암호 하드웨어 모듈의 테스트를 위한 검증 도구[1)]

양인제[O] 경동욱 김동규
부산대학교 컴퓨터공학과
{ijyang[O], dwkyoung, dkkim}@islab.ce.pusan.ac.kr

A Verification Tool for Testing Cryptographic Hardware Modules

In Je Yang[O] Dong Wuk Kyoung Dong Kyue Kim
Dept. of Computer Engineering, Pusan National University

요 약

암호 시스템들은 복잡한 연산과정을 수행하므로 소프트웨어적으로 구현할 경우 수행속도가 저하되는 단점이 있다. 이를 고속으로 수행하기 위하여 암호 시스템을 하드웨어적으로 구현하는 연구가 활발히 수행되고 있는 것이 현재의 추세이다. 암호 시스템의 하드웨어 모듈을 개발하는 과정 중에는 설계한 모듈이 올바르게 동작하는 지의 여부를 검증하는 과정이 필수적으로 포함된다. 이를 위하여 시뮬레이션을 이용한 방법, 테스트 보드를 직접 구현하는 방법 등과 같은 검증 방법들이 주로 사용되고 있다. 암호 하드웨어 모듈은 기존의 방법만을 적용하면 최대 1024, 2048 비트 정도의 많은 비트를 동시에 검증을 수행하므로 시각적으로 판별하기 곤란한 문제가 발생한다. 본 논문에서는 기존의 검증 방법을 보완하는 방법으로 PC 기반의 소프트웨어 통제하에서 암호 하드웨어 모듈을 효과적으로 검증할 수 있는 검증 방법을 제시하고자 한다.

1. 서론

통신 기술의 발달로 인해 많은 정보들이 인터넷과 같은 네트워크를 통하여 전파되고 있다. 이러한 정보 중에서 개인 신상이나 예금 입출력등과 같은 중요한 정보는 불법적으로 접근 할 수 없도록 하기 위하여 암호화/복호화 과정이 수행되고 있다. 그런데 암호화 알고리듬은 복잡한 연산과정을 거치므로 소프트웨어 기반 하에서 동작할 경우, 긴 수행 시간을 필요로 하기 때문에 빠른 속도를 요구하는 환경에서는 적합하지 않다. 그러므로 빠른 속도를 요구하는 환경에서는 암호 알고리듬이 하드웨어 모듈에 탑재 되어 동작되는 것이 바람직하다. 이러한 요구에 따라 현재 많은 암호 하드웨어 모듈이 개발되고 있으므로, 이러한 모듈을 효율적으로 검증하는 방법에 대한 연구가 필요한 실정이다.

현재의 검증 방법은 시뮬레이션을 이용한 방법, 테스트 보드를 이용한 방법 등이 있다. 하지만 암호화 알고리듬이 많은 입출력 정보를 가지고 있고, 이를 동시에 검증해야하므로 기존의 검증 방법만으로는 효율적으로 검증 할 수가 없다.

본 논문에서는 기존의 검증 방법을 보완하는 방법으로 PC 기반의 소프트웨어 통제 하에서 암호 하드웨어 모듈을 효과적으로 검증할 수 있는 검증 방법을 제시하고자 한다. 제시하는 검증 제어 프로그램은 하드웨어 모듈 입출력을 제어하여, 실시간으로 소프트웨어의 동작 결과와 동시에 하드웨어 모듈의 검증 작업을 수행하며, 결과를 시각적으로 보여 준다.

본 논문의 구성은 다음과 같다. 2장에서 검증 도구에서 사용하는 병렬 통신에 관한 기본 지식을 기술하고, 3장에서 검증도구의 전체 시스템 구성을 제안한다. 4장에서는 모듈의 검증에 필요한 FPGA (Field Programmable Gate Array)와 검증 제어프로그램과의 통신에 대한 세부적인 구현 방법을 설명하며, 5장에서 결론을 맺는다.

1) 본 연구는 한국과학재단 목적기초연구 (R01-2002-000-00589-0)지원으로 수행되었음

2. 기본지식

2.1 병렬 통신

PC와 주변기기 사이의 데이터를 병렬적으로 전송하기 위한 규약을 의미하며, 이를 위하여 다음과 같은 5가지의 모드를 지원한다[4].

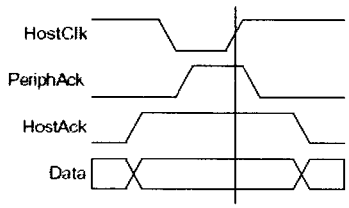
1) Compatibility Mode.
2) Nibble Mode.
3) Byte Mode.
4) EPP Mode(Enhanced Parallel Port).
5) ECP Mode(Extended Capabilities Port).

이러한 5가지 모드 중에서 본 논문에서는 ECP Mode를 사용하였다. ECP Mode란 주변기기에 대한 진보된 통신 모드로서 제안되었고 양방향 통신을 제공하며 양방향에 대해 두 가지 사이클 형식을 가진다: 첫 번째는 데이터 사이클(Data Cycle)로서 주변기기와의 데이터를 전송하는 것이고, 두 번째는 명령 사이클(Command Cycle)로서 주변기기의 제어를 목적으로 제어 정보를 전송하는 것이다. 본 논문에서는 첫 번째 사이클 형식인 데이터 사이클을 응용하여 검증 제어프로그램에 적용하였다.

2.2 통신방법

1) PC에서 주변기기로의 통신

[그림 1]은 PC에서 주변기기로 데이터를 보낼 때의 핸드쉐이크(handshake)를 나타낸 것이고 구체적인 과정을 기술하면 ①-⑤와 같다.

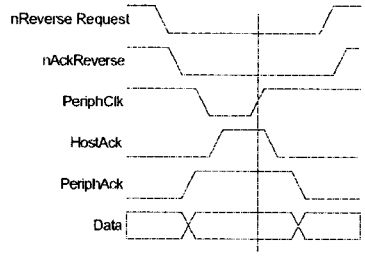


[그림 1] PC에서 주변기기로의 통신

① PC는 데이터를 준비하고, 데이터 사이클이면 HostAck 신호를 '1'로, 명령 사이클이면 HostAck 신호를 '0'으로 바꾼다.
② PC는 HostClk 신호를 '0'으로 바꾸어 유효한 데이터임을 알린다.
③ 주변기기는 PeriphAck 신호를 '1'로 바꾸어 PC에 받을 준비가 끝났음을 알린다.
④ PC는 HostClk 신호를 '1'로 하여 이와 동시에 주변기기로 데이터를 전송한다.
⑤ 주변기기는 PeriphAck 신호를 '0'으로 바꾸어 데이터를 받았음을 알린다.

2) 주변기기에서 PC로의 통신

[그림 2]는 주변기기에서 PC로 데이터를 보낼 때 핸드쉐이크(handshake)를 나타낸 것이고 구체적인 과정을 기술하면 ①-⑥과 같다.



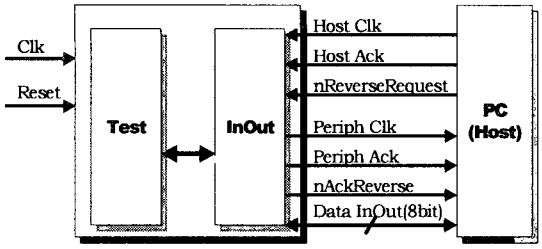
[그림 2] 주변기기에서 PC로의 통신

① PC가 nReverseRequest 신호를 '0'으로 바꾸어 데이터를 요구하고, PeriphAck 신호가 '1'이면 데이터 사이클임 알린 다음, '0'이면 명령 사이클임을 알린다.
② 주변기기는 PC의 요구에 대한 응답으로 nAckReverse 신호를 '0'으로 출력한다.
③ 주변기기는 데이터를 준비하고 PeriphClk 신호를 '0'으로 바꾸어 유효한 데이터임을 알린다.
④ PC는 HostAck 신호를 '1'로 출력하여 받을 준비가 끝났음을 알린다.
⑤ 주변기기는 PeriphClk 신호를 '1'로 바꾸고 동시에 PC로 데이터를 보낸다.
⑥ PC는 데이터를 받았다는 신호로 HostAck 신호를 '0'으로 한다.

3. 검증 모듈 설계

본 논문에서 제시하는 검증 모듈의 전체적인 구성은 기능에 따라 다음과 같은 3가지로 구분한다. 전체 모듈의 구성은 [그림 3]에 도시하였으며, 각 모듈과의 통신과정은 3.1-3절에서 자세히 설명한다.

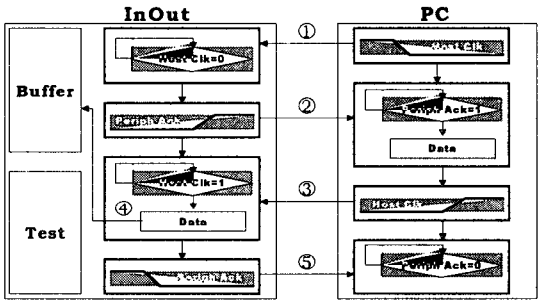
1) **Test 모듈** : 검증하고자 하는 암호 하드웨어 모듈이다.
2) **InOut 모듈** : PC와 Test 모듈간의 데이터 송수신을 담당하는 모듈이다.
3) **PC 모듈** : 검증하고자 하는 모듈의 입출력 데이터를 담당하고 검증하는 모듈이다.



[그림 3] 전체 검증 도구 시스템의 구성도

3.1 PC 모듈에서 InOut 모듈로의 데이터 전송 기능

[그림 4]는 PC 모듈에서 InOut 모듈로의 데이터 이동과 이에 관여하는 제어신호를 보여준다. 구체적인 통신과정은 ①-⑤와 같다.



[그림 4] PC 모듈에서 InOut 모듈로의 데이터전송

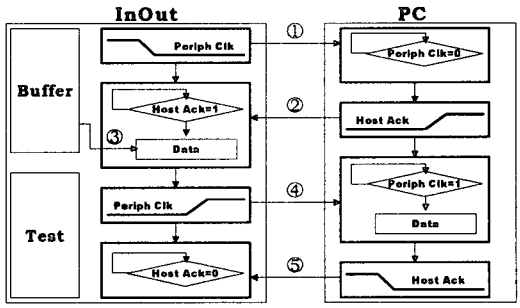
- ①은 PC 모듈에서 데이터를 보낼 준비가 되었을 때, Host Clk 신호를 '1'에서 '0'으로 변화시킨다.
- InOut 모듈은 ②와 같이 데이터를 받을 준비가 될 때 Periph Ack 신호를 '0'에서 '1'로 변화시킨다.
- PC 모듈은 InOut 모듈이 받을 준비가 되었다는 신호를 받으면 ③과 같이 Host Clk 신호를 '0'에서 '1'로 변화시킴과 동시에 8bit 데이터를 보낸다.
- 그러면, ④와 같이 데이터를 InOut 모듈로 보내고 InOut 모듈은 8비트 데이터를 버퍼에 저장한다.
- InOut 모듈에서는 데이터를 받았다는 신호로 ⑤와 같이 Periph Ack 신호가 '1'에서 '0'으로 변화된다.

3.2 InOut 모듈과 Test 모듈사이의 데이터 전송 기능

InOut 모듈에서는 Test 모듈로 데이터를 전송하며, Test 모듈은 이 데이터를 입력으로 하여 연산 수행 후 결과를 InOut 모듈로 전송한다.

3.3 InOut 모듈에서 PC 모듈로의 데이터 전송 기능

[그림 5]는 InOut 모듈에서 PC모듈로의 데이터 이동과 이에 관여하는 제어신호를 보여준다. 구체적인 통신과정은 ①-⑤와 같다.



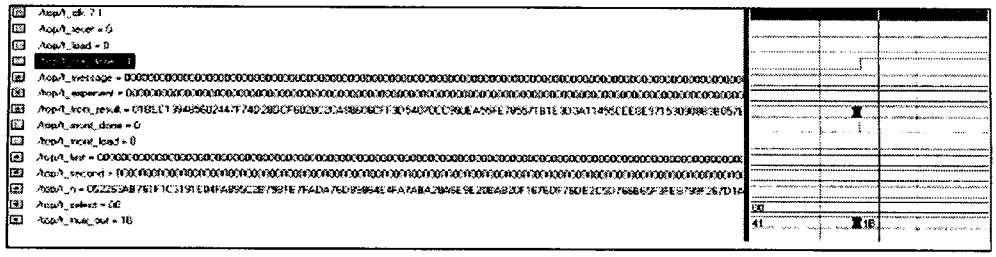
[그림 5] InOut 모듈에서 PC모듈로의 데이터 전송

- InOut 모듈은 Test 모듈의 출력 데이터를 PC 모듈로 전송한다. 즉 InOut 모듈은 데이터 전송 준비가 될 때 ①과 같이 Periph Clk 신호를 '1'에서 '0'으로 변화시킨다.
- PC 모듈은 받을 준비가 되었을 때 ②와 같이 Host Ack 신호는 '0'에서 '1'로 변화시킨다.
- 그러면 ③,④와 같이 InOut 모듈의 버퍼에서 8비트의 데이터를 PC모듈로 보낸다. 그리고 Periph Clk 신호가 '0'에서 '1'로 바뀌게 되면 PC모듈은 데이터를 받는다.
- 그리고, ⑤와 같이 받았다는 신호로서 Host Ack 신호는 '1'에서 '0'으로 변한된다.

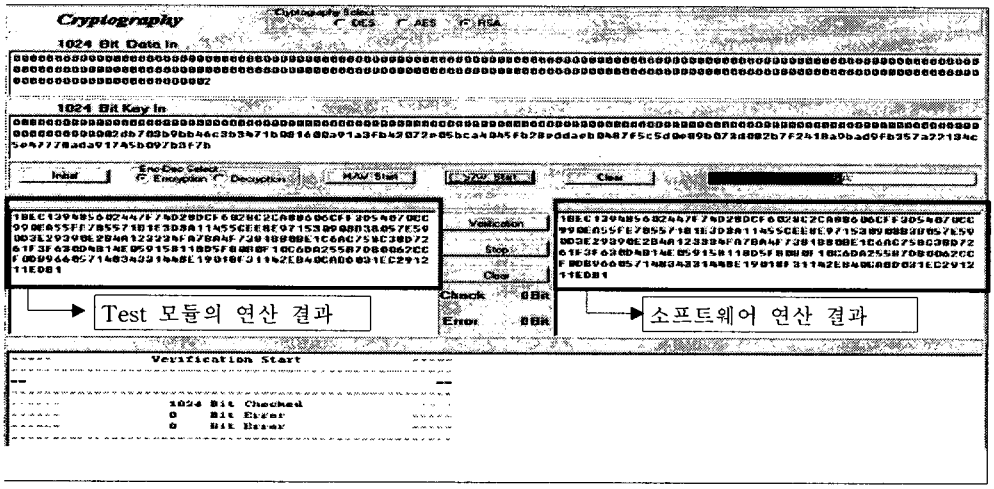
4. 구현

본 논문에서 제시하고 있는 Test 모듈과 InOut 모듈은 하드웨어 설계 언어인 VHDL로 구현하였다. 구현된 VHDL 코드는 Model Technology사의 ModelSim5.2c를 가지고 시뮬레이션 되었으며, 시뮬레이션 결과는 [그림 6]과 같다. 그리고, Altera 사의 Syplify Pro 7.0.1에서 로직 합성(logic synthesis)하였다.

검증 도구의 GUI를 위하여 Delphi 5언어로 프로그램을 작성하였다. 구현된 검증 도구는 실시간으로 임의의 값을 입력하여, 소프트웨어적인 수행과 Test 모듈의 수행 결과를 동시에 얻어 수행결과를 검증할 수 있다. [그림 7]에서는 DES, AES, RSA[1,2,3]의 하드웨어 모듈의 결과 값과 소프트웨어적인 결과값을 비교하는 과정을 검증 예로 보여준다.



[그림 6] 로직 시뮬레이션 결과



[그림 7] 검증 도구에서 검증한 결과

5. 결론

본 논문에서는 실시간으로 소프트웨어의 동작 결과와 동시에 하드웨어 모듈의 검증 작업을 수행하여 결과를 PC에서 시각적으로 보여줄 수 있는 검증 도구를 구현하였다. 구현된 검증도구는 기존의 검증 방법을 보완하며, PC 기반의 소프트웨어 통제 하에서 암호 하드웨어 모듈을 효과적으로 검증할 수 있다.

제시한 검증도구는 검증할 때 아래와 같은 두 가지 인터페이스 조정을 통하여, Test 모듈을 유연하게 범용으로 사용할 수 있다.

1) Test모듈과 InOut모듈과의 인터페이스 조정
- Test모듈의 입출력 데이터 길이에 맞추어 InOut 모듈의 버퍼 길이를 정해줄 수 있다.

2) InOut모듈과 PC모듈과의 인터페이스 조정
- InOut모듈의 버퍼 길이와 PC모듈의 버퍼 길이를 정해줄 수 있다.

앞으로의 연구는 현재 부분적으로 구현중인 검증 프로그램을 더욱 발전시켜, PC 레벨에서 단계별 디버깅이 가능하도록 확장할 예정이다.

[참고문헌]

[1] R.L. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key crytosystems", Commun. ACM 21 (2) pp. 120-126, 1978.

[2] P.L. Montgomery, "Modular multiplication without trial division", Math. Comp. 44 (170) pp. 644-654. 1976.

[3] C. K. Koc. "High-Speed RSA Implementation". Technical Report TR 201, RSA Laboratories, November 1994.

[5] Parallel Port Interfacing, http://www.beyondlogic.org.