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

Raju Halder, Parthasarathi Dasgupta, Saptarshi Naskar, and Samar Sen Sarma

***Abstract*—Due to emerging trend of design reuse in VLSI circuits, the intellectual property (IP) of design faces serious challenges like forgery, theft, misappropriation etc. These increasing risks of design IP stored in design repositories, or the threat of hacking the same during its Internet-based transmission, mandates design file encryption and its appropriate watermarking. In this paper, we propose a novel Internet-based scheme to tackle this problem. Input to the proposed scheme is a generic 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 designs 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 well-known benchmark problem sets are encouraging.**

***Index Terms*—Intellectual property protection (IPP), Watermarking, Encryption, Decryption.**

I. Introduction

THE need of design reuse has been incessantly encouraged by the increasing complexity of VLSI circuits, the pressures to reduce time to market, better design productivity, etc. It offers more integration on a single chip within a shorter design cycle. Circuit components, available in electronic form, signify intellectual property (IP) of VLSI design [1]. Unfortunately, unlimited and careless design reuse may lead to infringement of IP [1], [2], 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 [3] of a legal purchaser by the seller.
- Claiming false IP ownership.
- Illegal reselling by a purchaser without paying proper royalty to the designer.
- Attempt to recover original design 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 [4], [5], [6], [7], [2], [8], [9] have been reported so far, as discussed in Section II.

Manuscript received July 08, 2010; revised December 27, 2010.

Raju Halder is with the Department of Computer Science, Università Ca' Foscari Venezia, Italy, e-mail: halder@unive.it

Parthasarathi Dasgupta is with the Indian Institute of Management Calcutta, India, e-mail: partha@iimcal.ac.in

Saptarshi Naskar and Samar Sen Sarma are with the Department of Computer Science and Engineering, University of Calcutta, India, e-mail: {sapgrin@gmail, sssarma2001@yahoo}.com

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 the watermark of 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.

A. Motivation of the work

IP protection scheme is typically judged by several parameters, such as its strength against any attempt to infringe IP, imperceptibility, maximal capacity, the ease of watermark detection, and so on [10]. None of the existing IP protection techniques can ensure complete IP security in terms of all these parameters. 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 in such a case. Any misuse of the design tool, without being noticed and detected, causes huge loss of royalty to the IP owner. This warrants watermarking of design. As discussed in [4], a design IP may be supplied by a vendor to a buyer in either of 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 too many new opportunities for the creation and delivery of content in digital forms. Transmission of the encrypted (irregular) partial design rather than encrypted high quality complete design is more secure as far security is concerned. We propose an Internet-based IP protection scheme, where the design tool is split into two modules: the first module is executed at the creator's end, generating a watermark and encrypted intermediate design, which is transmitted electronically, whereas the 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 cryptosystem [3] using private or public keys almost eliminates the use of ultra-secure key transmission. A different Internet-based scheme for VLSI floorplans has already been proposed in [4]. 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 [11]. In the proposed scheme, the core concept involves watermarking and encryption of the design graph with the help of LFSR [12]. The seller uses two LFSRs: one for generating the watermark which is known to the seller only, and the other for encryption-cum-decryption which is known to both the seller and the buyer. During watermark generation, a mask bit string is also generated. This mask bit string is provided to the buyer only on demand for watermark verification. Encryption of the graph is performed by changing the interconnection pattern in the graph randomly. When a design IP is being sold to a buyer, the sold instance of the IP always consists of two components: the "*encrypted design graph*", and the associated "*watermark*". If the encrypted graph is tampered during its transmission, the hash value can be used to detect it at the receiver's end.

The rest of the paper[1] is organized as follows: Section II discusses related works in the literature. Section III recalls some preliminary concepts. Section IV discusses the proposed schemes for watermark generation and verification, and Section V illustrates the methods of encryption and decryption of the design graph. Section VI discusses the empirical observations, and Section VII concludes the paper.

II. Literature Survey

The core idea of constraint-based IP Protection is to shrink the original solution space by embedding signatures as additional constraints. A watermarked solution meets both the original and the additional constraints, and this fact is used to show the authorship. Smaller solution space makes the watermark stronger. Kahng el al. [2] first proposed constraint-based watermarking technique that consists of the following parts: (i) An optimization problem, which is an NP-hard problem that needs constraints and heuristics to be solved; (ii) An off-the-shelf optimization software/algorithm to solve such a problem; (iii) A set of constraints that should be applied to the design; and (iv) A well-formed grammar to add extra constraints to the previous ones for building the required watermarked design. The last part is the main watermarking tool where the watermark is converted into a set of extra constraints and is applied to the design. Due to the generic nature of this approach, it can be applied at different levels of the design flow.

The public constraint-based watermarking technique in [7] embeds public as well as private watermark where the former is visible to public, while the latter is visible to authorized people only. Both the public and private watermarks are in the form of additional design constraints. Author uses cryptographic techniques for data integrity to deter any attempt of removing or modifying the public watermark.

[1]The paper is a revised and extended version of [13]

Saha et al. [4] proposed a watermarking scheme with a completely different viewpoint to protect the physical design at its early stage. They explain the proposed scheme through the floorplanning phase. The input to floorplanning is a graph representing connectivity of the logic modules. The starting step of this scheme is to transform the graph into a Planar Triangulated Graph (PTG) from which two graphs, viz., Horizontal Path Directed Graph (HPDG) and Vertical Path Directed Graph (VPDG) are formed. The scheme consists of two steps: encryption of the design and watermark embedding. The encryption is done by inserting some new dummy nodes along different directed paths in HPDG and VPDG under the control of a secret key K. In watermark embedding phase, a unique integer computed from date-stamp and K is used to identify some modules in the design. Additional Flip-Flops are inserted into each of those selected modules as watermark.

In [5], authors suggest an encoding scheme for tree-based floorplan representation to ensure security during design storage or transmission. The encoding involves replacement of subtree with another subtree of same size using ranking-unranking technique, swapping of subtrees, tree rotation etc. under the control of a symmetric key K. The encoding has $O(n)$ timing requirement and zero space overhead for a floorplan with n modules.

In [14], authors introduced a generic IPP technique, called localized watermarking where the constraints of each watermark are placed in a smaller part (locality) of the design and can be detected in its locality independently. Therefore, such watermarks are able to protect parts of the design as the detection algorithm does not need to see the entire design.

Charbon and Torunoglo [15], [16] introduced a hierarchical watermarking technique which independently process multiple abstraction levels present in a design flow. This approach increases the robustness, since the deletion of a watermark at a certain abstraction level, leaves watermarks at most other abstraction levels intact. Also forgery can be traced to the source, since watermarks at the lowest abstraction levels are associated with the last "legal" IP buyers who ultimately caused the breach.

The watermarking and fingerprinting techniques for FPGA, proposed by Lach et al [17], is performed by inserting the signature in the unused look-up tables (LUTs) in the configurable logic blocks (CLBs) that do not increase the area of the system. However it suffers from the removal attack by reverse engineering a design to a stage before the signature has been applied. In [8], authors presented an IP protection technique for reusable modules used in field programmable logic (FPL) implementations that has the following stages: signature preparation using cryptographic hash function, signature spreading through used or unused positions of look-up tables (LUTs), signature extraction by using an LFSR-based additional logic, and finally, signature validation.

The watermarking of sequential circuits has been introduced in [18], [19]. The essence of these techniques is to use unused input/output sequence or adding new input/output sequences (in case of completely specified FSM) at the finite state machine (FSM) representation of the design.

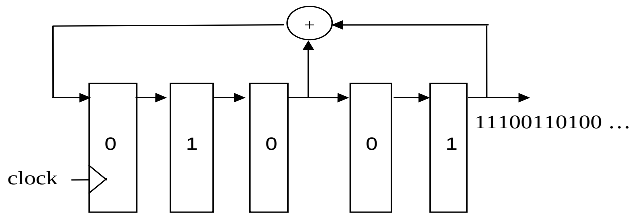

Fig. 1. An example of an LFSR with a feedback loop

III. Preliminaries

A. Introducing Linear Feedback Shift Register (LFSR)

Definition 1: The linear feedback shift register (LFSR) is a series of connected Flip-Flops, with XOR feedback. They have no input except for clocks. The basic components of linear feedback shift register are D flip-flops, modulo-2 adders and modulo-2 scalar multipliers [12].

The value with which an LFSR is initialized, is called seed value. The stream of values produced from the output of an LFSR is completely determined by its current (or previous) state. An example of a 5-stage LFSR is shown in Figure 1. However, an LFSR with a well-chosen feedback connection can produce a sequence of bits which appears random and which has a very long cycle.

Definition 2: 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 maximum length sequence or m-sequence [12].

Two types of Linear Feedback Shift Registers, called *type1/external/Fibonacci* and *type2/internal/Galois* are shown in Figures 2 and 3 respectively. The Fibonacci implementation consists of a simple shift register in which a binary-weighted modulo-2 sum of the taps[2] is fed back to the input. The Galois implementation consists of a shift register whose contents are modified at every step by a binary-weighted value of the output stage. For any given tap, the weight c_i = 0 means there is no connection, while c_i = 1 means the weight is fed back. There are, however, two exceptions: c_0 = c_m = 1, both are always connected. Note that the order of the Galois weights is opposite to that of the Fibonacci weights. An LFSR is characterized by two types of polynomials: one is characteristic polynomial $P(x)$ and the other one is reciprocal characteristic polynomial $P^*(x)$. The characteristic polynomial and reciprocal characteristic polynomial corresponding to the external and internal LFSR are:

$$P(x) = 1 + c_{m-1}x + c_{m-2}x^2 + \cdots + c_1x^{m-1} + x^m$$

$$P^*(x) = 1 + c_1x + c_2x^2 + \cdots + c_{m-1}x^{m-1} + x^m$$

A given set of feedback connections can be expressed in a convenient and easy-to-use shorthand form, called *feedback equation*, with the connection numbers being listed within a pair of brackets. In doing so, connection c_0 is implied, and not listed, since it is always connected. Although c_m is also always connected, it is listed in order to convey the shift register size (number of flip-flops). A set of feedback

[2]The bit positions that affect the next state are called the taps.

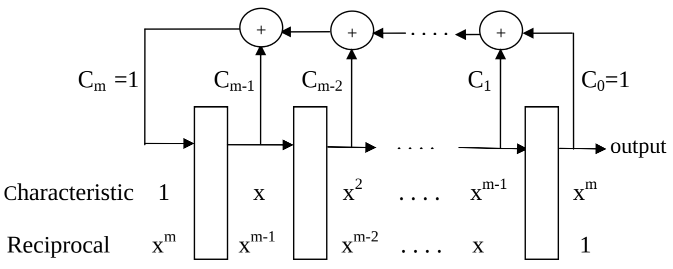
Cm =1
Cm-1
Cm-2
C1
C0=1
output
Characteristic 1 x x2 .... xm-1 xm
Reciprocal xm xm-1 xm-2 .... x 1

Fig. 2. Fibonacci implementation of LFSR

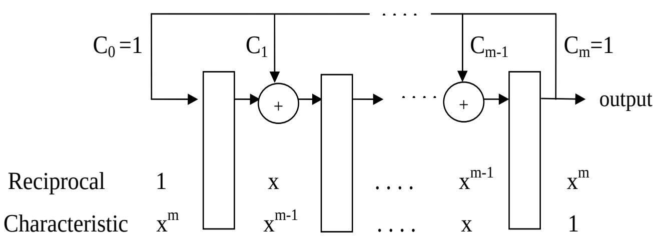

Fig. 3. Galois implementation of LFSR

taps for a Galois generator is denoted as $[f_1, f_2, f_3, ..., f_j]_g$, where subscript j is the total number of feedback taps (not including c_0), $f_1 = m$ is the highest-order feedback tap (and the size of the LFSR), and f_j are the remaining feedback taps. The subscript g signifies the Galois LFSR form. The set of feedback taps for the equivalent Fibonacci generator is denoted as $[f_1, m - f_2, m - f_3, ..., m - f_j]_f$, where the subscript f signifies the Fibonacci LFSR form.

Some properties of LFSRs 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 [12].

$Property$ 2 : For an m-sequence obtained from an L-stage maximum length LFSR, there is one run 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 [12].

B. Merkle-Damgård's Meta method for Hashing

Merkle-Damgård's generic method [20] to build cryptographic hash functions is depicted in Figure 4. In this method, the message x is divided into L blocks $x_1, x_2, x_3, \ldots, x_L$, each of length r, say. If the length of the last block is less than r, it appends 0s to make it of length r. In order to prove that the construction is secure, Merkle and Damgård proposed that messages be padded with a set of bits that encodes the length of the original message. This is called length padding or Merkle-Damgard strengthening. The algorithm starts with an initialization vector (IV) of length r whose value is algorithm or implementation specific. The function F is a one-way compression function that transforms two fixed length inputs to an output of same size. The input to the function F is the current block x_i and previous intermediate value h_{i-1} to produce current value h_i *i.e.* $h_i = F(x_i, h_{i-1})$

for $i = 2, \ldots, L$, where $h_1 = F(x_1, \mathrm{IV})$. After applying F to all L blocks we obtain the final hash value h_L of length r.

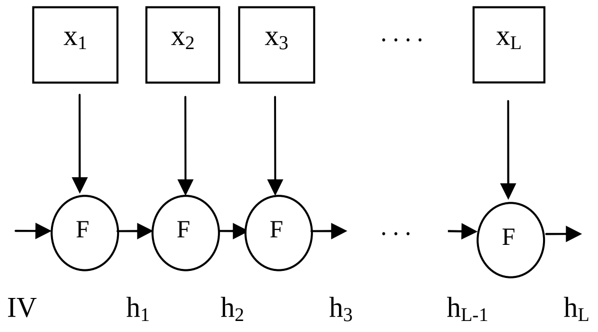

Fig. 4. Hashing using Merkle-Damgård's Meta method

The notations we use in the rest of the paper are listed in Table I.

TABLE I
TABLE OF NOTATIONS USED IN THE STUDY

Notation	Description
G	Input graph corresponding to a digital system design
L	Length of $LFSR$ $i.e.$ number of flip-flops in $LFSR$
$g(x)$	Polynomial generated from design graph G
g_c	Binary co-efficient of $g(x)$
$q(x)$	Quotient obtained from $LFSR$ after applying $g(x)$ to the input terminal of $LFSR$
q_c	Binary co-efficient of $q(x)$
$r(x)$	Remainder obtained from $LFSR$ after applying $g(x)$ to the input terminal of $LFSR$
r_c	Binary co-efficient of $r(x)$
S	Combined signature of both the seller and the buyer
W	Watermark used to authenticate the seller and the buyer
M	Mask bit string which is used in verification phase
arb_i	i^{th} arbitrary bit string of appropriately chosen length used to generate W and M
k_b	Symmetric private key
λ	Length of k_b
α	Fraction used to obtain a part of output sequence from the $LFSR$
V	Vertex set of G
N	Number of nodes in G $i.e.$ $\|V\|$
E	Set of edges of G
G'	Encrypted graph
G''	Decrypted graph
δ	Percentage of change in the number of edges due to encryption, called degradation factor
$\rho(G)$	Percentage of edges actually present in the graph G, called density of the graph

Our proposed scheme consists of two phases:

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

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 risks. During watermarking phase of the proposed scheme, a watermark of the design IP is generated based on the combined signature of both the seller and the buyer. 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 also trace the buyer in case of illegal reselling. Moreover, in order to check for any doubt of illegal design usage, hidden combination of seller's and buyer's signature of a design IP may be retained. The flowchart of the overall scheme is depicted in Figure 5.

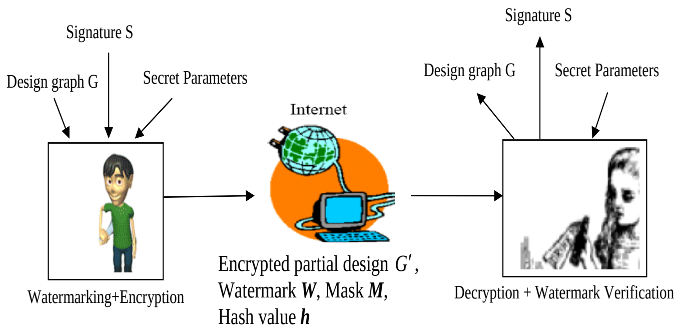

Fig. 5. Flowchart of the overall scheme

IV. PROPOSED WATERMARKING SCHEME

In this Section we describe the method to generate and verify unique and robust watermark of a design graph by considering signatures of both the buyer and the seller. The watermark generation is performed by exploiting the feature of linear feedback shift register-based polynomial division technique, whereas the verification phase is performed simply by using the technique of masking.

A. Watermark generation

For each of the source-destination pairs associated with a particular design graph G, the watermark W will be generated in such a way so that it can identify the pair uniquely. To ensure uniqueness as well as robustness of the generated watermark, an LFSR and the technique of shift register polynomial division using it is used. Figure 6 shows how an LFSR initialized by 0s can perform shift register polynomial division. The flowchart of the watermark generation phase is depicted in Figure 7. In this phase, the seller is free to choose an LFSR of length, say L, and any one of its primitive feedback connections. This LFSR information is treated as secret parameters and is known only to the seller. Given a design graph G, a polynomial $g(x)$ is generated from G as described in Section IV-C. This polynomial characterizes the graph G. However, given a polynomial $g(x)$, the construction of a unique G is not possible. Thus, generation of $g(x)$ from G is a one-way function. The polynomial $g(x)$ is applied at input of the LFSR initialized by 0s. Let $q(x)$ and $r(x)$ be the quotient and remainder obtained from the LFSR respectively. Suppose g_c, q_c, and r_c represent the binary coefficients of polynomials $g(x)$, $q(x)$ and $r(x)$ respectively. To generate watermark, the seller embeds the 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 and compute $g_{cs} = g_c \otimes S_c$. S

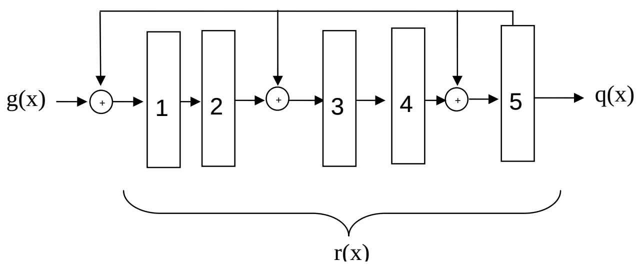

Fig. 6. An LFSR performing shift register polynomial division

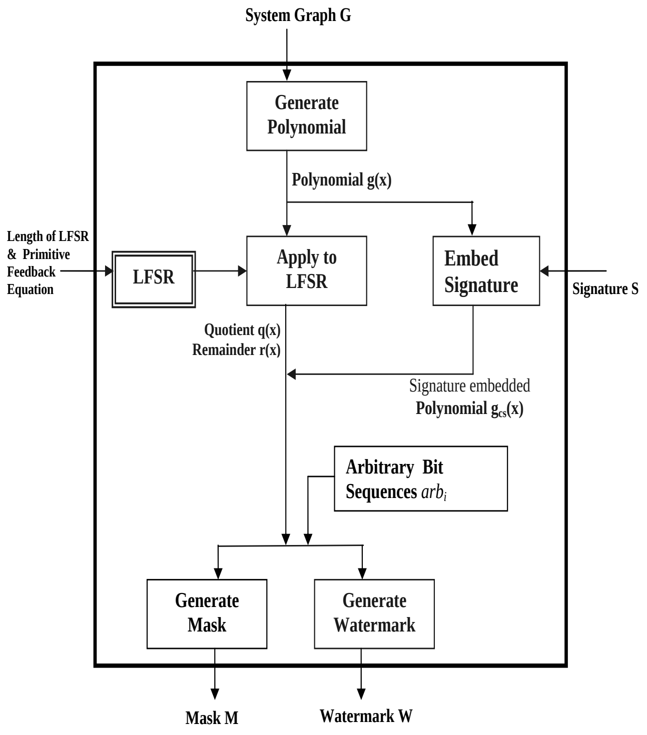

Fig. 7. Flowchart of watermark generation

may be signature of the seller or the buyer, or combination of the two. 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.

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

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 about g_{cs}, q_c, and r_c (so as to guess about LFSR and G). Bit '1' is used as $start\_bit$ and end_bit to indicate the begin and end of g_{cs}.

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

where $\{0\}^*$ represents a string of 0s used in generating mask M, and is of length (*start_bit* + g_{cs} + *end_bit*). The formal description of the proposed algorithm for watermark generation appears in Figure 8.

It may be noted that the use of secret LFSR ensures uniqueness as well as robustness of the watermark. Lemma 1 summarizes the utility of LFSR.

Lemma 1: Let G, L and S represent the set of design graphs, maximum length LFSRs and signatures respectively. Let W^l_{gs} denotes the watermark of $g \in G$ associated with signature $s \in S$ and is obtained by using LFSR $l \in L$. Then, $\forall G_1, G_2 \in G$, $\forall L_1, L_2 \in L$ and $\forall s \in S$:

$$W^{L_1}_{G_1 s} \neq W^{L_2}_{G_2 s} \quad if \ L_1 \neq L_2$$

where G_1 and G_2 are completely different designs or two versions of the same design.

Proof: Let $W^{L_1}_{G_1 s}$ and $W^{L_2}_{G_2 s}$ be two watermarks generated for the input design graphs G_1 and G_2 with signature s using LFSRs L_1 and L_2 respectively.

Let $p_1^*(x)$ and $p_2^*(x)$ be the reciprocal characteristics polynomials of L_1 and L_2 respectively. Suppose, $g_1(x)$ and $g_2(x)$ are two polynomials generated from G_1 and G_2 respectively. After applying $g_1(x)$ to L_1 and $g_2(x)$ to L_2 we get the quotients and remainders $q_1(x)$, $r_1(x)$ and $q_2(x)$, $r_2(x)$ respectively. Thus we get,

$$g_1(x) = p_1^*(x) q_1(x) + r_1(x) \qquad (3)$$

and

$$g_2(x) = p_2^*(x) q_2(x) + r_2(x) \qquad (4)$$

Consider the following cases where for two graphs G_1 and G_2 there is a chance that the generated watermarks for them can be same:

1) $G_1 \neq G_2$ but generates same polynomial.
2) G_1 and G_2 represent two versions of the same design for a given seller.

We already know that the watermark characterizes the design graph as well as the source of that design. To make unique watermark for each of the cases above, we use different LFSRs. Thus, we have $p_1^*(x) \neq p_2^*(x)$ and $g_1(x) = g_2(x)$. Hence, from equations 3 and 4 we get, $q_1(x) = q_2(x)$ implies $r_1(x) \neq r_2(x)$. Similarly, $r_1(x) = r_2(x)$ implies $q_1(x) \neq q_2(x)$. In other words, both $q_1(x) = q_2(x)$ and $r_1(x) = r_2(x)$ can not be satisfied simultaneously. Since the watermark is formed by concatenating the binary coefficients g_{cs}, q_c and r_c, different q_c or different r_c makes $W^{L_1}_{G_1 s} \neq W^{L_2}_{G_2 s}$. ■

Algorithm 1: GenWatermark

Input: System Graph G, An $LFSR$ 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 by 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 $g_{cs}$ = $S_c \otimes g_c$.
5. Create $W$ by concatenating $g_{cs}$, $q_c$, $r_c$ and $arb_i$, $i$=0,...,3.
6. Create corresponding mask $M$.
```

Fig. 8. Algorithm for watermark generation

B. Watermark verification

The flowchart of watermark verification phase is depicted in Figure 9. While verifying the watermark of the design, the buyer needs to use the mask M transmitted in the encrypted file. $W \otimes M$ 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 can now generate the polynomial $g(x)$ from the design graph (s)he bought. Finally, the signature S can be produced by performing $g_{cs} \otimes g_c$ which is used to verify the authenticity of the design. A formal description of the algorithm for watermark verification is shown in Figure 10.

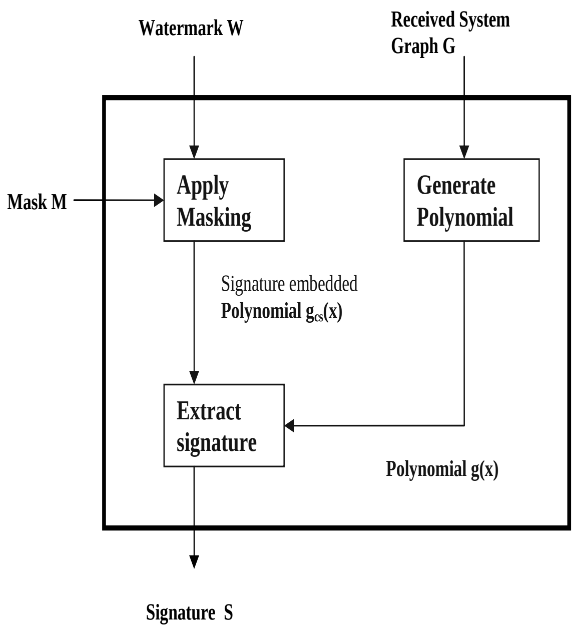

Fig. 9. Flowchart of watermark verification

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

Fig. 10. Algorithm for watermark verification

C. Generating a Polynomial from a given graph

Let G be an input design graph. Count the number of nodes of G having identical degrees in decreasing order of their degrees. The *degree* is used to denote the power of each term and *(count+degree)* contributes to the coefficient of the corresponding term. Since the coefficients of the polynomial are binary, apply mod-2 operation on *(count+degree)* to obtain the binary coefficients. In the worst case, even values for all *(count+degree)* may yield an empty polynomial. To ensure the generation of non-empty polynomial, we make the polynomial *monic* by enforcing the coefficient of the highest degree term to 1.

D. Version Control

The proposed watermark generation scheme assigns a unique watermark to 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 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 reflected in W. Moreover, in order to control different versions of same design graph, the seller can use different private LFSRs to generate different unique watermarks and corresponding masks. 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.

V. Proposed Encryption and Decryption schemes

In this Section we discuss the encryption and decryption of the design graph G with the help of an L-stage LFSR. Hereafter, by LFSR, we refer to only maximum length LFSR.

A. The encryption scheme

The flowchart of design graph encryption is depicted in Figure 11. We consider (i) An 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: generate a sub-key k_1 from the private key k_b. Let the length of K_1 be λ. We generate the seed (a bit string) of the L-stage LFSR by first converting K_1 into a compressed key K_1' and then choosing the first L-bits obtained from K_1' as the seed. We explain the method with an example. Let K_1 = ABCDEFGHIJKLMN. Then, K_1 is divided into a number of parts, each part having length = $\lceil \frac{L}{8} \rceil$ = 3, say. Thus, the parts obtained from K_1 are (ABC), (DEF), (GHI), (JKL), (MN). Now K_1' = (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, 2 for B, and so on) of the i^{th} characters in these components. Thus, the first alphabet in K_1' 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, the first alphabet of K_1' = 35 mod 26 + 1 = 10 = 'J'. Similarly, the second and the third alphabets of K_1' are 'O' and 'E' respectively. Thus, starting with K_1 = (a_1, a_2, …, a_λ) where $\lambda \geq \lceil \frac{L}{8} \rceil$, we obtain K_1' = (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_1' 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, otherwise only the first L bits of it is considered as the seed. Next, the LFSR is initialized with the seed value.

The maximum-length LFSR outputs an m-sequence of length $2^L - 1$ which repeats itself. We just pick a binary string of length $2 \times \sqrt{\alpha} \times N$ from the output of the LFSR, where $0 < \alpha < 1$ and $N = \mid V \mid$ is the number of nodes in the design graph G. Lemma 2 depicts the criteria for this sequence of length $2 \times \sqrt{\alpha} \times N$ to be random, in contrast to pseudorandom.

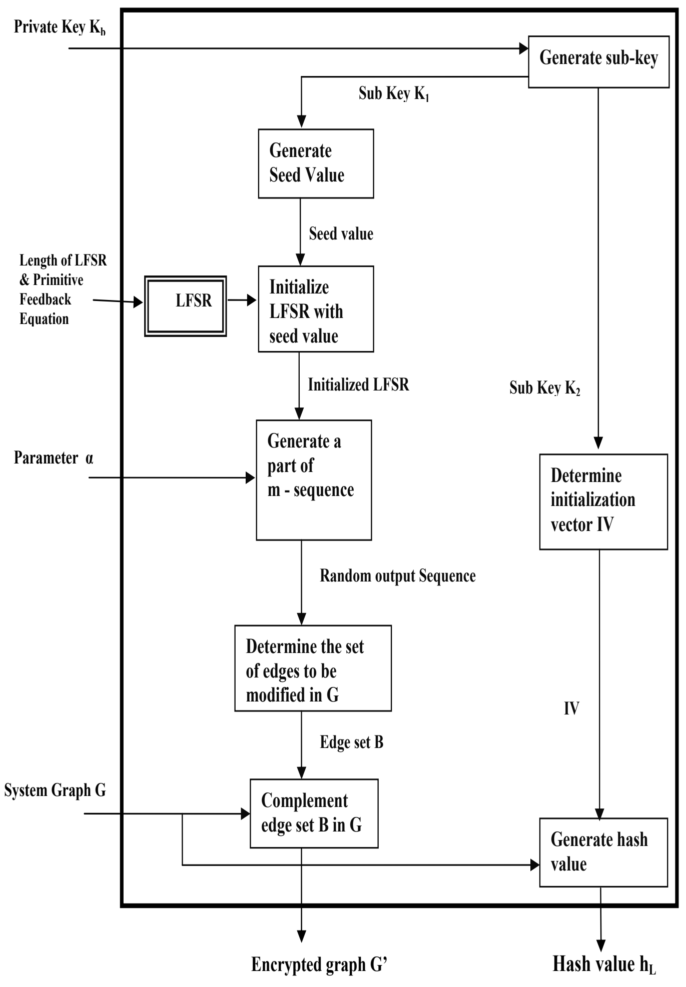
Private Key K_h
Generate sub-key
Sub Key K_1
Generate Seed Value
Seed value
Length of LFSR & Primitive Feedback Equation
LFSR
Initialize LFSR with seed value
Initialized LFSR
Sub Key K_2
Parameter α
Generate a part of m - sequence
Determine initialization vector IV
Random output Sequence
Determine the set of edges to be modified in G
IV
Edge set B
System Graph G
Complement edge set B in G
Generate hash value
Encrypted graph G'
Hash value h_L

Fig. 11. Flowchart of design graph encryption

Lemma 2: The sequence of length $2 \times \sqrt{\alpha} \times N$, where $0 < \alpha < 1$ and $N = \mid V \mid$ which is taken from the output of a maximum-length LFSR of length L must be random if $L \geq log_2((2 \times \sqrt{\alpha} \times N) + 1)$.

Proof: For an LFSR of length L, there exist 2^L possible states. Since for any maximum length LFSR all states except 0 must occur, the number of possible states is $2^L - 1$. Therefore, after the length $2^L - 1$, the output sequence repeats. Thus, if we consider an output sequence of length more than $2^L - 1$, it is pseudorandom. In contrast, since any state can not occur more than once within a length of $2^L - 1$, any part of output sequence of length equal or less than $2^L - 1$ must be random. Thus we have,

$$\begin{aligned} & 2 \times \sqrt{\alpha} \times N \;\leq\; 2^L - 1 \\ or, \quad & (2 \times \sqrt{\alpha} \times N) + 1 \;\leq\; 2^L \\ or, \quad & L \;\geq\; log_2((2 \times \sqrt{\alpha} \times N) + 1) \end{aligned}$$

■

In the next step of encryption, the positions of all 0s and 1s in the random 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: let S_1 and S_2 be two sets of integers formed by collecting the positions of all 0s and the positions of all 1s in the sequence respectively. In these two sets, if any integer element p is greater than or equal to N (= $\mid V \mid$), the value of '$p \bmod N$' is computed. If the value '$p \bmod N$' does not exist in the sets, p is replaced with '$p \bmod N$', otherwise p is simply deleted from the set. Thus, the integers in S_1 and S_2 represent the vertex numbers of the design graph G.

Lemma 3: The length $2 \times \sqrt{\alpha} \times N$, where $0 < \alpha < 1$ and $N = \mid V \mid$ of the sequence 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.

Proof: According to Property 1 (Section III), an m-sequence has 2^{L-1} 1s and $(2^{L-1} - 1)$ 0s. Also Property 2 says that there is one run of L consecutive 1s and one run of $L-1$ consecutive 0s in the m-sequence. 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. Since one run of L consecutive 1s and $L-1$ consecutive 0s always occur in an m-sequence, and the length of other runs of 0s or 1s are less, the length $2 \times \sqrt{\alpha} \times N$ of the sequence obtained from the output of LFSR must be greater than L to include at least one 1 and one 0. ■

Observe that Lemma 2 gives the lower bound for L, whereas lemma 3 gives the upper bound for L. Thus, from Lemma 2 and Lemma 3, we get

$$log_2(2 \times \sqrt{\alpha} \times N) \leq L < 2 \times \sqrt{\alpha} \times N$$

In the next step, the Cartesian product Π of the two sets S_1 and S_2 is formed. Any pair in it corresponds to an edge to be used to modify the design graph G. If any pair in Π corresponds to an edge in G, that edge is deleted from G. Otherwise the edge is inserted in G. This forms a modified graph G'. Lemma 4 shows that applying the encryption algorithm twice on an input graph will yield the original graph only.

Lemma 4: Assuming encryption of a graph G to be a function $F(G)$, $F(F(G))$ = G.

Proof: Suppose $G(V, E)$ represents an input graph and B be the set of edges to be modified in G, where B is obtained after performing cartesian product of S_1 and S_2. Let F be the encryption function that removes an edge $e \in B$ from G if $e \in E$, and inserts $e \in B$ into G if $e \notin E$. Suppose, application of the function F on G using B yields a graph G' *i.e.* $F(G) = G'$, and performing F again on G' using the same B yields a graph G'' *i.e.* $F(F(G)) = F(G') = G''$. We have to prove that $G = G''$ i.e. $V = V''$ and $E = E''$.

Since the encryption function F does not make any changes in the number of nodes when applied on a graph, we have $V = V''$. Now we prove, $E \subset E''$ first and then $E'' \subset E$. For the first, $\forall e \in E$ consider the following:

1) $e \notin B$: Since F complements from G only those edges that belong to B, the resultant graphs G' and G'' contain e.
2) $e \in B$: In this case, F removes e from G and thus, disappear in G'. Further application of F on G' using the same B will make the edge e again available in G''.

Thus, $\forall e \in E : e \in E''$ *i.e.* $E \subset E''$.
Now we prove $E'' \subset E$. To do that, $\forall e \in E''$ we again consider the following:

1) $e \notin B$: The function F does not change any edges $e \notin B$ from G or G' to generate G''. Thus, the fact $e \notin B \wedge e \in E''$ clearly says that $e \in E' \wedge e \in E$.

2) $e \in B$: In this case, the fact $e \in B \wedge e \in E''$ says that $e \notin E'$. Similarly, $e \in B \wedge e \notin E'$, in turn, says that $e \in E$.

Thus, $\forall e \in E'' : e \in E$ *i.e.* $E'' \subset E$. Hence, from $E \subset E''$ and $E'' \subset E$ we get $E = E''$ which proves our lemma. ■

In the final step of encryption, at the sender's end, a hash value of G is generated, which is transmitted along with the encrypted graph G'. This hash value is used in verification phase performed at the buyer's end to detect any alteration of G occurred during its Internet-based transmission. For generating the hash value, G 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). Merkle-Damgård's Meta method [20] is applied on this binary string to generate the hash value of G. Observe that the value of the initialization vector IV is computed from another subkey k_2 obtained from k_b.

B. Recovery of the original graph at buyer's end

At the buyer's end, the same procedure as that applied for encryption at the seller's end is applied on the modified design G' to produce G'' (= G). By Lemma 4, G'' should be the same as G. To check if the design has not been tampered during the transmission, the buyer also generates the hash value of G'' and compares with the hash value received from the sender. If these two values match, the buyer is sure that G'' = G and the received design has not been tampered during transmission.

C. Properties of the Encryption and Decryption schemes

The encryption scheme primarily is used to guard the valuable design against an intruder while transmitting through the Internet or when stored in the design repositories. 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.

We now attempt to justify the use of LFSR in the encryption of G. LFSR is used to implement random number generators. Thus, for a specific G, its modification with the use of random sequence from an LFSR yields a random graph. The LFSR can be simulated in software or can be implemented in hardware. However, if the feedback is non-linear for a given number of stages in LFSR, the resulting key sequence will be of higher linear complexity [21]. Therefore, to obtain a better security, we can combine multiple LFSRs, or we can use binary random sequences based on elliptic curve points [22] to introduce the nonlinearity in the sequence.

The following parameters used by the seller prior to the transmission are considered hidden from any intruder: (i) the private key [3] k_b of the seller, (ii) the length L of the LFSR and the feedback equation for the maximum-length LFSR, and (iii) 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, the number of possible seed values for initializing LFSR is $O(2^L)$.
- If the feedback equation is not known to the intruder, 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 during encryption.

Definition 3: The characteristic polynomial associated with a maximum length LFSR is called primitive polynomial. A characteristic polynomial is primitive if (i) it is prime *i.e.* it can not be factored, and (ii) it is a factor of $x^N + 1$, where $N = 2^L - 1$, x is a variable of the polynomial, and L is the length of the LFSR.

Number of primitive polynomials for an L stage LFSR is given by $\frac{\Phi(2^L-1)}{L}$, where $\Phi(n) = n \times \Pi_{p|n}(1 - \frac{1}{p})$, p is taken over all primes that divide n [12]. Thus for an L stage LFSR, the number of maximum length LFSRs is $O(2^L)$, and for an intruder knowing L, it is very difficult to guess the LFSR used in encryption.

D. 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 while encrypting the graph G, the time complexity of encryption would be exponentially large. The parameter α ($0 < \alpha < 1$) helps to reduce this time complexity drastically by considering a part of the output sequence rather than complete sequence. Moreover, it maintains the randomness of the sequence. The value of α determines how many edges will be modified in the graph during encryption, and thus, is chosen by the owner accordingly. Let the number of nodes in the input design graph G be N, and the owner would like to modify a maximum of $\alpha \times N^2$ edges. In such a case, owner chooses 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 $O(N^2)$ (see Lemma 5). The length of precision of α has to be chosen properly to improve the robustness of the encryption as well.

Lemma 5: 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 nodes in G.

Proof: In best case, any one of the two sets S_1 and S_2 (which are formed by taking into account the positions of 0s and 1s from the output sequence of length $2 \times \sqrt{\alpha} \times N$) contains only one element, whereas another set contains ($2 \times \sqrt{\alpha} \times N$) -1 elements. Hence the Cartesian product of these two sets yields $O(N)$ edges to modify. In worst case, the two sets contain equal numbers of elements *i.e.* $\sqrt{\alpha} \times N$ which yield $O(N^2)$ edges after Cartesian product. Thus, the best and worst case time complexities of encryption and decryption are $O(N)$ and $O(N^2)$ respectively. ■

VI. Experimental results

The proposed scheme is implemented using C language and is executed over some ISPD98 benchmark suite [23] in a Sun workstation running Solaris. The ISPD98 benchmark suite includes a set of circuits for physical design applications, such as partitioning and placement. These circuits were translated from internal IBM designs that represent many types of parts, including bus arbitrators, bus bridge chips,

TABLE II
SUMMARY OF RESULTS FOR WATERMARK GENERATION AND VERIFICATION

Problem	Number of nodes	LFSR Length	CPU time(sec) for watermark	
			generation	verification
ibm01	12752	10	1	0.028
		20	1	0.028
ibm02	19601	10	1	0.044
		20	1	0.044
ibm03	23136	10	1	0.054
		20	1	0.054
ibm04	27507	10	2	0.065
		20	2	0.065
ibm05	29347	10	1	0.07
		20	1	0.07
ibm06	32498	10	1	0.078
		20	1	0.078
ibm07	45926	10	2	0.113
		20	2	0.113
ibm08	51309	10	6	0.126
		20	6	0.126
ibm09	53395	10	3	0.131
		20	3	0.131
ibm10	69429	10	4	0.171
		20	4	0.171

memory and PCI bus interfaces, communication adaptors, memory controllers, processors, graphics adaptors etc. While performing the simulations, we generate the watermark and the corresponding mask, and encrypt the netlist graphs (partial physical design) obtained from the benchmark problems. Finally, we decrypt the graphs into their original form and verify the watermark using the mask bit string.

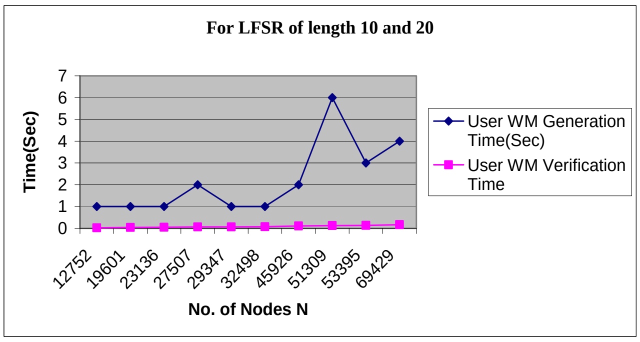

Fig. 12. CPU times for watermark generation and verification vs. number of nodes for LFSR of length 10 and 20

Table II depicts the CPU time required to generate and verify watermarks for two different LFSRs of length 10 and 20. The variation of CPU times for watermark generation and verification with the number of nodes in the graph are depicted in Figure 12 for LFSRs of length 10 and 20. Observe that generation and verification of the watermark requires nominal time, which is a desirable feature of such a scheme. The watermark generation time mainly depends on two factors: (i) generation of polynomial from the input graph, and (ii) length of the generated polynomial and its processing by the LFSR, whereas verification time depends on the first factor only. This explains the difference between watermark generation and verification times. The increase in the number of nodes from $ibm01$ to $ibm10$ leads to an increment in the polynomial generation time which in turn increases the watermark generation and verification time. The maximum degree in a graph determines the length of the polynomial generated from that graph. If the maximum degree is large, the number of terms in $g(x)$ would be large, implying more processing time by the LFSR. As the watermark generation time also depends on the length of the generated polynomial and its processing by LFSR, the watermark generation time for $ibm04$ and $ibm08$ become little bit larger due to the larger value of the maximum degree nodes in them. This is indicated by peaks in the chart. It is worthwhile to mention that no significant changes in the processing time of the polynomial by LFSR is observed while using two different LFSRs of length 10 and 20, resulting into same CPU times for watermark generation and verification.

Before discussing the experimental results on encryption and decryption, we first define two parameters: $density$ of a graph and the $degradation\ factor$.

Let $|V|$ and $|E|$ denote the number of nodes and edges in a graph G respectively, the $density$ $\rho(G)$ of G is given by the following equation:

$$\rho(G) = \frac{2 \times |E|}{|V| \times (|V| - 1)} \tag{5}$$

Let $|E'|$ and $|E|$ denote the number of edges in the encrypted graph and the original graph respectively, the $degradation\ factor$ δ is defined by the following equation:

$$\delta = \frac{|E'| - |E|}{|E|} \tag{6}$$

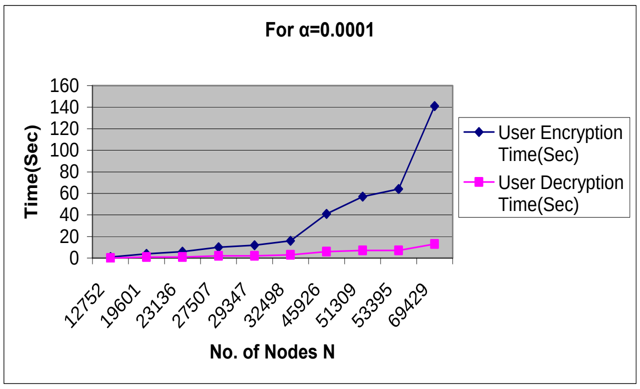

Fig. 13. CPU times for encryption and decryption vs. no. of nodes for α=0.0001

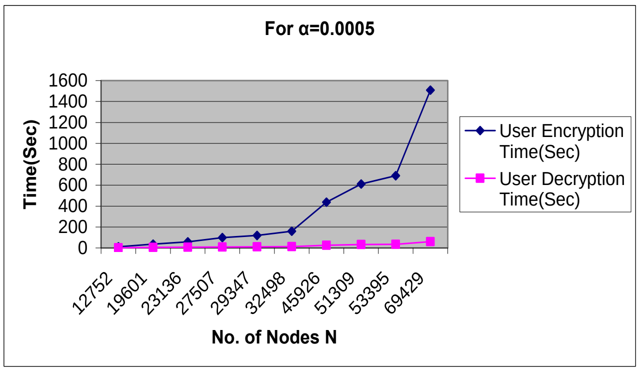

Fig. 14. CPU times for encryption and decryption vs. no. of nodes for α=0.0005

The encryption and decryption time for different values of α ($i.e.$ 0.0001, 0.0005, 0.001 and 0.005) is depicted in Table III. In our experiment, due to want of space, the graph is implemented using linked list, leading to a difference in

TABLE III
SUMMARY OF RESULTS FOR ENCRYPTION AND DECRYPTION FOR KEY-LENGTH = 25 AND LFSR LENGTH = 16

Problem	Number of nodes	Number of 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
ibm02	19601	54426	93032	0.000283	0.0001	0.709	4	1
			246556		0.0005	3.53	36	5
			438606		0.001	7.06	101	10
			1975516		0.005	35.3	1069	49
ibm03	23136	60355	113929	0.000226	0.0001	0.888	6	1
			328101		0.0005	4.44	58	7
			595795		0.001	8.87	164	13
			2734843		0.005	44.31	1753	66
ibm04	27507	67117	143003	0.000177	0.0001	1.13	10	2
			445829		0.0005	5.64	99	10
			823650		0.001	11.27	271	19
			3849763		0.005	56.36	2925	93
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
ibm06	32498	87605	193208	0.000166	0.0001	1.205	16	3
			615831		0.0005	6.03	160	13
			1144185		0.001	12.06	438	26
			5366355		0.005	60.26	4776	131
ibm07	45926	114158	325274	0.000108	0.0001	1.849	41	6
			1168712		0.0005	9.24	437	26
			2222200		0.001	18.47	1237	53
			10657408		0.005	92.357	13369	260
ibm08	51309	133610	397272	0.000102	0.0001	1.973	57	7
			1449968		0.0005	9.85	611	33
			2765871		0.001	19.70	1716	65
			13294838		0.005	98.51	18708	325
ibm09	53395	114088	399224	0.00008	0.0001	2.499	64	7
			1539085		0.0005	12.49	690	35
			2963586		0.001	24.98	1923	69
			14368822		0.005	124.95	21024	348
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

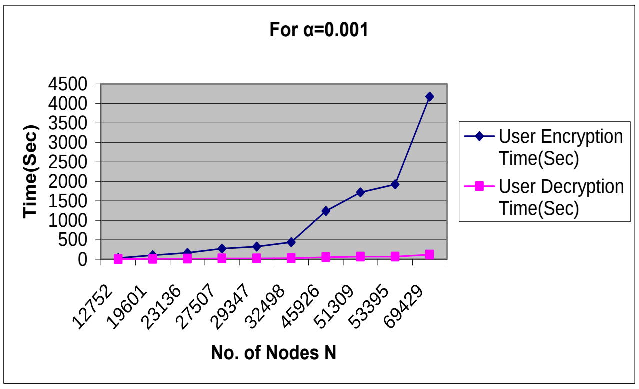

Fig. 15. CPU times for encryption and decryption vs. no. of nodes for α=0.001

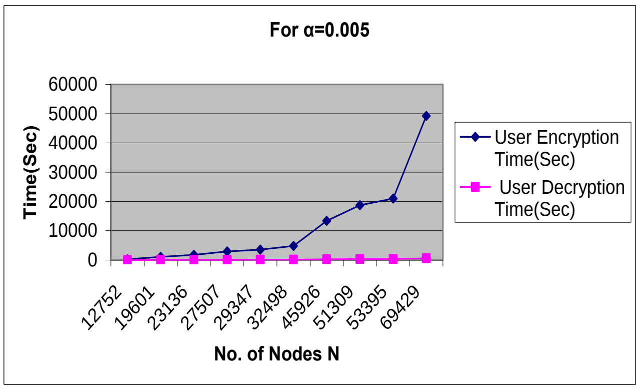

Fig. 16. CPU times for encryption and decryption vs. no. of nodes for α=0.005

encryption and decryption times. The matrix representation of the graph, however, would yield identical encryption and decryption times.

We already know that the parameter α is used to obtain a part of m-sequence from which two sets are formed by taking the positions of 0s and 1s. For a larger value of α, the length of the part of m-sequence is larger, and thus there will be more number of 0s and 1s in that part, resulting in more number of elements in the two sets. Thus, the Cartesian product of the two sets yields more number of edges to modify, which in turn increases the encryption and decryption time. This fact is reflected in Table III $i.e.$ as the value of α increases from 0.0001 to 0.005, the CPU time for encryption and decryption increases.

As mentioned earlier, the density ρ of a graph refers to the percentage of edges actually presents in the graph, whereas the degradation factor δ refers to the percentage of change in the number of edges in input graph due to encryption. When the density ρ of the graph decreases from $ibm01$ to $ibm10$, the probability of edge insertion during encryption increases than the edge deletion. As a result, the number of edges in the encrypted graph increases $i.e.$ the degradation factor δ increases from $ibm01$ to $ibm10$ for a given value of α. Since we use linked list implementation that introduces a difference in edge insertion and edge deletion times, the increase of δ from $ibm01$ to $ibm10$ leads to an increment in encryption and decryption time as well. Figures 13, 14, 15 and 16 respectively depict the variation of the CPU times with the number of nodes in the graph for encryption and decryption for four different values of α $i.e.$ 0.0001, 0.0005, 0.001 and 0.005.

VII. 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. We are currently investigating the benefit of the use of Cellular Automata (CA) [24] instead of LFSR in our proposed scheme, as the pattern generation in CA does not involve shifting of data that results into patterns with more randomness in nature as compared to LFSR.

References

[1] E. Charbon and I. H. Torunoglu, "On intellectual property protection," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, Orlando, USA, 2000, pp. 517–522.

[2] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Constraint-based watermarking techniques for design ip protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 10, pp. 1236–1251, 2001.

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

[4] D. Saha, P. Dasgupta, S. Sur-Kolay, and S. Sen-Sarma, "A novel scheme for encoding and watermark embedding in vlsi physical design for ip protection," in *Proceedings of the International Computing Conference: Theory and Algorithms, ICCTA '07*. Kolkata, India: IEEE Computer Society, March 2007, pp. 111–116.

[5] D. Saha and S. Sur-Kolay, "Encoding of floorplans through deterministic perturbation," in *Proceedings of the 22nd International Conference on VLSI Design, VLSID '09*. New Delhi, India: IEEE Computer Society, January 2009, pp. 315–320.

[6] A. T. Abdel-Hamid, S. Tahar, and E. M. Aboulhamid, "A public-key watermarking technique for ip designs," in *Proceedings of the conference on Design, Automation and Test in Europe, DATE '05*. Messe Munich, Germany: IEEE Computer Society, March 2005, pp. 330–335.

[7] G. Qu, "Publicly detectable watermarking for intellectual property authentication in vlsi design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 11, pp. 1363–1368, 2002.

[8] E. Castillo, L. Parrilla, A. Garcia, A. Loris, and U. Meyer-Baese, "Ipp watermarking technique for ip core protection on fpl devices," in *Proceedings of the International Conference on Field Programmable Logic and Applications*. Madrid, Spain: IEEE Press, August 2006, pp. 1–6.

[9] Y. C. Fan and H. W. Tsao, "Watermarking for intellectual property protection," *IEEE Electronics Letters*, vol. 39, no. 18, pp. 1316–1318, 2003.

[10] P.-C. Chen, Y.-S. Chen, and W.-H. Hsu, "A communication system model for digital image watermarking problems," *IAENG International Journal of Computer Science*, vol. 34, no. 2, November 2007.

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

[12] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital System testing and Testable Design*. Jaico Publishers, 2001.

[13] R. Halder, P. S. Dasgupta, S. Naskar, and S. S. Sarma, "An internet-based ip protection scheme for circuit designs using linear feedback shift register (lfsr)-based locking," in *Proceedings of the 22nd Annual Symposium on Integrated Circuits and System Design, SBCCI '09*. Natal, Brazil: ACM Press, 31st Aug–3rd Sep 2009.

[14] D. Kirovski and M. Potkonjak, "Localized watermarking: methodology and application to template mapping," in *Proceedings of the IEEE International Conference on the Acoustics, Speech, and Signal Processing, ICASSP '00*. Istanbul, Turkey: IEEE Computer Society, June 2000, pp. 3235–3238.

[15] E. Charbon, "Hierarchical watermarking in ic design," in *Proceedings of the IEEE Custom Integrated Circuits Conference*. Santa Clara, CA, USA: IEEE Press, May 1998, pp. 295–298.

[16] E. Charbon and I. Torunoglu, "Intellectual property protection via hierarchical watermarking," in *Proceedings of the International Workshop on IP Based Synthesis and System Design*, Grenoble, France, Dec 1998, pp. 776–781.

[17] J. Lach, W. Mangione-Smith, and M. Potkonjak, "Fpga fingerprinting techniques for protecting intellectual property," in *Proceedings of the IEEE Custom Integrated Circuits Conference*. Santa Clara, CA, USA: IEEE Press, May 1998, pp. 299–302.

[18] A. L. Oliveira, "Robust techniques for watermarking sequential circuit designs," in *Proceedings of the 36th annual ACM/IEEE Design Automation Conference, DAC '99*. New Orleans, Louisiana, USA: ACM Press, June 1999, pp. 837–842.

[19] I. Torunoglu and E. Charbon, "Watermarking-based copyright protection of sequential functions," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 3, pp. 434–440, 2000.

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

[21] M. Robshaw, "Stream ciphers," RSA Laboratories Technical Report TR-701, Version 2.0, Tech. Rep., 1995.

[22] S. V. Sathyanarayana, M. K. Aswatha, and K. N. H. Bhat, "Symmetric key image encryption scheme with key sequences derived from random sequence of cyclic elliptic curve points," *International Journal of Network Security*, vol. 12, no. 3, pp. 137–150, 2011.

[23] C. J. Alpert, "The ispd98 circuit benchmark suite," in *Proceedings of the 1998 international symposium on Physical design, (http://vlsicad.ucsd.edu/UCLAWeb/cheese/ispd98.html)*. Monterey, California, USA: ACM Press, April 1998, pp. 80–85.

[24] A. Seth, S. Bandyopadhyay, and U. Maulik, "Probabilistic analysis of cellular automata rules and its application in pseudo random pattern generation," *IAENG International Journal of Applied Mathematics*, vol. 38, no. 4, November 2008.

Raju Halder received the B.Sc. degree in Physics and the B.Tech. and M.Tech. Degrees in Computer Science from University of Calcutta, India, in 2002, 2005, and 2007 respectively. Currently, he is pursuing Ph.D. degree in Computer Science from Università Ca' Foscari Venezia, Italy. Meantime, he had worked in IBM India Pvt. Ltd. as Associate System Engineer during 2007-2008. His area of research interests includes Static Analysis, Abstract Interpretation, Databases, Security etc.

Parthasarathi Dasgupta received the B.Tech. degree in radiophysics and electronics and the M.Tech. and Ph.D. degrees in computer science from the University of Calcutta, India, in 1983, 1985, and 1997 respectively. He visited the Department of Computer Science and Engineering, University of California, San Diego as a Project Scientist during 2001-2002 (on leave from Indian Institute of Management Calcutta). He was nominated as an Invited Fellow of Japan Society for Promotion of Science (JSPS) in 2009-2010. He is currently a Professor of the Management Information Systems Group at Indian Institute of Management Calcutta (IIMC). He is a Senior member of IEEE and a member of ACM and ACM-SIGDA. His current research interests include VLSI CAD, E-Commerce and Graph algorithms.

Saptarshi Naskar is a technologist by birth (1980). His vision, fancies and prophesies are reflected in his role of signature analysis in fault detection. Rooted as physicists he did his masters in Computer Science and Application, and joined in University of Calcutta as U.G.C. research fellow in Computer Science and Engineering. Now he is faculty member in substantial post in a college under University of Calcutta. He has nearly twenty research papers published in peer reviewed journals and ancillary proceedings. His area of interests includes Combinatorial Algorithms-theory and practice, and Internet Technologies.

Samar Sen Sarma is founder faculty member of the department of Computer Science and Engineering, University of Calcutta, India. He is the first Ph.D. (Tech.) in Computer Science from University of Calcutta. He is a kind of teacher who believes in ancient Gurukul method. He has more than fifty papers published in peer reviewed journals and conferences. His area of interests includes aesthetic appeal in analysis and synthesis of algorithms with humanity in the forefront.