Validation of an Advanced Encryption Standard (AES) IP Core

Valeri Tomashau, Tom Kean

Algotronix Ltd., PO Box 23116, Edinburgh EH8 8YB, United Kingdom. Contact e-mail: tom@algotronix.com

Abstract This paper describes the package of test bench code required to verify the Algotronix' AES IP Core. Several authors (see the references in [3]) have published papers detailing the implementation of the Advanced Encryption Standard (AES) on FPGA chips; however, the design goals of this AES core are somewhat different from previous work. Rather than emphasizing performance our design emphasizes portability and customer confidence in the security of the VHDL code.

1. Introduction

The AES algorithm was accepted by the National Institute of Standards and Technology (NIST) of the United States as a Standard in November, 2001 [1]. We present a VHDL model of the AES algorithm and the complete test bench for its verification according to the NIST requirements in "The Advanced Encryption Standard Algorithm Validation Suite (AESAVS)" [2].

In most applications AES hardware performance can easily exceed the ability of the system to supply data; therefore Algotronix believes the emphasis on raw performance in the literature is misplaced. Moreover, the performances quoted are for pipelined designs. Pipelining cannot be used in common applications where AES is run in Cipher Block Chaining (CBC) mode with a single stream of data because of a feedback loop which prevents encryption of the next word of data beginning until the ciphertext of the previous word is available. Algotronix has created an IP core design that is area efficient and is portable between FPGA families and between FPGAs and ASICs so that customers can make the most cost effective technology selection.

Cryptographic cores are different from most Intellectual Property cores because, as well as the possibility of design errors customers must consider the possibility of intentional and malicious features being added to the design. In sensitive applications of cryptographic IP cores customers will require to inspect the core source code - this may be mandated by certification schemes such as Common Criteria. Only a source code review can demonstrate that there is no 'backdoor' mechanism incorporated in the core which would compromise security. A simplistic example of a backdoor would be a design which, when provided with a certain pattern of input data, caused the secret key to be written to the ciphertext output. If the pattern which triggers the key to be written is 128 bits long it would be almost impossible to detect the backdoor using test vectors. More subtle and harder to detect variants of this attack are possible. We believe that FPGA IP Core designs which are heavily optimized for performance at the expense of code simplicity and supplied as 'black box' generated netlists are less appropriate for high security applications than simpler cores supplied as HDL source code.

2. Test Bench

The test bench consists of three main components: Known Answer Test, Monte Carlo Test, and Multi-block Message Test [2]. Each of them involves four particular test benches to verify various ciphering processes (encryption in the Electronic Code Book (ECB) mode, encryption in the Cipher Block Chaining (CBC) mode, decryption in the ECB mode, and decryption in the CBC mode) as shown in the table below.

Test Suite / AES Cipher Mode	Known Answer Test (KAT)	Monte Carlo Test (MCT)	Multi-block Message Test (MMT)
ECB	Encrypt	Encrypt	Encrypt
	Decrypt	Decrypt	Decrypt
CBC	Encrypt	Encrypt	Encrypt
	Decrypt	Decrypt	Decrypt

2.1 Known Answer Test

The NIST Known Answer Test (KAT) provides comprehensive coverage of all components of the implementation by applying a set of plaintext (or ciphertext in the case of decryption tests), key, and initial value to the implementation and checking that the correct ciphertext (plaintext) is generated. The various KAT tests are intended to stress different elements of the implementation. The valid expected responses on the KAT tests along with input data are tabulated in the appendixes B, C, D, E to the document [2].

2.2 Monte Carlo Test

The function of the Monte Carlo Tests (MCT) is somewhat different. The KAT tests provide confidence that there are no implementation errors in the design. However, it would be possible to create an AES design which passes all the KAT tests (since the responses are known in advance) but behaves improperly on some other values. For this reason a particular set of test vectors and known-answers are not specified in the MCT test. Instead a generic method of iteratively running the AES algorithm is specified. In formal compliance testing the actual starting value for Monte Carlo testing is chosen by the NIST accredited laboratory to which the design is submitted.

Each Monte Carlo Test is organized as a sequence of 100 multi-block messages of 1000 blocks each. Hence any MCT test involves 100,000 encryption (or decryption) operations.

Monte Carlo testing using the NIST specified conditions requires a long simulation time because it involves 100,000 runs through the implementation under test. The test bench allows the user to specify the conditions (m - the number of messages and n - the number of blocks in the multi-block message) of the Monte Carlo Test. This allows fast preliminary testing with smaller test sets: to carry out the MCT according with the NIST requirements the user specifies n=100, m=1000.

The AESAVS document [2] supplies some known-answer data for two initial iterations of the MCT. In addition we provide a wrapper C program around a software implementation of the AES which allows the user to generate correct values for their own Monte Carlo testing before submitting an application for certification to a NIST accredited laboratory.

The software implementation of AES used for Monte Carlo testing is the well known open source implementation by Brian Gladman [4] not code written by Algotronix. Customers can download the 'known good' AES software used to test the VHDL directly from the third party website. If the C program was written by the same person as the VHDL implementation there could be a question as to whether they had (possibly deliberately) inserted the same faulty logic in both programs so that Monte Carlo testing would not detect some error or backdoor function hidden in the IP core.

2.3 Multi-block Message Test The Multi-block Message Test (MMT) is designed to verify the ability of the AES implementation to process multi-block messages. This test involves ten multi-block messages of different length with number of blocks from 1 to 10. The MMT test is particularly important for verification of the AES implementation in the CBC mode (Cipher Block Chaining mode).

Some sample data sets along with the valid responses of the AES algorithm are provided. In addition there is a wrapper C program around the software implementation of the AES algorithm which allows the user to generate correct response values for any MMT input data.

3. Implementation Details

The AES core processes data blocks of 128 bits using a cipher key of length 128 bits in the CBC and ECB mode only. Encryption and decryption processes are implemented as separate designs. The AES key expansion process is implemented in the hardware design.

Rather than using FPGA memory blocks for the algorithm of the SubBytes Transformation (and the Inverse SubBytes Transformation as well) was expressed with the aid of the eight 8-variable Boolean functions. To implement them on the 4-input LUT based FPGA an S-box substitution table was automatically converted to the set of 4-input Boolean functions by the use of the Shannon expansion under a subset of the four variables. Each target 8-input Boolean function is represented as the two-level composition of 4-variable functions. Some sharing of logic is possible between target functions.

The VHDL code has initially been mapped to the Xilinx Virtex FPGA, but since device specific features such as RAM blocks have been avoided it should be highly portable between FPGA families and even to ASIC. Performance details for the basic un-pipelined design are given below. Since the design is supplied in VHDL source code users can add pipelining as required to trade area for performance. The core is smaller than highly pipelined designs described in the recent literature (see references in [3]) and has a higher clock frequency. ECB mode throughput is lower because there is no pipelining. In CBC mode with a single data stream (which we believe will be the most common mode of use) pipelining cannot be used. This is because the feedback structure of CBC means the next encryption cannot start before the previous one has finished.

Xilinx XCV300-8	Encryption	Decryption
# Slices & (%)	1,495 (48%)	2,587 (84%)
# 4-input LUTs	2,455 (39%)	4,795 (77%)
Gate count	20,541	48,049
Clock (MHz)	51.1	47.5
Speed grade	-8	-8

Conclusion

The combination of a simple, portable and efficient AES IP core supplied as VHDL source code with a comprehensive test bench provides an excellent platform for high security applications.

The test bench described here can also be used (with minor modifications to the wrapper code) to provide additional confidence in AES cores bought from other vendors or developed in house. Using a test bench from an unrelated party provides additional confidence that the test bench has not been manipulated so that it will not detect backdoor code in a particular AES implementation.

References

[1] National Inst. of Standards and Technology "Federal Information Processing Standard Publication 197, The Advanced Encryption Standard (AES)", November 26, 2001

[2] National Inst. of Standards and Technology "The Advanced Encryption Standard Algorithm Validation Suite (AESAVS)", November 15, 2002,

[3] Saqib et al., "Two Approaches for a Single Chip FPGA Implementation of an Encryptor/Decrytor AES Core." Proc. FPL 2003, Springer LNCS2778.

[4] Gladman, Brian, "Implementations of AES (Rijndael) in C/C++ and Assembler." Web Page: http://fp.gladman.plus.com/cryptography_technology/rijndael/index.htm