The Functional Verification of AES RTL Design Using H/W Assisted Co-Emulation

Jae-Deok Ji, KISA
Byung-Kwon Lee, KISA
Byung-kyu Noh, KISA
2008/09/25
Agenda

I. Introduction
   A. AES (Advanced Encryption Standard)
   B. AESAVS (AES Algorithm Validation Suite)
   C. SystemC and SystemC Verification Extensions

II. H/W Assisted Co-Simulation
   A. Testing Environment
   B. Testing Vector
   C. Testing Method

III. Conclusion
Introduction

- Advanced Encryption Standard algorithm
  - Specified in 2001 by NIST
  - Widely adopted for a variety of encryption need
- But, how can one confirm that a design or IP core of AES is implemented correctly and without security hole?
  - The Verification of AES design is also important
  - Cryptographic Module Validation Program of NIST specifies a test method of validation for cryptographic modules
Introduction

- The validation of AES implementation
  - The test procedures specified in AESAVS of NIST to validate S/W or H/W implementation of AES
  - The verification procedure for H/W implementation of AES is not simple
    - The outputs of H/W design for test vectors are compared manually with those of reference model.
    - It is slow and inefficient process to verify H/W implementation of AES according to AESAVS
Introduction

- Automatically Self-checking testbench
  - Logic Simulation
    - SystemC Transaction Model
  - H/W Assisted Co-Emulation
    - SystemC + FPGA Board
PRELIMINARY - AES

- AES (Advanced Encryption Algorithm)
  - Symmetric Block Cipher Algorithm
  - 128/196/256-bit Key Size, 128-bit input/output
  - consists of 10 rounds
  - Each round composed of 4 transformations:
    - Byte Substitution
    - Shift Rows
    - Mix Columns
    - Bitwise XOR operations with round key
PRELIMINARY - AES
PRELIMINARY – CMVP

- CMVP (Cryptographic Module Validation Program)
  - CMVP maintained by NIST
  - It provides validation test procedures for cryptographic modules.

- AES Algorithm Validation Suite (AESAVS)
  - Especially it is designed for performing validation testing on H/W or S/W implementation of the AES algorithm.
  - 3 Type Tests are specified in AESAVS:
    - the Known Answer Tests (KAT)
    - the Multi-block Message Tests (MMT)
    - the Monte-Carlo Test (MCT)
PRELIMINARY - SystemC

- SystemC
  - It is consisted of a C++ library class and Simulation Kernel.
  - It is mainly used for Modeling or Verification of a design.

- SystemC Verification Extensions (SCV)
  - SystemC + Verification Library
  - It Supports
    - Random Number Generation API
    - Transaction API
H/W Assisted Co-Simulation

- Environment
  - AES Design
    - Described using Verilog HDL
    - Total 13 Cycle, Working Clock@220MHz
    - Total 3497 LUTs on Virtex4-LX60
H/W Assisted Co-Simulation

- **TestBench**
  - Described using SystemC and SCV

- **RTL Logic Simulation**
  - Run on Linux Platform, Simulated using Cadence IUS56

- **Emulation Board**
  - FPGA Board, VIRTEX 4, Dynalith
H/W Assisted Co-Simulation

- H/W Assisted Co-Emulation

![Diagram](image)

- TestBench in high-level language (C/C++)
- HOST COMPUTER (PC, Window XP)
- Automatically generated Interface
- DUT
- Signal
- USB

<HOST PC>

<Emulation Board>
H/W Assisted Co-Simulation
A PROPOSED TESTBENCH

- **Test Vectors**
  - KAT vector set (ECB mode)
  - GFSbox / KeySbox / Variable Key / Variable Text

<table>
<thead>
<tr>
<th>Key</th>
<th>Ciphertext</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x10a58869d74be5a374cf867cfb473859</td>
<td>0x6d251e6944b051e04eaa6fb4dbf78465</td>
</tr>
<tr>
<td>0xcaea65cbb75c9169ced22ebe6e54675</td>
<td>0x6e29201190152df4ce058139def610bb</td>
</tr>
<tr>
<td>0xa2e2fa9baf7d20822ca9f0542f764a41</td>
<td>0xc3b44b95d9d2f25670eee9a0de099fa3</td>
</tr>
<tr>
<td>0xb364ae4e1de1e285eaf144a2415f7a0</td>
<td>0x5d9b05578fe944b3cf1ccf0e746cd581</td>
</tr>
</tbody>
</table>

Ex) KeySbox Known Answer Test Values from AESAVS Appendix C
A PROPOSED TESTBENCH

- The KAT Flow in Test Bench

1. Read vector set

2. Key
   - Make Data Signals

3. Encryption
   - Plain Text

4. Cipher Text
   - Checker
   - Cipher Text
   - Encryption

AESAVS All Test Passed

AES RTL Block

GFSbox / KeySbox
VarKey / VarTxt

testvector.txt
A PROPOSED TESTBENCH

- Test Vectors
  - MCT vector set (ECB mode)
    - In KAT, Fixed test vector set is used as specified in AESAVS appendix
    - In MCT, only test algorithm is specified, not fixed initial values for security reasons
    - In our test bench model
      - Initial value randomly generated using SCV library API
      - The modified MCT used to verify the design efficiently and effectively
Algorithm 1. Monte Carlo Algorithm
1) Key[0] = Key
2) PT[0] = PT
3) For i = 0 to 99
   3.1) Output Key[i]
   3.2) Output PT[0]
   3.3) For j = 0 to 999
      3.3.1) CT[j] = AES(Key[i], PT[j])
      3.3.2) PT[j+1] = CT[j]
      3.3.3) Output CT[j]
      3.3.4) If( Keylen = 128 )
        Key[j+1]=Key[i] xor CT[j]
      3.3.5) If( Keylen = 192 )
        Key[j+1]=Key[i] xor (last 64-bit of bCT[j-1]|CT[j])
      3.3.6) If( Keylen = 256 )
        Key[j+1]=Key[i] xor (CT[j-1]| CT[j])
   3.3.7) PT[0] = CT[j]

Algorithm 2. Modified Monte Carlo Algorithm
1) Key[0] = Key
2) PT[0] = PT
3) For i = 0 to 99
   3.1) Output Key[i]
   3.2) Output PT[0]
   3.3) For j = 0 to 999
      3.3.1) CTReference = AESreference(Key[i], PT[j])
      3.3.2) CTRef= AESRef(Key[i], PT[j])
      3.3.3) If(CTReference != CTref)
        simulation_stop()
      3.3.4) CT[j] = CTRef
      3.3.5) PT[j+1] = CT[j]
      3.3.6) Output CT[j]
      3.3.7) If( Keylen = 128 )
        Key[j+1]=Key[i] xor CT[j]
      3.3.8) If( Keylen = 192 )
        Key[j+1]=Key[i] xor (last 64-bit of CT[j-1]|CT[j])
      3.3.9) If( Keylen = 256 )
        Key[j+1]=Key[i] xor (CT[j-1]| CT[j])
      3.3.10) PT[0] = CT[j]
H/W Assisted Co-Simulation

- Modified Test Method For MCT

Fig. 3) The MCT Flow in TestBench
Conclusion

- The features of the proposed model
  - use AESAVS test value sets and test procedures
    - One can validate the design of AES for the conformance to FIPS–197
  - has an Automated and Self–Checked Structure
    - One can verify the AES design fast and efficiently
  - has Reusability
    - SystemC is based on object oriented programming language C++
    - This model can be easily modified to verify other cryptographic module
  - We can validate implementation of an AES RTL design for the conformance to the FIPS–197 effectively.
Q & A