가 발견되었을 때, 기기의 내부 상태를 추적하기 어려우므로 상당한 디버깅 노력이 필요할 수 있다.

FPGA를 이용한 하드웨어 프로토타입은 개선된 해결책을 제공할 수 있다. 거의 최종 구현 단계에서 신속한 호환성 테스트를 제공한다. 그러나 이 방법은 오랜 셋업 시간을 필요로 하며 높은 비용을 요구할 수 있다.

통합 시뮬레이션 접근법은 하드웨어/소프트웨어 통합 검증 목적에 유용하게 사용될 수 있다. 통합 시뮬레이션은 C, verilog, SystemC[11] 와 같은 다양한 언어를 지원한다. 이 방법은 유연한 모델링을 제공하고 시뮬레이션 중 내부 추적을 할 수 있다는 장점을 가지고 있다. 반면, 긴 시뮬레이션 시간을 요구할 수 있다는 단점도 존재한다.

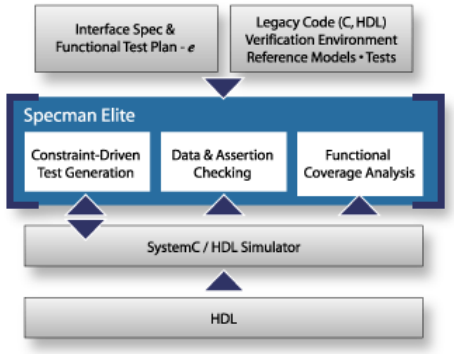

그림 4. Versity사의 Specman Elite
Fig. 4. Specman Elite

임베디드 시스템의 검증을 위한 상용 툴들도 존재한다. 특히, Versity사의 Specman[7,13]과 Synopsys사의 Vera[8]는 산업계에서 유용성이 인정될 수준에 도달하였다. Specman은 e라는 자체 모델링 언어를 이용하며 모델링 및 검증에 유용한 기능을 제공한다 (그림 4 참조). 이 중 constraint solving 및 coverage analysis는 특기할 만하다. Constraint solver는 사용자로 하여금 실제 데이터 값에 대한 제약 조건과 함께 매개화된 테스트 시나리오를 작성하는 것을 가능하게 해준다. Coverage analyzer는 테스트가 완료된 후 완전성(completeness)의 정도를 리포트한다.

IV. 통합 검증 환경

그림 5는 제안하는 통합 검증 환경을 보인 것이다. 호스트 모델은 MMC 카드 규정에 따라 Esterel 언어를 이용하여 구현되었다. MMC 카드의 하드웨어/소프트웨어 모델은 각각 Verilog와 C 언어를 이용하여 모델링 되었다. 호스트 모델의 컴파일 후 자동적으로 C 코드가 생성되며 이는 통합 시뮬레이션 환경에서 MMC 카드 모델과 결합된다. 하드웨어/소프트웨어 통합 시뮬레이션으로는 SeamlessTM을 사용하였다

[4]. 비록 호스트 모델이 규정에서 허용하는 명령 순서의 모든 가능한 조합을 생성할 수 있지만, 동작중인 기기 모델의 내부 상태를 추적하기 위해 결함을 재현하는 용도로는 Set Error DB로부터의 세트 에러 패턴을 사용할 수 있다. 호스트와 기기 모델의 시뮬레이션 중에는 호스트 시스템에서 발생된 명령 트레이스에 대한 로그와 함께 통과/실패의 결과가 보고된다.

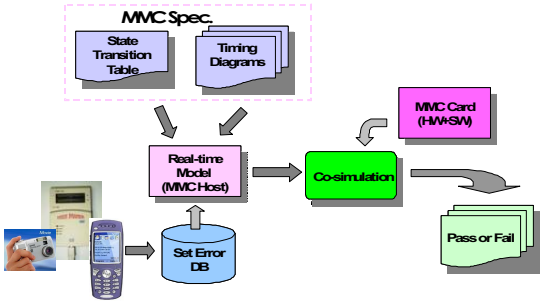

그림 5. 통합 검증 환경
Fig. 5. Covalidation environment

4.1 Esterel을 이용한 호스트 시스템 모델링

행위 모델링 언어로서 동기 언어 (synchronous language) 중 하나인 Esterel을 이용하였다[4,10]. 이 언어는 복잡하고 동시 진행적이며 서로 통신하는 상태 기계들을 매우 효과적으로 기술할 수 있다. Eserel은 병렬성이 중요한 응답 시스템(reactive system)을 프로그래밍하는데 사용되도록 설계되었다. Java와 같은 기존 병렬 언어와는 달리 Esterel은 병렬 구조를 순차적인 형태로 컴파일함으로서 병렬성 (concurrency)과 확정성(determinism)을 모두 제공한다. 따라서 Esterel은 응답 시스템의 복잡한 제어 오토마타를 프로그래밍하는데 유용하다. Esterel로 작성된 모델은 C 코드로 변환될 수 있다. MMC 규정은 많은 복잡성을 갖는다. 따라서 기능을 모듈로 분류하고 이들간의 통신을 신중하게 설계함으로서 복잡성을 관리할 수 있어야 한다. 가장 큰 복잡성은 명령들 간의 상호 작용에 대한 규정에 있다. 예를 들어, 호스트는 데이터의 읽기/쓰기 중에 Stop Transmission, Deselect Card, Inactive State, Go Idle State 등의 명령을 발생시킬 수 있다[6]. 따라서 이러한 명령들에 대한 모듈은 이러한 상황을 올바르게 처리하기 위해 잘 조율되어야 한다. 이 모듈들 집합은 세 가지 그룹으로 분류될 수 있다. 이에 대해서는 이후에 기술한다.

```
1 loop
2  pause;
3  trap Reset_after_Violation in
4    every Violation_Found do
5      pause;
6      exit Reset_after_Violation
7    end every
8    ||
9    run CMD00 || run Scheduler ||
10    run Dataline_Moniter ||
11    run CMD_Sender || run Response_Receiver ||
12    run Data_Receiver || run Data_Sender
13    ||
14    loop
15      abort
16        run CMD01 || run CMD02 || run CMD03 ||
17        run CMD07 || run CMD09 || run CMD10 ||
18        run CMD12 || run CMD13 || run CMD17 ||
19        run CMD18 || run CMD23 || run CMD24_25
20      when Soft_Reset
21    end
22  end trap;              % Reset_after_Violation
23
24  % Send CMD0 for reset
25  trap CMD0_INIT in
26    loop
27      run CMD_Sender
28      ||
29      [
30        pause;
31        emit Put_cmd(0);
32        emit Put_arg(0);
33        pause;
34        await 48 tick;
35        exit CMD0_INIT
36      ]
37    end loop             %loop
38  end trap               %CMD0_INIT
39 end     % Loop-end
```

그림 6. main 모듈
Fig. 6. main module

4.1.1 Main module

각 Esterel 프로그램은 고유한 main module을 포함하게 된다. 이 모듈은 시스템의 인터페이스, 설정, 전역 상태 변화를 정의한다.

그림 6은 main module을 보인 것이다. loop으로 둘러싸여 있는데 이는 시스템이 종료하지 않는다는 뜻이다. loop는 두 파트로 분리될 수 있다. 제어가 첫 번째 파트에 있을 때, 카드는 정상적으로 동작하는 것이다. 즉 규정을 위배하지 않은 것이다. 몇몇 모듈은 run 명령에 의해 생성되며 (9-12 라인), 다른 모듈들은 다른 내부 loop에서 abort-when 구문을 통해 생성된다 (14-21 라인). 이 구분은 CMD0에 의한 소프트 리셋을 처리하는데 필요하다. CMD00 모듈은 Soft_Reset 시그널을 발생시킬 수 있으며 이는 모듈의 재시작을 유발한다. 따라서 CMD00 모듈 자체는 리셋 시그널에 의해 리셋되어서는 안된다.

4.1.2 인터페이스를 위한 모듈

MMC는 명령과 응답을 위한 CMD 라인과 데이터 전송을 위한 DATA 라인을 포함한다. 인터페이스 모듈은 이 라인들을 관리한다. 이 모듈들은 bits로부터 (로의) 토큰 packing (unpacking)을 담당하며 토큰의 도착/전송을 알리는 역할을 한다.

SeamlessTM 환경과의 인터페이스 또한 인터페이스 모듈에 의해 이루어진다. 예를 들어, CMD_Sender (11 라인) 모듈은 명령 토큰을 명령 라인에 적재하는 역할을 한다. SeamlessTM 는 한 비트의 데이터를 인터페이스 라인에 쓰는 것을 시뮬레이션하는 API 함수를 제공한다. CMD_Sender 모듈 토큰을 적재하기 위해 이 함수를 호출한다. 그림 6의 10-12 라인은 이러한 모듈들의 생성을 보이고 있다.

4.1.3 명령을 위한 모듈

MMC 규정은 여러 명령을 정의하고 있다. 제안하는 MMC 호스트 모델에서 각 MMC 명령은 이를 처리하는 중 통신 제어 흐름을 관리하는 모듈에 의해 표현된다. 명령 토큰이 발생하면 CMD 라인 담당하는 모듈 (그림 6의 9 라인에 존재하는 Scheduler)은 이 명령의 시작을 뜻하는 시그널을 발생시킨다. 다음, 이에 해당하는 모듈이 이를 통지 받고 적절한 처리를 시작한다.

```
1 var cmd : integer in
2 loop
3 trap T in % wait for CMD0
4   every End_cmd do
5         cmd := ?End_cmd;
6 if cmd = 0 then exit T end if
7       end
8   end trap;
9 if not(?State_changed= INA) then % Legal CMD
10       pause;
11       emit State_changed(IDLE);
12       emit Soft_Reset
13   end if;
14   await NCC tick
15 end loop
16 end var
```

그림 7. CMD0 모듈
Fig. 7. CMD0 module

지면상의 제약으로 인해 가장 단순한 모듈을 예를 들어 기술한다. 그림 7은 CMD0, 즉 Reset 명령을 처리하는 모듈을 보인 것이다. 이는 초기화 단계 혹은 동작 단계 중 카드를 리셋하는데 사용된다. 이 명령은 INA (Inactive state)를 제외한 모든 상태에서 실행될 수 있다. 카드가 CMD0를 수신하면, 내부 상태를 IDLE로 변경한다.

loop 구문은 CMD0가 발생할 때마다 이 코드가 실행되어야 한다는 것을 의미한다. 3-8 라인은 CMD0를 대기하는 역할을 한다. 명령이 발생하면 (4 라인), 이것이 CMD0인지 체크하여 만약 그렇다면 트랩에서 빠져나옴으로서 (6 라인) every loop에서 나오게 된다. 10-12 라인은 명령이 INA 이외의 상태에서 명령이 발생했을 때 실행된다 State_changed (IDLE)의 발생은 IDLE로의 시스템 상태 전이를 표현한다. 실제 리셋은 Soft_Reset 시그널을 발생시킴으로서 일어난다. 이는 그림 7의 main module에 의해 처리된다.

V. 실험 결과

본 절에서는 제안된 기법에서 사용된 검증 테스트 방법을 보이고 MMC에 대한 검증 테스트 결과를 요약한다.

첫 번째 검증 방법은 시나리오 기반 방법이다. 시나리오 기반 테스트 방법에서는 시나리오 케이스에 따라 호스트가 일련의 MMC 명령 수행한다. 시나리오 기반 방법은 디버깅 목적으로 특정 결함을 반복적으로 재생할 때 유용하다. 특정 결함에 대한 재생은 대상 기기의 내부 상태를 조사하여 결함의 원인을 찾을 때 큰 도움을 준다. 두 번째 검증 방법은 임의 테스트 방법이다. 호스트는 임의의 MMC 명령을 수행한다. 임의 테스트가 시작되면 명령과 이에 대한 지연시간이 로그에 저장된다. 이 로그는 특정 결함을 재현할 때 이용된다.

5.1 결함 재현

첫 번째 시나리오로서 특정 결함이 재현될 수 있는지 살펴보고 그 원인을 설명한다. 몇몇 호스트는 Relative Card Address (RCA)를 0으로 설정하며 이는 MMC가 초기 상태를 통과하지 못하도록 한다. 기존의 디버깅 방법에 의하면, 이 결함은 logic analyzer 및 packet analyzer를 사용하여 이틀 동안 디버깅하여 발견될 수 있었는데 반해, 통합 검증 환경에서는 수 시간 만에 발견될 수 있었다. 이 결함을 재현하는데 필요한 명령 순서는 그림 8에 나타나 있다.

Command index	*delay*
CMD0	*4208*
CMD1	*200*
CMD2	*200*
CMD3 (RCA = 0)	*200*
CMD7 (RCA = 0)	*200*

그림 8. 결함 재현을 위한 시나리오
Fig. 8. Scenario for fault reproduction

그림 9. 결함 재현 결과
Fig. 9. Result of fault reproduction

첫번째로 CMD0 (Reset) 명령이 발생되어 카드가 리셋되며 4208 단위시간 후에는 CMD1 그리고 이후 CMD2, CMD3, CMD7이 지정된 지연을 가지며 도착한다. 이 시나리오에서 MMC는 CMD3 (SET_ RELATIVE_ADDR)이 도착할 때까지 올바르게 동작하였다. 하지만, CMD7 (SELECT/DESELECT _CARD)가 RCA=0과 함께 전송된 이후로는 더 이상 반응하지 않았다. CMD7 매개변수 중 하나인 RCA가 0으로 세트되면 모든 MMC 카드에 대한 선택이 해제되며 MMC 카드로부터의 더 이상의 응답이 없었다. RCA가 0이면 MMC 호스트는 MMC 카드로부터의 응답을 받을 수 없으며 따라서 대기 상태에서 전송 상태로 이동할 수 없다. 이에 따라 MMC는 초기화될 수 없었으며 호스트 프로그램은 위반 상태로 이동하면서 실행을 중지하였다. 그림 9은 이 테스트 시나리오의 화면 출력을 보인 것이다.

5.2 임의 테스트 벡터 생성

시나리오 기반 테스트가 디버깅 용도로 기존 설계상의 오류를 재현하는데 필요한 반면, 임의 테스트는 테스트 케이스의 광범위한 탐색을 통해 잠재적인 결함을 다루는데 필요하다.

임의 테스트 중 발견된 결함의 예로서 데이터 쓰기 중 불법 명령 전송 결함이 발생한 것을 들 수 있다. 이 결함은 512 바이트의 데이터 전송 후에 발생한 CRC 상태 에러 응답을 지칭한다. 규정에 따르면, CRC 상태 프레임은 start bit, DATA CRC, end bit 로 구성된다. 정상적인 동작 하에서는 DATA CRC는 010 혹은 101 이어야 한다. 만약 데이터 쓰기 동작이 오류 없이 완료되면 DATA CRC는 010이어야 한다. 만약 오류가 발생하면 DATA CRC는 101이 된다. 새로 발견된 결함에 대한 명령 순서는 그림 10과 같다.

```
...
TRAN state
CMD25(Open-ended Multiple block write)
First data block writing
CMD9 (illegal case)
```

그림 10. 임의 테스트 시나리오
Fig. 10. Scenario for random test

실험에서 호스트가 데이터 블록 전송중 잘못된 명령을 전송하게 되면 CRC 상태 프레임의 DATA CRC가 111이 되는 것으로 밝혀졌다. 이는 잘못된 값이다. 규정에 따르면, 데이터 전송 중일 경우 CMD9는 잘못된 명령이다. 그림 11은 임의 테스트 결함에 대한 화면 출력을 보인 것이다.

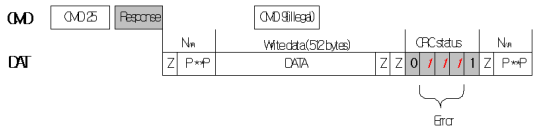

그림 11. 임의 테스트 결과
Fig. 11. Result of random test

전술된 바와 같이, 시나리오 기반 테스트는 동일한 결함을 재현함으로서 결함을 수정하는데 유용하다. 한편, 임의 테스트는 잠재적인 호환성 문제를 발견하고 MMC가 규정을 준수하여 잘못된 동작을 유발하는 것을 방지한다.

VI. 결론

본 논문에서는 메모리 카드 호환성 테스트를 위한 Esterel 기반의 통합 검증 환경을 제안하였다. Esterel 기반의 호스트 모델링은 다른 언어에 비해 정형성 및 유연성 측면에서 많은 장점을 제공한다. 호스트 모델은 MMC 카드 규정에 따라 Esterel 언어를 이용하여 구현되었다. MMC 카드의 하드웨어/소프트웨어 모델은 각각 Verilog와 C 언어를 이용하여 모델링 되었다. 호스트 모델의 컴파일 후 자동적으로 C 코드가 생성되며 이는 SeamlessTM 통합 시뮬레이션 환경에서 MMC 카드 모델과 결합되도록 하였다. 제안된 호환성 테스트 환경을 실제 카드 개발에 적용한 결과, 타이밍 위반, 불법 응답과 같은 설계 오류들이 초기 개발 단계에서 발견될 수 있었으며 기기의 내부 상태 추적이 가능하므로 디버깅 노력을 크게 줄일 수 있었다.

참고문헌

[1] Berry, G.: The Esterel Language Primer, version v5.91. ftp://ftp-sop.inria.fr/esterel/pub/ papers/primer.pdf.

[2] Samsung Electronics Co., "NAND Flash Memory & SmartMedia Data Book", 2007.

〔3〕 http://www.samsung.com/Products/Semiconductor/index.htm.
〔4〕 Mentor Graphics, http://www.mentorgraphics.com.codesign.
〔5〕 Halbwachs, N.: Synchronous Programming of Reactive Systems.
〔6〕 The MultiMediaCard System Specification, 3.1. MMCA Technical Committee, http://www.mmca.org/tech/tech.html.
〔7〕 Verisity, http://www.verisity.com.
〔8〕 Synopsys, http://www.synopsys.com.
〔9〕 Amar Bouali. Xeve: an esterel verification environment. In the 10th International Conference on Computer Aided Verification, 1998.
〔10〕 Gerad Berry, Michael Kishinevsky and Satnam Singh, "System-Level Design and Verification Using a Synchronous Language", ICCAD'03, Nov. 2003, San Jose, CA, USA.
〔11〕 Open SystemC Initiative, http://www.systemc.org.
〔12〕 The Esterel Language, http://www-sop.inria.fr/meije/esterel /esterel-eng.html.
〔13〕 Specman tutorial, http://www.asic-world.com/specman/tutorial.html.

저 자 소 개

성 민 영

- 1995년 2월 : 서울대학교 컴퓨터 공학과 공학사
- 1997년 2월 : 서울대학교 컴퓨터 공학과 공학석사
- 2002년 8월 : 서울대학교 전기컴퓨터공학부 공학박사
- 2002년 9월 ~ 2006년 8월: 삼성전자 소프트웨어연구소 책임연구원
- 2006년 9월 ~ 현재 : 상명대학교 컴퓨터소프트웨어공학과 전임강사

〈관심분야〉 : 실시간시스템, 임베디드시스템, 운영체제, 멀티미디어 시스템 등