ATLANTIS PRESS

Advances in Engineering Research (AER), volume 82

2016 International Conference on Engineering and Advanced Technology (ICEAT-16)

A UVM-based AES IP Verification Platform with Automatic Testcases Generation

Lin Zhu[1], Ligang Hou[1], Qiuyun Xu[1], Jingsong Zhi[1], Jinhui Wang[2]

[1]VLSI & System Lab, Beijing University of Technology, Beijing 100124, China
[2]Department of Electrical and Computer Engineering, North Dakota State University, ND 58102 USA.

Keywords: UVM,Verification,AES,Automatic,Coverage

ABSTRACT:This paper applied UVM (Universal Verification Methodology), an advanced verification methodology which was based on SystemVerilog language to build AES (Advanced Encryption Standard) IP verification platform and environment. Functional verification of the AES module, through a large number of testcases and constrained random test could achieve 100% functional coverage. In addition, the testcases will be run by the verification platform with automatic generation. It both can run the specified testcases in the terminal, which can customize the testcases variables through the orientation test method, and can automatically run all the testcases of the verification platform. This method could improve the efficiency and reusability of the verification, and the simulation results show that the AES IP design is successful.

INTRODUCTION

With the integration circuit design to VLSI (Very Large Scale Integration Circuit) development, the chip verification is becoming more and more difficult. The workload of verification has accounted for about 70% of the whole SoC development. Therefore, it is essential for us to improve the chip verification of efficiency[1]. The Universal Verification Methodology (UVM) is a powerful verification methodology, which can verify all types of digital logic designs, large or small, FPGA or ASIC, and so on. Furthermore, UVM[2] is derived from OVM (Open Verification Methodology), VMM (Verification Methodology Manual) and AVM (Advanced Verification Methodology). This means that UVM testbench architecture and classes are inherited from other methodologies that have proven effective for verification of digital designs.

AES (Advanced Encryption Standard) also called Rijndael algorithm. It is a symmetric encryption algorithm, which can encrypt and decrypt with the same key. As the DES[3] (Date Encryption Standard) is short, and it is easy to crack. AES has replaced the DES and became the new generation of encryption standard. AES can use cipher keys with lengths of 128, 192 and 256 bits. This paper mainly uses the UVM methodology and the typical UVM platform to verify 128 bits of the AES module, which can verify process of encryption and decryption.

AES ALGORITHM

AES[4] is a kind of high efficient and secure symmetric advanced encryption standard. According to the different ways of encrypting the message, the symmetric cipher algorithm can be different. This standard specifies the Rijndael algorithm, which is a symmetric block cipher. This algorithm can process data blocks of 128 bits, using cipher keys with lengths of 128, 192, and 256 bits. AES-128 algorithm runs

throughout the process of HDCP, including HDCP authentication stage and HDCP encryption stage. AES algorithm is the core of HDCP.

Due to the limited length of key, it is used to extend key into a longer key with KeyExpansion, as a key encrypted with the all rounds. A 128 bits plaintext block is entered through Nr round of transformation, which is encrypted into a 128 bits block cipher. Other each round of encryption process is the same besides the final round of the encryption process.

When it comes to AES algorithm process, plaintext with expanded key uses an XOR operation, then following these transformations SubBytes(), ShiftRows(), MixColumns(), and AddRoundKey() transformation– are described in Figure 1. These processes should repeat nine times, and then the last time algorithm does not need MixColumns(). Finally, the Key Expansion uses an XOR operation and outputs ciphertext.

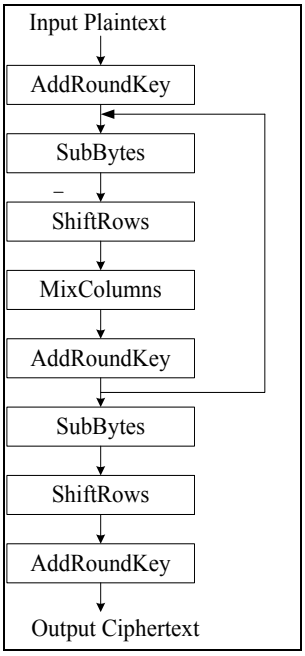

Figure 1. Aes cipher.

UVM VERIFICATION PLATFORM

The UVM verification platform is composed of a series of reusable components, which can be used as a Universal Verification Component. These components have many advantages, such as configuration, encapsulation, easy-to-use and high reusability. A Universal Verification Component has a fixed structure, including a series of components to complete for a particular protocol standard simula tion, inspection and collection coverage. These components play an important role in UVM verification platform.

AES IP VERIFICATION ENVIRONMENT

A AES IP verification platform based on UVM is designed as shown in Figure 2. The design of platform can be partitioned into three primary parts: the test section, the environment section, and the sequence (stimuli) section. The base_test module contains aes_env module that consists of aes_agent_i, aes_agent_o, aes_model, aes_scoreboard. It is called an environment because it can contain the components, which are necessary to create an effective verification environment.

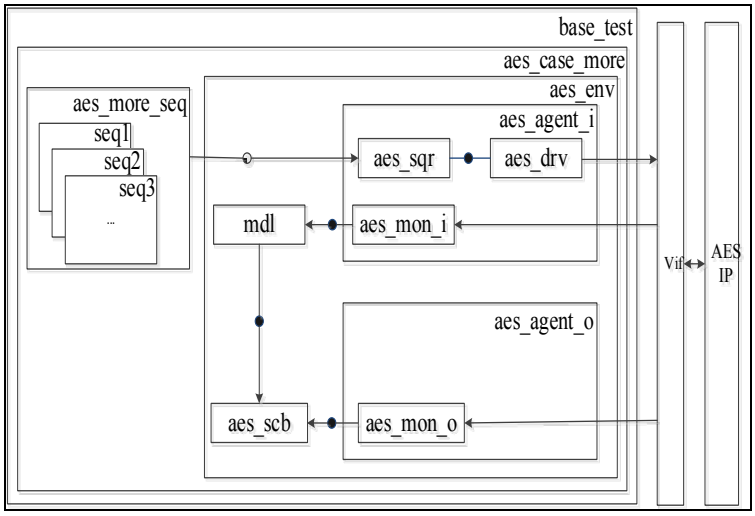

Figure 2. Aes IP verification platform.

As shown in Figure 2 we can see that the aes_agent_i is configured active model and contains a aes_driver, aes_sequencer, and aes_monitor_i. The aes_driver not only can drive the interface to the DUT (AES IP) with that transaction, but also can send a response to the aes_sequencer. The aes_driver receives the transaction through the aes_sequencer. The aes_sequencer can control the flow of a series of sequences that are created either with constrained random or directed data. And the aes_sequence represents a sequence item as a packet of data. The aes_monitor_i can capture the transactions and verify validity. The aes_agent_o is configured passive mode and only contains aes_monitor_o. Its action is capturing the aes_sequence_item and passing the information along to the aes_scoreboard.

The aes_sequence_item contains the configuration of AES IP, including the key, the plaintext, the ciphertext and enable signals and so on. Red points represent aes_sequence_item, which reflects the flow direction of data package as shown in Figure 2. The basic function of the aes_scoreboard is to check the correctness of the AES IP design. It can compare the output data of the aes_model and the aes_monitor_o.

AES IP VERIFICATION SCHEME

Both constrained random and directional incentives can be applied in AES verification platform depended on the basic idea of generation incentive. Firstly, 128 bits key and 128 bits datain of aes_sequence_item use the constrained random. One bit aesready signal uses directional incentive. According to the protocol, data package could not transmit if the aesready=1'b1. Data package began to transmit if the aesready=1'b0.

AES IP VERIFICATION RESULTS

When the verification engineer has built the UVM verification platform, including the test, the environment and the sequence section. So, now we should pass them into the top_tb via the uvm_config_db using the set() function call, and start UVM using run_test() method. After running, we can verify the AES IP design using waveform, functional coverage, code coverage and scoreboard results, making it easier to observe the correctness.

FUNCTIONAL COVERAGE

Functional coverage[6] is an important part of verifying the AES IP design. It is an integral part of the SystemVerilog[7] language, including covergroup, coverpoint, bins definitions, and a sample() method. In this AES IP verification platform, the aes_driver is an ideal location for collecting functional coverage information. For instance, the codes for covergroup in the aes_driver is to sample the aesready signal as shown in Figure 3.

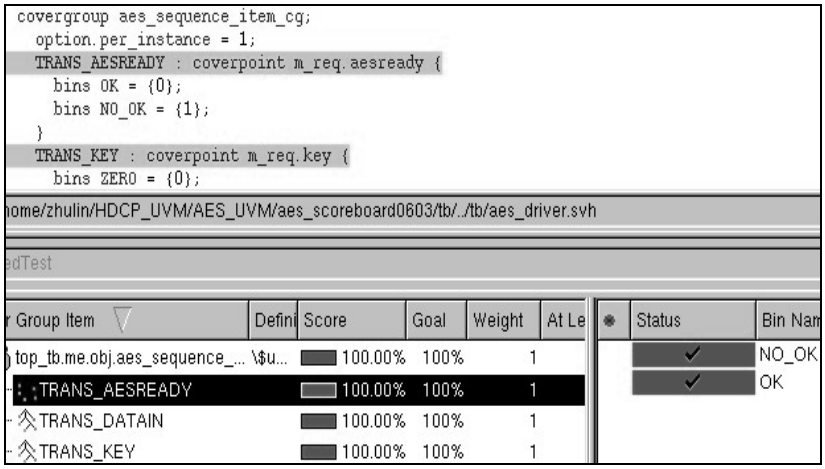

Figure 3. Codes and results for covergroup.

As shown in Table 1, this platform can achieve 100% functional coverage. The covergroup name is aes_sequence_item_cg, there is three coverpoint: key, datain and aesready. In the aes_seq_lib.sv, there are different sequences. Reusable testcases[5] can generate a new sequence by overloading an exsisting sequence.

TABLE I. FUNCTION COVERAGE

Name	Score	Covergroup
Function Groups	100%	100%
Aes_sequence_item_cg	100%	100%

CODES COVERAGE

As shown in Table 2 we can see that branch coverage and condition coverage reaches 100%. Toggle coverage reaches 96.84%, because some signals are not required to flip the state. And line coverage achieves 95.03%, because there are some default states in the KeyExpander.v and a lot of redundancy in this sequence.

TABLE 2. CODES COVERAGE

Name	Line	Toggle	Branch	Condition
Top_tb	95.03%	96.84%	100%	100%
I_aes	96.48%	96.02%	100%	100%
SubBytes	99.62%	100%	100%	-
ShiftRow	100%	100%	100%	-
MixColumn	100%	100%	100%	-
KeySelector	100%	93.99%	100%	-
KeyExpander	83.19%	97.32%	100%	-
AddRoundkey	100%	100%	100%	-
Input_if	-	97.95%	-	-

SCOREBOARD RESULT AND WAVEFORM

Table 3 shows the scoreboard results of executing the aes_case0. From the log panel of VCS, actual values and expect values compare successfully.

The essential role of a scoreboard in UVM platform is to verify that actual outputs from DUT match predicted output values. After running, the DUT outputs can be found whether correct or not. If finding errors, having access to these inputs can help in debugging the incorrect results.

TABLE 3. Results of case0_sequence.

Name	Size	Value
Key	128	'h68bcc51ba9db1bd0faf15e9ad8a5afb9
Datain	128	'h18fae4206afb51493ba0bede0c46a991
Dataout	128	'h4f148d11dd4918106fab166ff6fda6ed
Aesready	1	'h0
Aesdone	1	'h0

As specified in Figure 4, there is a waveform of running aes_case0. The key, aesready, datain and dataout all meet the requirements.

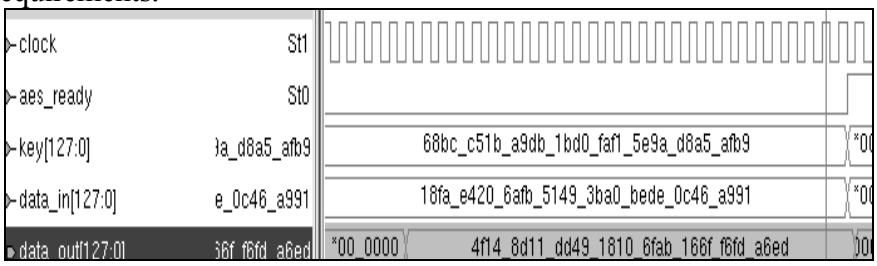

Figure 4. Waveform of AES IP.

AUTOMATIC TESTCASES GENERATION

When we need to verify the IP level or SoC level with a large number of testcases, it will increase time costs and the manpower to manually add and modify testcases. So, automatic testcases generation can simplify steps by script. As shown in Figure 5 we can see the flow chart.

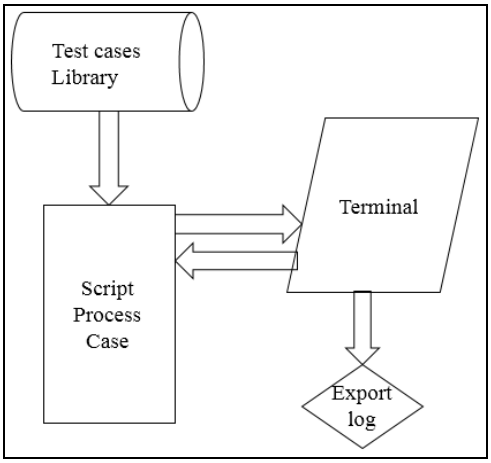

Figure 5. Script flow chart.

All the testcases which are in the validation platform will be put in a folder, firstly, method will generate a file that contains all the testcases name.

Then, according to the words that appear on the terminal selects whether customizing the testcase variables, or automatically running all the testcases of the verification platform. As shown in Figure 6.

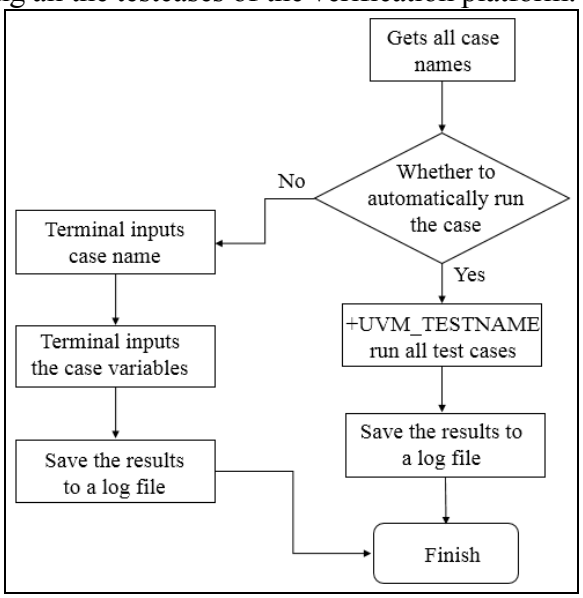

Figure 6. Script flow chart.

If choosing " Yes", it will automatically run all the testcases. The practice is to run the script "run_tc", and change name of "+UVM_TESTCASE = xxx", which replacing case name with xxx. But, testcases will need to be selected by verification engineer if choosing " No". They can input case name in the terminal and input the case variables that they want to get these results. In the end, the results that we want will be saved in a log file.

SUMMARY

By analyzing the verification results, coverage and waveform are all achieved the expected requirements. In this paper, it has built a reusable verification platform of AES IP environment by UVM methodology, and verified a AES IP design. The UVM mechanism used is refered to as an override. This

functionality is useful to change sequence functionality by using the constrained random and directed tests. It makes verification more efficient and reusable. And using the verification platform with automatic generation can advance the speed of verification quickly and shorten the verification time. The results show that 100% functional coverage and waveform have been reached on the AES IP verification platform. According to the results of UVM verification platform, the AES IP design is successful.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (No. 61204040), and Beijing Municipal Natural Science Foundation (No. 4152004).

REFERENCES

1. TIAN Jin and WANG Xiao Li, Microeletronics & Computer, 29, p.86 (2012).
2. Michael Mefenza, Franck Yonga and Christophe Bobda, 15th International Microprocessor Test and Verification Workshop, p.16 (2014).
3. Zhang Jie and ZHU Li Juan, Software Guide, p.95 (2007).
4. Zhao Xuemei, Natural Sciences, 24 p.105 (2010).
5. Zhou Jun and Chang Guofeng, Bulletin of Science and Technology, 28, p.70.
6. He Dong Ming, China Integrated Circult, p.74 (2015).
7. Martin Keaveneyt and Anthony McMahont, ISSC 2008, p.325(2008).