An AES-based Intellectual-Property Identification in System-on-a-Chip Design

P.G.Gopinath
Dept., of ECE,
SIETK,Puttur,
A.P- 517583, India.

K.Adithya
Dept., of ECE,
SIETK,Puttur,
A.P- 517583, India.

B.Ajay
Dept., of ECE,
SIETK,Puttur,
A.P- 517583, India.

ABSTRACT

This paper presents a novel intellectual-property (IP) identification scheme using the existing System-on-a-Chip (SOC) watermarking design. An efficient and novel principle is established for IP identification which depends on the current IP design flow. The principle is embedding different Advanced Encryption Standard (AES) encoders in to System-on-a-Chip (SOC) based watermarked devices at behavior design level. Here the AES encoders provide additional security to the previously generated watermark sequences. This method is efficient as it survives synthesis, placement, and routing and can identify the IP at various levels. It may be easy to detect the identification of the provider even after the chip has been manufactured. The proposed method increases security for the System-on-a-Chip (SOC) based IP identification and protection scheme.

Keywords

Intellectual-Property (IP) identification, system-on-a-chip (SOC), very large scale integration (VLSI) design, watermarking, Advanced Encryption Standard (AES).

1. INTRODUCTION

An historic technological change in the integrated-circuit (IC) design complexity has resulted from the rapid increase in semiconductor processing technology [1], [2]. It poses challenges to the system designers' assumptions about performance being the design bottleneck. Other factors that are hiking into designers' top wish list more complex processors and architectures, larger code size, more complicated functionalities, less power consumption, lighter and smaller devices, shorter time-to-market, lower cost, etc. Due to increase in the silicon capacity and evolution of new fabrication technologies, it is possible to make systems on a single chip of silicon (System-On-a-Chip). As the design complexity goes up, we should expect lengthy design cycle. But what we get in actually is the time-to-market stress. The gap between silicon capacity and design productivity seems to be broadening at an even greater pace, making the growth of the semiconductor industry sluggish. To boost the productivity and to lessen the time to market, the reuse of previous modules is gaining importance and thus new models are drastically plummeting. Practically, Reuse-based and intellectual property (IP)-based design methodologies are extensively used very large scale integration (VLSI) design flow in IC industries [1].

The development of IP-identification techniques has been directed by Reuse-based and intellectual property (IP)-based design methodologies. IP identification should provide the design information, ownership rights, ownership proof, and IP management. After the IP has been incorporated into a whole chip and packaged, designers will be able to check the identity of the IP. An efficient and effective IP-identification technique should take in several characteristics such as ability to identify the IP at any design level, Low overhead, Low tracking cost Proof of identification. There are several approaches for IP identification in the literature. One possible method for stating ownership is to use watermarks. Watermarking is a technique that is conventionally used to safely identify the authenticity of the source of image, video, or audio media [3]–[8].

There are several techniques have been proposed for watermarking based IP identification. Kahng *et al.* [9], [10], [11] developed the protocols for IP watermarking at the physical design level, using the concept of constraint-based watermarking. One of the pilot approaches for IP watermarking is the constraint-based IP watermarking. This method encodes a user's digital identification as a set of additional design limitations. Then, the scheme adds the constraints into the original design specification, with the help of a tool that retrieves the final optimized design specification. Narayan *et al.* [12] proposed a process for embedding a watermark by modifying the number of vias or bends utilized to route the nets in a design. All these techniques embed the watermark at the physical design stage [10], [13]–[16], design partitioning [17], [18], and combinational logic synthesis [15]. After synthesis, placement, or routing, the layout of the soft IP core will be modified. These techniques are, therefore, inadequate in proving the identity of the soft IP core. Further, to check the identification we must look at the photomicrograph. These methods are not only making difficult but are also not convenient. Once the chip has been packaged, it is not very easy to distinguish the identity of the IP provider.

Oliveira [19] and Torunoglu and Charbon [20] proposed two diverse techniques to design watermarking circuit. The former adds new input/output sequences to the finite-state machine (FSM) representation of the design. While the latter initiated the FSM watermarking approach that extracts the idle transitions in a state transition graph of the behavioural model. This approach starts with constructing the FSM representation of the sequential design, then visiting every state and finding the unused state transitions (input/output pairs). When the IP is incorporated into a whole chip, the user runs into difficulty in tracking the FSM function. The watermark is hidden in the SOC after the chip has been packaged. The identifications are not easy to prove. Chapman *et al.* [21], [22] presented a digital-signal processing (DSP) watermarking scheme. In this method the watermark will be introduced to both algorithmic and architectural levels, to achieve more robustness. This approach is based on using different structures of the filter building block according to the distinct bits needed to be embedded.The designer (of a high-level digital filter) should encode one character as the hidden watermark data in this approach [23]. This design does not have an apparent way to track and take out the watermark at lower levels [23]. The watermark must be designed case by case according to the identification of various IPs. This is not convenient.

Yu-Cheng Fan [24], presented a watermarking scheme based on Testing-based system-on-chip (SOC) design. This concept incorporates Watermarking Generating circuit (WGC) and Test-Circuit (TC) in to the Intellectual Property (IP). After integrating the IPs into the full SOCs, the only signal in the IP that can be traced is the test signal. In the test mode, the selected IP sends output test patterns and watermark sequences in a predefined patterns. The identity of the IP provider will be known according to the watermark sequence sent by the IP. This method has low hardware costs (no more than 5%), low tracking costs, low processing time (PT) costs, and high fault coverage (between 90% and 96%).

2. ADVANCED ENCRYPTION STANDARDS (AES)

AES is the Advanced Encryption Standard, a United States government standard algorithm for encrypting and decrypting data. The standard is described in Federal Information Processing Standard (FIPS). On January 2, 1997, The National Institute of Standards and Technology (NIST) published a request for development of a Federal Information Processing Standard for Advanced Encryption Standard which sought to offer a higher level of security than that offered by the Data Encryption Standard (DES), which grew vulnerable to brute-force attacks due to its 56-bit key length. AES candidates were required to implement a symmetric block cipher that supported multiple key lengths and algorithm had to be publicly defined, free to use, and able to run efficiently in both hardware and software. Five finalists were chosen whose details are tabulated in Table 1.

Table 1: Finalists of AES Competition

Name	Author	Type
MARS	IBM	Extended Feistel Network
RC6	RSA	Feistel Network
Rijndael	Joan Daemen and Vincent Rijmen	Substitution Permutation Network
Serpent	Ross Anderson, Eli Biham, and Lars Knudsen	Substitution Permutation Network
Twofish	Bruce Schneier, John Kelsey, Niels Ferguson, Doug Whiting, David Wagner, and Chris Hall	Feistel Network

2.1 Rijndael Algorithm

Rijndael algorithm [25] is the winner of the AES competition conducted by the NIST. It is a symmetric block cipher with a variable block size. Key lengths can be 128, 192, or 256 bits; called AES-128, AES-192, and AES-256, respectively. AES- 128 uses 10 rounds, AES-192 uses 12 rounds, and AES-256 uses 14 rounds. However, AES merely allows a 128 bit data length that can be divided into four basic operation blocks. These blocks operate on array of bytes has four rows, the number of columns is denoted by N_b and is equal to the block length divided by 32 which is called the state. For full encryption, the data is passed through Nr rounds (Nr = 10, 12, 14) corresponding to their key length. These rounds are governed by the following transformations:

2.1.1 Bytesub transformation

This is a non linear byte Substitution, using a substation table (s-box), which is constructed by multiplicative inverse and affine transformation.

2.1.2 Shiftrows transformation

This is a simple byte transposition, the bytes in the last three rows of the state are cyclically shifted. The offset of the left shift varies from one to four bytes depending on the block size.

2.1.3 Mixcolumns transformation

This is equivalent to a matrix multiplication of columns of the states. Each column vector is multiplied by a fixed matrix. It should be noted that the bytes are treated as polynomials rather than numbers.

2.1.4 Addroundkey transformation

In this operation a simple XOR is done between the working state and the round key. The Round Key is derived from the Cipher Key by means of the key schedule. The Round Key length is equal to the block length Nb. This transformation is its own inverse.

The encryption procedure consists of several steps. After an initial addroundkey, a round function is applied to the data block (consisting of bytesub, shiftrows, mixcolumns and addroundkey transformation, respectively). It is performed iteratively (Nr-1 times) depending on the key length. In the final round the mixcolumns transformation is not done and rest of order is followed. The decryption structures have exactly the same sequence of transformations as the one in the encryption structure but are implemented in reverse order. The transformations

Inv-Bytesub, the Inv-Shiftrows, the Inv-Mixcolumns, and the Addroundkey allow the form of the key schedules to be identical for encryption and decryption.

2.2 Serpent Algorithm

Serpent algorithm [26] is a Substitution Permutation (SP) network operating on four 32 bit words giving total block size of 128 bits. It consists of an initial permutation which performs bit slicing by taking each and every bit and then packing them in to four 32-bit words. Then 32 rounds of operations are done. Here the output of the initial permutation is then XOR's with first round key and then passes it through the 32 copies of the first S-box where first four bits are passed through the first copy of the S-box to form a four bit intermediate vector. Then the next four bits are passed through the next copy of the S-box and returns next four bits and so on. After each round of operation the next S-box of eight different S-boxes are used. After the eighth round, then S-boxes are repeated again from the first. Then the each 32 bits in

each of the output words are linearly mixed by a predefined sequence of register operations. Following it an inverse of the initial permutation was performed.

2.3 Two Fish Algorithm

Two fish algorithm [27] is a 128-bit block cipher that accepts a variable-length key up to 256 bits. The cipher is a 16-round Feistel network with a bijective *F* function made up of four key-dependent 8-by-8-bit S-boxes, fixed 4-by-4 maximum distance separable matrix over GF(2^8), a pseudo-Hadamard transform, bitwise rotations, and a carefully designed key schedule. Two Fish algorithm contains following blocks:

2.3.1 Feistel Networks

A Feistel network is a general method of transforming any function (usually called the *F* function) into a permutation. The fundamental building block of a Feistel network is the *F* function: a key-dependent mapping of an input string onto an output string. An *F* function is always non-linear and possibly non-surjective1:

$$F: \{0,1\}^{n/2} \times \{0,1\}^{N} \rightarrow \{0,1\}^{n/2}$$

where *n* is the block size of the Feistel Network, and *F* is a function taking *n/*2 bits of the block and *N* bits of a key as input, and producing an output of length *n/*2 bits.

2.3.2 MDS Matrices

A maximum distance separable (MDS) code over a field is a linear mapping from *a* field elements to *b* field elements, producing a composite vector of *a + b* elements, with the property that the minimum number of non-zero elements in any non-zero vector is at least *b*+ 1. MDS mappings can be represented by an MDS matrix consisting of $a \times b$ elements. A necessary and sufficient condition for an $a \times b$ matrix to be MDS is that all possible square sub matrices, obtained by discarding rows or columns, are non-singular.

2.3.3 Pseudo-Hadamard Transforms

A pseudo-Hadamard transform (PHT) is a simple mixing operation that runs quickly in software. Given two inputs, *a* and *b*, the 32-bit PHT is defined as:

$$a' = a + b \bmod 2^{32}$$
$$b' = a + 2b \bmod 2^{32}$$

2.3.4 Whitening

Whitening is the technique of XORing key material before the first round and after the last round. Whitening substantially increases the difficulty of key search attacks against the remainder of the cipher. Two fish eXORs 128 bits of sub key before the first Feistel round, and another 128 bits after the last Feistel round. These sub keys are calculated in the same manner as the round sub keys.

2.3.5 Key Schedule

The key schedule is the means by which the key bits are turned into round keys that the cipher can use. Two fish needs a lot of key material, and has a complicated key schedule. To facilitate analysis, the key schedule uses the same primitives as the round function.

Two fish uses a 16-round Feistel-like structure with additional whitening of the input and output. The plaintext is split into four 32-bit words. In the input whitening step, these are xored with four key words. This is followed by sixteen rounds. In each round, the two words on the left are used as input to the *g* functions. (One of them is rotated by 8 bits first.) The *g* function consists of four byte-wide key-dependent S-boxes, followed by a linear mixing step based on an MDS matrix. The results of the two *g* functions are combined using a Pseudo- Hadamard Transform (PHT), and two keywords are added. These two results are then XORed into the words on the right (one of which is rotated left by 1 bit first, the other is rotated right afterwards). The left and right halves are then swapped for the next round. After all the rounds, the swap of the last round is reversed, and the four words are XORed with four more key words to produce the cipher text.

2.4 RC6 Algorithm

RC6 [28] is a feistel network and more accurately specified as RC6-w/r/b where the word size is w bits, encryption consists of a nonnegative number of rounds r, and b denotes the length of the encryption key in bytes. RC6-w/r/b operates on units of four w-bit words using the basic operations like 2^w modulo integer addition, 2^w modulo integer subtraction, 2^w modulo integer multiplication, bitwise exclusive-or of w bits, left and right rotations. RC6 works with four w-bit registers A,B,C,D which contain the initial input plaintext as well as the output cipher text at the end of encryption. The first byte of plaintext or cipher text is placed in the least-significant byte of A and the last byte of plaintext or cipher text is placed into the most-significant byte of D.

Those w-bit registers are then subjected to basic operations and their outputs which is cipher text is again stored in the registers. During decryption the exact inverse operations of the encryption are performed to obtain the plain text.

2.5 MARS Algorithm

MARS [29] is a shared-key block cipher, with a block size of 128 bits and a key size of 128 bits. It was designed to meet and exceed the requirements for a standard shared-key encryption. It takes four 32-bit words plaintext as input and produces four 32-bit words ciphertext as output. The cipher itself is word-oriented, in that all the internal operations are performed on 32-bit words, and hence the internal structure is endian-neutral. When the input (or output) of the cipher is a byte stream, a little endian byte ordering to interpret each four bytes as one 32-bit word.

The cipher consists of a "cryptographic core" of keyed transformation, which is wrapped with two layers of cryptographic cores providing rapid key avalanche. The first phase provides rapid mixing and key avalanche, to frustrate chosen-plaintext attacks, and to make it harder to "strip out" rounds of the cryptographic core in linear and differential attacks. It consists of addition of key words to the data words, followed by eight rounds of S-box based, un-keyed type-3 Feistel mixing (in "forward mode"). The second phase is the "cryptographic core" of the cipher, consisting of sixteen rounds of keyed type-3 Feistel transformation. To ensure that encryption and decryption have the same strength, we perform the first eight rounds in "forward mode" while the last eight rounds are performed in "backwards mode". The last phase again provides rapid mixing and key avalanche, to protect against chosen-ciphertext attacks. This phase is essentially the inverse of the first phase, consisting of eight rounds of the same type-3 Feistel mixing as in the first phase (except in "backward mode"), followed by subtraction of key words from the data words.

3. AES BASED IP IDENTIFICATION

In this paper IP identification is performed using System-on-a-Chip (SOC) design and AES based encryption algorithms. Fig.1 shows the flow chart of the IP identification using AES algorithms in SOC design. The watermarking scheme used in this paper is an SOC based design as in [24]. The watermarking scheme [24] is made more secured using the AES encryption algorithms. The Watermark Generating Circuit (WGC) which generates the encoded watermark sequences and Random sequence generator circuit which generates the random sequences that are used to combine with WGC output are constructed.

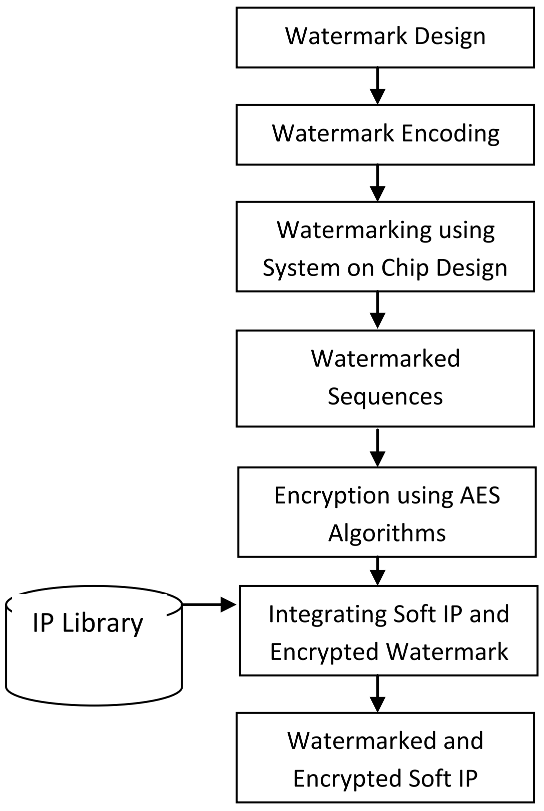
Watermark Design
Watermark Encoding
Watermarking using System on Chip Design
Watermarked Sequences
Encryption using AES Algorithms
IP Library
Integrating Soft IP and Encrypted Watermark
Watermarked and Encrypted Soft IP

Figure.1. Flow chart for AES based IP protection using SOC design

The test sequences which are combination of watermark sequences and random sequences are produced by different integrating methods as shown in [24]. Before this unit being integrated in to the Soft IP present in the IP library, the test sequences are encrypted using AES algorithms Rijandeal, Serpent, Twofish, RC6 and MARS. Encryption of test sequences through AES was done by a key which is known only to IP provider. The length of the key may be of 128, 192, 256 bits. The test sequences are passed through the AES encoders and an encrypted test sequences are obtained. The watermarking scheme is operated using a Test Mode signal. The Test Mode signal specifies the operation of the soft IP, whether the normal operation of the IP was to be performed or the encrypted test sequences has to be provided. Whenever the identity is to be checked, then test mode signal was operated accordingly and the obtained encrypted sequences are decrypted by the key with the IP provider to get original test sequences. These test sequences are then used to extract the watermark to provide IP identification. The extraction of the watermark was done by using the knowledge of the method of integrating the WGC and the Random sequence generator as in [24].

Table 2: Watermarking Characteristics

	Rijndael	Serpent	Twofish	RC6	MARS
No of Logic Gates Used	2831	5587	2458	4070	12021
Power Consumption	1363.82	1360.82	1357.82	1351.82	1381.82
Processing Time	5:01	9:53	7:34	7:01	36:10
Maximum Delay	11.084	13.904	13.304	13.555	13.079

4. EXPERIMENT RESULTS

This watermarking scheme was synthesized using the Xilinx ISE 13.2. The entire soft IP was implemented in a Virtex 6 FPGA. The IP core and the AES algorithms are written in VHDL. The aspects of the watermarked IP core encrypted with AES algorithms are given in Table 2. The aspects that are analysed are number of logic gates used, Processing time which required for the synthesis tool to synthesize the circuit (in minutes: seconds), Power consumption (in mW) and Maximum delay experienced by the signal to get transferred to the output(in nanoseconds).

5. CONCLUSION

In this paper, an AES based watermarking scheme for IP identification using System-on-a-Chip (SOC) design is presented. The identity of the IP was protected using the watermark sequences that are encrypted by the AES algorithms. The AES algorithm gives the encryption to the watermarked sequences that are generated by using different watermarking schemes. The watermark is a general-purpose design methodology that does not need to be designed case by case according to various IPs. The watermark function is not changed after logic synthesis, placement, and routing because the watermark is embedded into the IP at the behavioural design level. It is still easy to detect the identity of the IP designer after the chips have been packaged, if the key is known. This method entails low hardware overhead and processing time. By selecting the required parameters any one of the AES algorithms will serve the maximum IP protection using this method. Until now there have been few papers that has discussed this problem. The complete DRM platform for SOC/VLSI IP is worth researching.

6. REFERENCES

[1] H. Chang, *Surviving the SOC Revolution: A Guide to Platform-Based Design*. Norwell, MA: Kluwer, 1999.

[2] G. Martin and H. Chang, *Winning the SoC Revolution: Experiences in Real Design*. Norwell, MA: Kluwer, 2003.

[3] I. J. Cox, M. L. Miller, and J. A. Bloom, *Digital Watermarking*.San Mateo, CA: Morgan Kaufmann, 2002.

[4] I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," *IEEE Trans. Image Process.*, vol. 6, no. 12, pp.1673–1687, Dec.1997.

[5] Y. P. Wang, M. J. Chen, and P. Y. Cheng, "Robust image watermark with wavelet transform and spread spectrum techniques," in *Proc. Asilomar Conf. Signals Syst., Comput.*, Nov. 2000, vol. 2, pp. 1846–1850.

[6] Q. Sun and S. F. Chang, "Semi-fragile imageauthentication using generic wavelet domain features and ECC," in *Proc.Int. Conf. Image Process.*, Sep. 2002, vol. 2, pp., 901–904.

[7] Y. L. Ho and H. C. Wang, "An audio watermarking algorithm based on significant component modulation," in *Proc. IEEE Int. Conf. Consum. Electron.*, Jun. 2003, pp. 212–213.

[8] T. H. Tsai and C. Y. Wu, "An implementation of configurable digital watermarking system in MPEG video encoder," in *Proc. IEEE Int. Conf. Consum. Electron.*, Jun. 2003, pp. 216–217.

[9] A. B. Kahng, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H.Wang,and G. Wolfe, "Robust IP watermarking methodologies for physical design,"in *Proc. IEEE Des. Autom. Conf.*, Jun. 1998, pp. 782–787.

[10] A. B. Kahng *et al.*, "Constraint-based watermarking techniques for design IP protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 10, pp. 1236–1252, Oct. 2001.

[11] A. B. Kahng, D. Kirovski, S. Mantik, M. Potkonjak, and J. L. Wong, "Copy detection for intellectual property protection of VLSI designs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 1999, pp. 600–604.

[12] N. Narayan, R. D. Newbould, J. D. Carothers, J. J. Rodriguez, and W. T. Holman, "IP protection for VLSI designs via watermarking of routes," in *Proc. IEEE Int. Conf. ASIC/SOC*, Sep. 2001, pp. 406–410.

[13] D.Kirovski and M. Potkonjak, “Localized watermarking: Methodology and application to template mapping,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.,* Jun. 2000, vol. 6, pp. 3235–3238.

[14] A. E. Caldwell, H. J. Choi, A. B. Kahng, S. Mantik, M. Potkonjak, G. Qu, and J. L. Wong, “Effective iterative techniques for fingerprinting design IP,” in *Proc. Des. Autom. Conf.*, Jun. 1999, pp. 843–848.

[15] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, “Fingerprinting digital circuits on programmable hardware,” in *Proc. Int. Workshop Inf. Hiding*, 1998, pp. 16–31.

[16] E. Charbon, “Hierarchical watermarking in IC design,” in *Proc. IEEE Custom Integr. Circuits Conf.*, May 1998, pp. 295–298.

[17] G. Wolfe, J. L. Wong, and M. Potkonjak, “Watermarking graph partitioning solutions,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 10, pp. 1196–1204, Oct. 2002.

[18] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe,“Watermarking techniques for intellectual property protection,” in *Proc. IEEE Des. Autom. Conf.*, 1998, pp. 776-781.

[19] L. Oliveira, “Techniques for the creation of digital watermarks in sequential circuit designs,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 9, pp. 1101–1117, Sep. 2001.

[20] I.Torunoglu and E. Charbon, “Watermarking-basedcopyright protection of sequential functions,” in *Proc IEEE Custom Integr. Circuits Conf.*, 1999, pp. 35–38.

[21] R. Chapman and T. S. Durrani, “IP protection of DSP algorithms for system on chip implementation,” *IEEE Trans. Signal Process.*, vol. 48, no. 3, pp. 854–861, Mar. 2000.

[22] R. Chapman, T. S. Durrani, and A. P. Tarbert, “Watermarking DSP algorithms for system on chip implementation,” in *Proc. IEEE Int. Conf. Electron., Circuits Syst.*, 1999, pp. 377–380.

[23]A. T. Abdel-Hamid, S. Tahar, and E. M. Aboulhamid, “IP watermarking techniques: Survey and comparison,” in *Proc. IEEE Int. Workshop System-on-Chip Real-Time Appl.*, 2003, pp. 60–65.

[24]Yu-Cheng Fan, “Testing-Based Watermarking Techniques for Intellectual-Property Identification in SOC Design,” *IEEE Trans. Instrumentation AND measurements*, vol. 57, no. 3, march 2008

[25]Joan Daemen, Vincent Rijmen,”*AES Proposal: Rijndael*”.

[26] Ross Anderson, Eil Biham, Lars Knuden,” *Serpent: A Flexible Block Cipher With Maximum Assurance*”.

[27] Bruce Schneier_ John Kelseyy Doug Whitingz David Wagnerx Chris Hall Niels Ferguson, “*TwoFish: A 128- Bit Block Cipher*”

[28] Ronald L. Rivest, M.J.B. Robshaw, R. Sidney, and Y.L. Yin,”*The RC6 Block Cipher*”.

[29] C Burwick, D Coppersmith, E D’Avignon, R Gennaro, S Halevi, C Jutla, S M Matyas, L O’Connor, M Peyravian, D Safford *et al.*“*The Mars Encryption Algorithm*”.