An Internet-based IP Protection Scheme for Circuit Designs using Linear Feedback Shift Register(LFSR)-based Locking

Raju Halder
Department of Computer Science and Engineering
University of Calcutta
Kolkata 700009, India
halder_raju@yahoo.com

Parthasarathi Dasgupta
Indian Institute of Management Calcutta
Kolkata 700104, India
partha@iimcal.ac.in

Saptarshi Naskar
Department of Computer Science and Engineering
University of Calcutta
Kolkata 700009, India
sapgrin@gmail.com

Samar Sen Sarma
Department of Computer Science and Engineering
University of Calcutta
Kolkata 700009, India
sssarma2001@yahoo.com

ABSTRACT

IP reuse is rapidly proliferating recent automated circuit design. It is facing serious challenges like forgery, theft and misappropriation of intellectual property (*IP*) of the design. Thus, protection of design *IP* is a matter of prime concern. In this paper, we propose a novel Internet-based scheme to tackle this problem. Input to the proposed scheme is a graph corresponding to a digital system design. Watermarking of the graph and its encryption are achieved using a new linear feedback shift register(*LFSR*)-based locking scheme. The proposed scheme makes unauthorized disclosure of valuable design almost infeasible, and can easily detect any alteration of the design file during transmission. It ensures authentication of the original designer as well as non-repudiation between the seller and the buyer. Empirical evidences on several benchmark problem sets are encouraging.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]; E.3 [**Data Encryption**]

General Terms

Design, Experimentation, Security

Keywords

Intellectual property protection (IPP), Watermarking, Encryption, Decryption

1. INTRODUCTION

In this Section, we introduce *VLSI IP* protection, briefly describe the existing works in this area, and the contribution of our work.

1.1 IP Protection

Recent *VLSI* circuit design involves more integration on a single chip within a shorter design cycle. Design reuse has been incessantly encouraged by the pressures to reduce time to market, better design productivity, and others. Circuit components, in electronic form, signify intellectual property (*IP*) of *VLSI* design [6]. Unfortunately, unlimited and careless design reuse may lead to infringement of *IP* [6, 1], and enhance the possibility of certain typical attacks such as:

- interception and recovery of original design
- malicious attempt to randomly modify the design
- denial of validity of a legal purchaser by the seller [15]
- claiming false *IP* ownership
- illegal reselling by a purchaser without paying proper royalty to the designer
- attempt to recover original design [2] by a buyer from its modified version prepared for another buyer.

Since the design of a circuit is extremely expensive, it is essential that it should not be vulnerable to any of these attacks. Several works on the use of encryption and watermark embedding of circuits have been reported so far, as discussed in the next sub-section.

Major contributions of our work include: (i) it is Internet-based, (ii) it is quite robust due to the use of a strong encryption algorithm, and an easily verifiable, but hard to tamper watermark embedding in the design, (iii) it is generic as the input is considered to be a representative graph of a digital system, and (iv) it makes use of Linear-Feedback Shift Registers (*LFSR*) for achieving the objectives of protecting the design. In general, the proposed approach can be used for any type of graph.

1.2 Related work

IP protection (*IPP*) in *VLSI* design has been studied in recent past. In hierarchical watermarking, topological information of a design is uniquely mapped into topological signature [5]. Robust *FPGA IP* protection [10] proceeds through embedding distinct set of multiple watermarks for different recipients in the *LUT*s of *FPGA*. Watermarking techniques for *IP* protection in *FPL* devices have also been reported in [8]. On the other hand, constraint-based watermarking [1] maps signature of the designer into a set of additional constraints, consequently shrinking the optimal solution space. [3] discusses zero-overhead watermarking on *FPGA* by applying timing constraint on signal delays. [11],[17],[14] emphasize on protection of design tool instead of the design itself. [4] was the first proposed scheme for watermark embedding at the physical design level.

1.3 Motivation of the work

IP protection scheme is typically judged by several parameters, such as its strength against any attempt to infringe *IP* without incurring much overhead, the ease of watermark detection, and so on. None of the existing *IP* protection techniques can ensure complete *IP* security in terms of all these parameters. Most of the existing watermarking techniques also fail to handle repudiation attack. The tradeoffs between the solution quality and the strength of the watermark, and the lack of correlation between size of watermark and reduction in solution space motivate us to adopt a new viewpoint for *IP* protection. Selling of design tool instead of design itself, is more risky as indirect *IP* protection techniques can not prevent *IP* infringement. However, any misuse of *IP* tool, without being noticed and detected, causes huge loss of royalty to *IP* owner. This warrants watermarking of design tools. As discussed in [7], an *IP* tool may be supplied by a vendor in the following two ways:

- selling design tool to customer
- running the design tool at vendor's end with design parameters from the customers and supplying the customer only with required design.

In the former case, if the customer uses the tool to generate new design for a third party, genuine creator suffers from royalty loss. This can be controlled through watermarking. In the latter case, unless the channel of transmission is highly secured, the high quality design may be hacked, and hence the design needs to be encrypted. The Internet has yielded new opportunities for the creation and delivery of content in digital forms. We propose an Internet-based *IP* protection scheme, where the design tool is split into two modules: first module is executed at the creator's end, generating a watermarked and encrypted intermediate design, which is transmitted electronically. Second module of the tool is executed at the buyer's end generating the final design, only after the legitimate buyer decrypts the intermediate design using the private key. A similar scheme has already been proposed in [7]. A cryptosystem [15] using private or public keys almost eliminates the use of ultra-secure key transmission. This paper also proposes an effective watermarking scheme to protect the *IP* rights of the authentic designer. A system such as a digital computer, or *VLSI* circuit, may be defined as a collection of objects, connected to form a coherent entity with a well-defined function. A natural and effective way of modeling such a system is the use of a graph [9]. In the proposed scheme, the core concept involves encryption obtained by changing the interconnection patterns in the graph with the help of a *LFSR* [13]. The seller uses two *LFSR*s: one for watermarking, known to seller only, and the other for encryption-cum-decryption, known to both the seller and the buyer (*through some prior agreement between the two parties*). During watermark generation, a mask bit string is also generated. This mask string is provided to the buyer only on demand for watermark verification. The file which is transmitted from the seller to the buyer contains two components, viz., the encrypted graph, and the watermark. If the encrypted graph is tampered, the hash value can be used to detect it. However, an intruder might like to replace the watermark with his own watermark using his own *LFSR*. Since the watermark is characterized by a polynomial $g(x)$ (obtained from original graph) and a signature S (obtained from both seller and buyer), such replacement is not possible without knowing the $g(x)$ for the original (un-encrypted) graph.

The rest of the paper is organized as follows: Section 2 discusses the proposed schemes for creation and verification of the watermark, and Section 3 illustrates the methods of encryption and decryption. Section 4 discusses the empirical observations, and Section 5 concludes the paper.

2. WATERMARK CREATION AND VERIFICATION

The proposed scheme works in two phases:

- Embedding watermark in the input graph G considering signatures of both the seller and the buyer.
- Encrypting the watermarked graph G_m, say, with the help of a linear feedback shift register (*LFSR*) by inserting some new edges in and deleting some existing edges from G_m.

An alternative scheme could be to generate the complete design, encrypt the design file, embed watermark in it, and transmit it through a secure channel. However, this scheme may not be feasible due to the huge volume of the design files, and associated risk. The proposed watermarking scheme is based on identity of both the seller and the buyer of the *IP*. The buyer can verify the signature of the seller to make sure that the product is being purchased from the legal source. The seller can trace the buyer in case of illegal reselling. Moreover, in order to check for any doubt of illegal design usage, hidden signatures of the valid seller and buyer of a design *IP* may be retained.

2.1 Watermark creation

For a particular design graph G, for each pair of source and destination, the watermark uniquely identifies the pair. The seller is free to choose a *LFSR* privately of length L, say, and any one of the primitive feedback connections. This *LFSR* and a technique of shift register polynomial division using it ensures uniqueness and robustness of the generated watermark. Figure 1 shows a *LFSR* with a feedback loop.

A polynomial $g(x)$ is generated from G as described later. This polynomial characterizes the graph G. However, for a given $g(x)$, the construction of a unique G is not possible *i.e.* it is one-way function. The polynomial $g(x)$ is applied at the input of *LFSR* that is initialized with zeros. Let $q(x)$ and

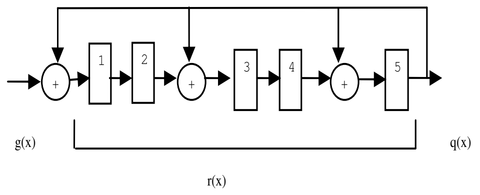

Figure 1: A LFSR with a feedback loop

$r(x)$ be the quotient and remainder respectively obtained from the $LFSR$. Let g_c, q_c, and r_c respectively denote the binary coefficients of polynomials $g(x)$, $q(x)$ and $r(x)$ respectively. To generate a watermark, the seller embeds a signature S into polynomial $g(x)$, and obtains a binary coefficient g_{cs} in the following manner: concatenate the ASCII code of each character in S to generate a binary string S_c. Now, g_{cs} is obtained from $g_{cs} = g_c \bigotimes S_c$. S may be signature of the seller or the buyer, or combination of the two (*for source and destination based watermarking*). Next, g_{cs}, q_c, and r_c are concatenated to generate a unique watermark W (*see Lemma 1*). A mask M is also generated along with W to be used in the verification phase. Generation of W and M are according to the Equations 1 and 2.

$$W = arb_0 + q_c + arb_1 + r_c + arb_2 + start\_bit + g_{cs} + end\_bit + arb_3 \quad (1)$$

where '+' denotes the concatenation operator of bits, arb_i, $i = 0$ to 3 denotes some arbitrary bit strings of appropriately chosen lengths. The use of these bit strings make it impossible for an intruder to guess g_{cs}, q_c, and r_c (*so as to guess $LFSR$ and G*). Each of *start_bit* and *end_bit* is a '1' used to indicate the begin and end of g_{cs}.

$$M = arb_0 + q_c + arb_1 + r_c + arb_2 + \{0\}^* + arb_3 \quad (2)$$

$\{0\}^*$ represents a string of 0s used in generating mask M, and is of length ($start\_bit + g_{cs} + end\_bit$). The binary values of W and M are then converted into their respective decimal values. A formal description of the proposed algorithm for watermark creation appears in figure 2.

It may be noted that the use of $LFSR$s ensures uniqueness of the watermark. Thus, for the same input graph, use of different $LFSR$s yield different watermarks. Lemma 1 summarizes the utility of $LFSR$s.

LEMMA 1. *For any two graphs G_1 and G_2, if the watermarks W_1 and W_2 are created using different $LFSR$s, then $W_1 \neq W_2$.*

The proof is omitted for brevity.

2.2 Watermark verification

In order to verify the watermark embedded by the seller in a design, the buyer need to use the mask M transmitted in the encrypted file. $W \bigotimes M$ is computed to find the value of the string "*start_bit* + g_{cs} + *end_bit*", from which the value of g_{cs} can be easily obtained. Buyer may now extract the values of $g(x)$, and signature S applying same procedure as the seller, for verification of the authenticity of the design. A formal description of the algorithm for watermark verification is shown in Figure 3.

We now discuss the technique for generating $g(x)$ from the input graph.

```
Algorithm 1:   Create_Watermark
Input: System Graph $G$ and $LFSR$ of length $L$ with primitive
feedback loop, Signature $S$
Output: Watermark $W$ and mask $M$
1. Generate a polynomial $g(x)$ from $G$
2. Apply $g(x)$ to the input terminal of $LFSR$(initialized with
zeros) to produce quotient $q(x)$ and remainder $r(x)$
3. Represent signature $S$ into binary string $S_c$ by concatenating
the ASCII code of each character in $S$
4. Compute $g_{cs} = S_c \bigotimes g_c$
5. Create $W$ by concatenating $g_{cs}$, $q_c$, $r_c$ and $arb_i$, i=0,...,3
6. Create corresponding mask $M$
```

Figure 2: Algorithm for Watermark creation

```
Algorithm 2:   Verify_Watermark
Input: System Graph $G$, Mask $M$, and watermark $W$
Output: Ownership claim as true or false
1. Generate polynomial $g(x)$ from $G$
2. Compute $g_{cs}$ from $W \bigotimes M$.
3. Obtain signature $S$ from $g_c \bigotimes g_{cs}$.
4. If signature $S$ matches with the original signature transmitted
by the seller then claim := true
5. else claim := false
```

Figure 3: Algorithm for Watermark verification

2.2.1 *Generating a Polynomial from a given graph*

Let G represents the input graph. Count the number of vertices of G having identical degrees, in decreasing order of their degrees. The degree is used to denote the power and count contributes to the coefficient of the corresponding term. Since the coefficients of the polynomial are binary, apply $mod-2$ operation on the counts to obtain the binary coefficients. To ensure that the polynomial is monic, we enforce the coefficient of the highest degree term to be 1.

2.3 Version Control

The proposed watermark creation scheme assigns a unique watermark with each input design graph G, and uniquely identifies a source-destination pair for G. A privately chosen $LFSR$ is used to make W robust and unique. Suppose the seller wants to sell different versions of the same G to more than one buyer. As S represents the combined signature of both seller and buyer, S will vary for same seller and different buyer, and this change is embedded in W. But to control the different versions of the same design graph for the buyers, the seller can use different private $LFSR$s to generate different unique watermark and corresponding mask. The above version control mechanism generates robust watermark and corresponding mask for the following reasons:

- $LFSR$ is hidden and known to the seller only.
- Different versions use different $LFSR$, reducing chances of duplication.

3. ENCRYPTION AND DECRYPTION SCHEMES

The proposed encryption method depends on the modification of a watermarked graph G_m with the help of a $LFSR$. The following Section introduces $LFSR$s, states some of its properties, and then justifies its use.

3.1 Introducing LFSR

Definition 1. An L-stage $LFSR$ is a maximum length $LFSR$ if some initial state will result in a sequence of a number of bits repeating at intervals of every 2^L - 1 bits. The sequence is called a m-sequence ($m = 2^L$ - 1) [13].

Some properties of $LFSR$s are as follows:

Property 1: The number of 0s and 1s in an m-sequence obtained from an L-stage maximum-length $LFSR$ are 2^{L-1} and $2^{L-1} - 1$ respectively and thus, differs by only one [13].

Property 2: For an m-sequence obtained from an L-stage maximum length $LFSR$, there is one run (maximal sequence) of L consecutive 1s and one run of L - 1 consecutive 0s. For L - 1 $< r <$ 0, there are $2^{L-(r+2)}$ runs of length r for 1s and the same number of runs of 0s [13].

In the following, we attempt to justify the use of $LFSR$ in the encryption of G_m. $LFSR$ is used to implement random number generators. Thus for a specific G, its modification with the use of the random sequence from a $LFSR$ generates a random graph. $LFSR$ can be simulated software or can be implemented in hardware. Multiple $LFSR$s may be combined for better security. Thus, the use of $LFSR$ increases the robustness of the proposed encryption method.

The following section describes the design of the encryption scheme using an L-stage $LFSR$. Hereafter, by $LFSR$, we refer to only maximum length $LFSR$.

3.2 The encryption scheme

We consider (i) A L-stage $LFSR$ at the seller's end, and (ii) A symmetric private key K_b of the seller. The $LFSR$ is first initialized with a seed of length L generated from K_b in the following way. Let the length of K_b be λ. We generate the seed (bit string) of the L-stage $LFSR$ by first converting K_b into compressed key K'_b and then choosing the first L-bits obtained from K'_b as the seed. We explain the method with an example. Let K_b = ABCDEFGHIJKLMN. Then, K_b is divided into some number of parts, each part having length = $\lceil \frac{L}{8} \rceil$ = 3, say. Thus, the parts obtained from K_b are (ABC), (DEF), (GHI), (JKL), (MN). Now K'_b = (b_1, b_2, ..., $b_{\lceil \frac{L}{8} \rceil}$) will be formed, where b_i is obtained by adding the ordinal values (1 for A) of the characters in these components. Thus, the first alphabet in K'_b is obtained by (i) taking the sum of ordinal values of A, D, G, J, and M = 1 + 4 + 7 + 10 + 13 = 35, and (ii) taking (mod 26 + 1) over the integer generated. Thus, first alphabet of K'_b = 35 mod 26 + 1 = 10 = 'J'. Similarly, the second and the third alphabets of K'_b are 'O' and 'E' respectively. Thus, starting with K_b = (a_1, a_2, ..., a_λ), we obtain K'_b = (b_1, b_2, ..., $b_{\lceil \frac{L}{8} \rceil}$). Next, the ASCII values of all the characters represented by b_j , j = 1, 2, ..., $\lceil \frac{L}{8} \rceil$ in K'_b are concatenated. This gives the binary string of length $\lceil \frac{L}{8} \rceil \times 8$ bits. If L is a multiple of 8, all bits of the binary string together is considered to be the seed (for initializing the $LFSR$), otherwise the first L bits of the binary string is considered as the seed. Next, the $LFSR$ is initialized with the seed value. It is clear to see that the seed can not have all 0s as shown in Lemma 2.

Lemma 2. *The method of generating the seed value as described above guarantees the presence of at least one 1 in the seed value for L $\geq$ 2.*

We skip the proof for brevity.

The maximum-length $LFSR$ outputs a m-sequence of length $2^L - 1$ which repeats itself. We just pick a binary string having length $2 \times \sqrt{\alpha} \times N$ from the output of the $LFSR$, where $0 < \alpha < 1$ and $N = \mid V_m \mid$, the number of vertices in the watermarked graph G_m. Clearly, this sequence is random, as it is part of the maximum length random (or pseudo-random) sequence of length 2^L - 1.

Lemma 3. *The sequence of length 2 $\times \sqrt{\alpha} \times N$, where 0 $< \alpha <$ 1 and $N = \mid V_m \mid$ which is taken from the m-sequence of a maximum-length LFSR of length L must be completely random.*

We skip the proof for brevity.

Now the positions of all 0s and 1s in that sequence of length $2 \times \sqrt{\alpha} \times N$ are marked. The positions of all 0s and all 1s respectively in the sequence form two sets of integers (*see Lemma 4*). Let these integers be the vertex numbers of the watermarked graph G_m. If in the two sets, any element $p \geq N$ ($N = \mid V_m \mid$), then the value of p mod N is computed. If the value p mod N does not exist in the set of integers, then, p is replaced with p mod N. Otherwise p is simply deleted from the set.

Lemma 4. *The sequence of length 2 $\times \sqrt{\alpha} \times N$, where 0 $< \alpha <$ 1 and $N = \mid V_m \mid$ must be greater than L (length of LFSR) to include at least one 1 and one 0 so as to enable the formation of two sets.*

We skip the proof for brevity.

Next, the Cartesian product Π of the two sets of integers is formed. Any pair in it corresponds to an edge to be used to modify the watermarked graph G_m. If any pair in Π corresponds to an edge in G_m, that edge is deleted from G_m. Otherwise the edge is inserted in G_m. This forms a modified graph G'_m.

In the next step of encryption, at the sender's end, a hash value of G_m is generated, which is transmitted along with the encrypted graph G'_m. Lemma 5 shows that applying the encryption algorithm twice on an input graph will yield the original graph only.

Lemma 5. *Assuming encryption of a graph G_m to be a function $E(G_m)$, $E(E(G_m)) = G_m$.*

We skip the proof for brevity.

Since the encryption scheme uses a private key, for verification at the buyer's end, a hash value for the watermarked graph is also transmitted from the seller. Hashing is based on Merkle-Damgard's Meta method [12]. For generating the hash value of G_m, G_m is first represented as an adjacency matrix containing 0s and 1s. The adjacency matrix is then converted into a binary string by concatenating all its rows (or all its columns), and Merkle-Damgard's Meta method applied on this binary string to generate the hash value of G_m.

3.3 Recovery of original graph at buyer's end

At the buyer's end, a procedure same as that applied for encryption at the seller's end, is applied on the modified design G'_m to produce G''_m. By Lemma 5, G''_m should be the same as G_m. To check if the design has not been tampered during the transmission, the buyer also generates the hash value of G''_m and compares with the hash value received from the sender. If these two values match, the buyer is sure that $G''_m = G_m$ and the received design has not been tampered during transmission.

3.4 Properties of the Encryption and Decryption schemes

The encryption scheme primarily is used to guard the valuable design against an intruder on the net. As discussed earlier, an additional precaution through watermark embedding helps to detect any unauthorized use of the valuable design by any user other than the valid buyer. In our proposed scheme, the watermarking helps to check non-repudiation as well.

The following parameters used by the seller prior to the transmission are considered hidden from any intruder:

- The private key K_b of the seller.
- The length L of the $LFSR$ and the feedback equation for the maximum-length $LFSR$.
- The value of α.

In case the length L of the $LFSR$ is known to the intruder, the following possibilities arise:

- If key K_b is not known to the intruder then number of possible seed values for initializing $LFSR$ is $O(2^L)$.
- If the feedback equation is not known to the intruder, then the number of possible feedback equations for maximum length $LFSR$ is $O(2^L)$.

In all the above cases, it is extremely difficult for the intruder to guess the maximum-length $LFSR$ used by the seller, and hence to obtain the original valuable design being transmitted over the Internet.

Definition 2. The characteristic polynomial associated with a maximum length $LFSR$ is called primitive polynomial. A characteristic polynomial is primitive if

- it is prime
- it is a factor of $X^N + 1$, where $N = 2^L - 1$, and X is a variable of the polynomial.

Number of primitive polynomials for a L stage $LFSR$ is given by $\frac{\Phi(2^L-1)}{L}$, where $\Phi(n) = n \times \Pi(1 - \frac{1}{p})$, p is taken over all prime factors of n [13]. Thus for an L stage $LFSR$, the number of maximum length $LFSR$s is $O(2^L)$, and for an intruder knowing L, it is very difficult to guess the $LFSR$ used in encryption.

3.5 Use of the parameter α

The length of the output sequence of maximum length $LFSR$ is $2^L - 1 \simeq O(2^L)$. If we consider the complete output sequence, then the time complexity of encryption would be very large. The parameter α $(0 < \alpha < 1)$ helps to reduce this time complexity drastically. Let the number of vertices in input systems graph G be N, and we would be modifying a maximum of $\alpha \times N^2$ vertex pairs (for edge insertion or deletion). In such a case, we choose a random sequence of length $2 \times \sqrt{\alpha} \times N$ from the entire output sequence of the maximum length $LFSR$. It can be shown that the worst-case time complexity of encryption, considering the parameter α is $O(N^2)$. The length of precision of α has to be chosen properly to improve the robustness of the encryption as well.

Table 1: Summary of results for watermark creation and verification

Problem	# vertices	LFSR Length	CPU time (sec) watermark	
			creation	verification
ibm01	12752	10	1	0.028
ibm02	19601	10	1	0.044
ibm05	29347	10	1	0.07
ibm07	45926	10	2	0.113
ibm08	51309	10	6	0.126
ibm10	69429	10	4	0.171

LEMMA 6. *The best case and worst case time complexities of encryption and decryption are $O(N)$ and $O(N^2)$ respectively, where N is the number of vertices in G.*

We skip the proof for brevity.

4. EXPERIMENTAL RESULTS

The proposed scheme is implemented using C language on a Sun workstation running Solaris for some $MCNC$ Benchmark circuits [16]. Table 1 shows the CPU times required to create and verify watermarks for $LFSR$ of length 10. Table 2 shows the times of encryption and decryption for different values of α. In our experiment, due to want of space, the graph is implemented using linked list, leading to a difference in encryption and decryption times. The matrix representation of the graph, however, would yield identical encryption and decryption times. Let E_m and E respectively denote the number of edges in the modified graph and the original graph. Let δ denote the Degradation factor. Then Degradation factor is given by the following equation:

$$\delta = \frac{E_m - E}{E} \qquad (3)$$

Also, if $\rho(G)$ denotes the density of the graph G, N and E respectively denote the number of vertices and edges in G, then density is defined using the following equation:

$$\rho(G) = \frac{2 \times E}{N \times (N-1)} \qquad (4)$$

Table 1 shows that the creation and verification of the watermark requires nominal time, which is a desirable feature of such a scheme. In Table 2, we see that density, ρ, of edges decreases from *ibm*01 to *ibm*10. As a result, degradation factor δ increases. δ refers to the % change in the number of edges in the input graph due to encryption. As density of the graph decreases, the probability of number of edges inserted in the graph increases, i.e. degradation factor increases. During encryption, α is used to take a part of output m-sequence from $LFSR$ from which two sets are formed taking the position of 0s and 1s. For larger value of α, the length of output sequence is large, there will be more number of 0s and 1s in the part of the output sequence of the $LFSR$, resulting in more elements in the two sets. Thus cross product of that two sets yields more edges to modify, which in turn increases the encryption and decryption time.

Figures 4 and 5 respectively show the variation of the CPU times with # of vertices in the original input graph for encryption and decryption, and for watermark creation and verification. In the former, the value of α is taken as 0.0001, and in the latter, the length of the $LFSR$ is considered to be 10. It may be noted that watermark creation time depends on the processing of polynomial $g(x)$ by the

Table 2: Summary of results for encryption and decryption for key-length = 25 and LFSR length = 16

Problem	# vertices	# edges		ρ	α	δ	CPU time (sec)	
		before encryption	after encryption				for encryption	for decryption
ibm01	12752	31309	47682	0.000385	0.0001	0.523	1	0
			112769		0.0005	2.6	11	2
			194025		0.001	5.2	29	5
			844487		0.005	26	298	20
ibm05	29347	81768	167886	0.00019	0.0001	1.053	12	2
			512518		0.0005	5.27	119	11
			943610		0.001	10.54	326	23
			4387154		0.005	52.65	3540	108
ibm10	69429	140482	622612	0.000058	0.0001	3.43	141	13
			2548818		0.0005	17.14	1508	60
			4961302		0.001	34.32	4176	117
			24241432		0.005	171.56	49212	584

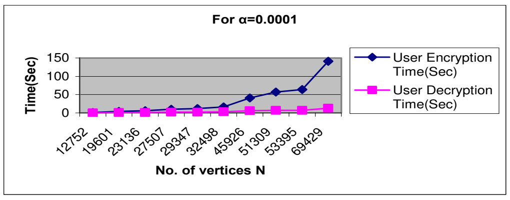

Figure 4: CPU times for encryption and decryption vs. no. of vertices

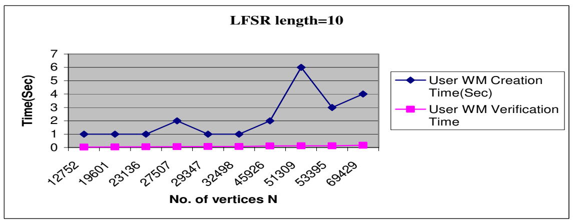

Figure 5: CPU times for watermark creation and verification vs. number of vertices

$LFSR$. However, the watermark verification time depends on the use of mask M only. This explains the difference in watermark creation and verification times. If the maximum degree of the graph is large, then number of terms in $g(x)$ would be large, implying more processing time for $LFSR$. The peaks in the chart for some graphs, such as for $ibm08$ with 51309 vertices, indicate that the maximum degrees of these graphs are very large, resulting more number of terms in $g(x)$.

5. CONCLUSIONS

In this paper, we propose an Internet-based scheme that ensures both direct and indirect IP protection. To the best of our knowledge, this is one of the very few IP protection schemes encompassing both cryptography and watermarking. The novelty of the work is that it is Internet-based, and uses $LFSR$ perhaps for the first time in IP protection. The proposed method has scopes of improvement in terms of (i) embedding the watermark within the input graph, (ii) using public-private key combinations.

6. REFERENCES

[1] J. Lach. A. B. Kahng and H. Mangione-Smith. Constraint-based watermarking techniques for design ip protection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(10):1236–1251, 2001.

[2] H. J. Choi A. E. Caldwell and A. B. Kahng. Effective iterative technique for fingerprinting design ip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(2), 2004.

[3] P. R. Pari G. Qu. A. Jain L. Yuan. Zero overhead watermarking technique for fpga designs. In *Proc. of the Great Lakes Symposium on VLSI (GLSVLSI)*, pages 147–152, April 2003.

[4] I. L. Markov M. Potkonjak P. Tucker H. Wang A. B. Kahng, S. Mantik and G. Wolfe. Robust ip watermarking methodologies for physical design. In *Proc. of the Design Automation Conference*, pages 782–787, June 1998.

[5] E. Charbon. Hierarchical watermarking in ic design. In *Proc. of the Custom Integrated Circuits Conference*, 1998.

[6] E. Charbon and I. H. Torunoglu. On intellectual property protection. In *Proc. of the Custom Integrated Circuits Conference*, pages 517–522, 2000.

[7] S.Sur-Kolay S. Sen-Sarma D. Saha P. Dasgupta. A novel scheme for encoding and watermark embedding in vlsi physical design for ip protection. In *Proc. of the International Computing Conference: Theory and Algorithms*, pages 111–116, 2007.

[8] A. Garcia A. Loris U. Meyer-Baese E. Castillo, L. Parrilla. Ipp watermarking technique for ip core protection on fpl devices. In *Proc. of the Field Programmable Logic and Applications*, pages 1–6, August 2006.

[9] J. P. Hayes. *Computer Architecture and Organization.* Tata Mc-graw Hill, 1996.

[10] W. H. Mangionne-Smith J. Lach and M. Potkonjak. Robust fpga intellectual property protection through multiple small watermarks. In *Proc. of IEEE/ACM Design Automation Conference*, pages 831–836, June 1999.

[11] L. Ghouti L. Yuan, G. Qu and A. Bouridane. Vlsi design ip protection: Solutions, new challenges, and opportunities. In *Proc. of NASA/ESA Conf. on AHS*, 2006.

[12] R. C. Merkle. *Secrecy, authentication, and public key systems.* Stanford Ph.D. Thesis, 1979.

[13] A. D. Friedman Miron Abramovici Melvin A. Breuer. *Digital System testing and Testable Design.* Jaico Publishers, 2001.

[14] M. Ni and Z. Gao. Detector-based watermarking technique for soft ip core protection in high synthesis design level. In *Proc. of the*, pages 1348–1352, 2005.

[15] W. Stallings. *Cryptography and network security.* Prentice-Hall India, 2005.

[16] The ISPD98 Circuit Benchmark Suite. http://vlsicad.ucsd.edu/uclaweb/cheese/ispd98.html.

[17] G. Qu L. Yuan P. R. Pari. Soft ip protection: Watermarking hdl codes. In *Proc. of the 6th Information Hiding Workshop (IHWŠ04)*, pages 224 – 238, May 2004.