

Robust Verification of Public Marks in FPGA Design through a Zero-Knowledge Protocol

Debasri Saha and Susmita Sur-Kolay
A.C.M. Unit, Indian Statistical Institute, Kolkata, India
{debasri_r, ssk}@isical.ac.in

Abstract—With more integration in VLSI technology, Intellectual Property (IP) cores are reused to meet the customer's specifications in time. For intellectual property protection (IPP), various kinds of IP marks, such as watermarks, fingerprints, are embedded into the design for establishing the veracity of a legal IP owner. However, convincing public verification of such marks is not leakage-proof. Attackers not only manage to obtain potential clues to tamper public marks rendering public verification invalid, but can also suitably override the marks to include own signature, resulting in wrong public identification of IP vendor and IP buyer. Furthermore, current technique includes a sufficiently large set of public marks containing a header and a message body in addition to private ones to facilitate only public verification at the cost of significant increase in design overhead. We propose a zero-knowledge protocol to ensure robust and leakage proof convincing public verification with a smaller set of public marks consisting of only header part. We have tested our protocol for FPGA benchmarks and the results on robustness and overhead are encouraging.

Index terms—FPGA design, intellectual property protection, mark verification, zero-knowledge protocol.

I. INTRODUCTION

Design reuse has become essential to meet shrinking time-to-market and more integration on a single chip. Circuit components are available in electronic form, known as virtual components, and constitute the Intellectual Property (IP) of a design [1]. Instead of designing from scratch, appropriate IPs (may be soft or firm IPs) are assembled to satisfy the customer's specifications in time. However, misappropriation of IP is more probable [1]. Various kinds of IP marks, such as watermarks, fingerprints, are embedded into the design to ensure Intellectual Property Protection (IPP) through identification of legal IP owner (could be vendor or buyer).

On one hand, inclusion of robust marks incurring low design overhead is of major concern of IPP. On the other, convincing verification of marks while maintaining robustness of the process, i.e. without leaking any relevant information, is a prime issue. Mark verification is necessary during selling a design IP as well as later on, if any claim is raised. During selling, IP vendor's signature is extracted from the design to convince the buyer that the product is from a bonafide IP owner and buyer's signature is extracted to ensure buyer's protection against repudiation attack [5]. For convincing and easy verification, *public* verification is preferable where IP

vendor can verify the marks to IP buyer without direct intervention of verification team. However, it reveals the location of marks providing potential hints to the legal buyer for (i) removal/alteration of marks before illegal reselling for the destruction of legal proof, and (ii) overriding genuine watermarks to raise false claim for IP ownership.

In this paper, we address this issue by proposing a marking system equally robust for both public and private verification with low design overhead.

The paper is organized as follows. Section II assesses prior approaches. Our proposed protocol for public mark verification in VLSI FPGA design is given in Section III. Section IV presents the experimental results and the concluding remarks appear in Section V.

II. PREVIOUS WORKS

IPP techniques have immense significance in EDA. Constraint-based watermarking [1] maps the signature of a designer into a set of additional constraints, which are incorporated into the physical design. Embedding of IP vendor's watermark and buyer's fingerprint in the LUTs of FPGA is discussed in [2]. None of these discusses on public verification of IP marks.

A two level verification method, discussed in [3] works by splitting the watermark into two parts - public and private, where the first kind is publicly detectable. If public verification fails, then the private one is verified privately by an independent verification team. During public verification, the locations those host the public watermark are made public; hence public watermark is prone to be deleted or tampered rendering public verification inadequate and private verification essential. Moreover, the message body of public watermark is obtained by applying only a public one-way hash function on the header without usage of any secret key. This idea of keyless public watermarking [4] is vulnerable to overriding the header by the attacker's information, and then the message body accordingly resulting in wrong identification of IP owner without signaling any necessity for private verification. Hence, both these techniques are not effective. It is claimed that, as the design is an integrated process, it is unlikely that one can make a change without altering the behavior of the design. It can be argued that, if insertion of marks is possible through local change without affecting the behavior of the design greatly, deletion or tampering of the same is also possible with similar effects.

In this paper, a smaller set of public marks (with header only) containing IP owner's public information in encoded form, is embedded into the design and convincing public verification is performed on it through a Zero-Knowledge Protocol. The proposed public verification protocol Verify_ZKP is robust, leakage-proof and remains keyless. Moreover, the design overhead incurred due to large set of public marks is reduced.

III. PUBLIC MARK VERIFICATION THROUGH ZERO KNOWLEDGE PROTOCOL

A. Basics of Zero-Knowledge Protocol

Zero-knowledge protocol [6] is an interactive method between two parties so that one can prove to another that a statement is true, without revealing anything other than the veracity of the statement.

A zero-knowledge proof must satisfy three properties:

Completeness: If the statement is true, the honest verifier will be convinced of this fact by an honest prover.

Soundness: If the statement is false, no cheating prover can convince the honest verifier that it is true, except with some small probability called soundness error.

Zero-knowledge: If the statement is true, no cheating verifier learns anything other than the fact.

The basic idea of a zero-knowledge protocol is splitting the entire proof into two parts, say proof $P1$ and $P2$, each associated with an inherently *difficult* problem preferably in NP or NP-Complete complexity class. The prover and the verifier play several rounds of a two-person game. In each round, the prover arranges for proving either of the two $P1$ and $P2$ without any prior knowledge of which one will be asked for. The verifier randomly chooses one of the two parts to be proved. So after a constant t number of rounds, if the prover is able to prove the part asked for in each of the rounds, the statement stands proved to the verifier. This proof has a very small soundness error of 2^{-t} .

B. Overview of the Proposed Protocol Verify_ZKP

A zero-knowledge protocol for public mark verification is proposed here. The objective is to prove convincingly that the desired mark is present in VLSI FPGA design without revealing (i) the exact locations of the marks, and (ii) the exact signature string embedded as marks. Instead of verifying the marks in the original marked design, the protocol verifies the public marks in a mapped design derived from the original one. Such a method can be divided into two proofs:-

P1: the mapping itself between the original design and the mapped one without revealing the locations of the marks in either design;

P2: the presence of encoded public marks in the mapped design.

In our scheme, a set of private marks M_S similar to [3] and a set of public marks M_P are embedded in the design. The total set of marks $M = M_S \cup M_P$. M_P contains a bit -

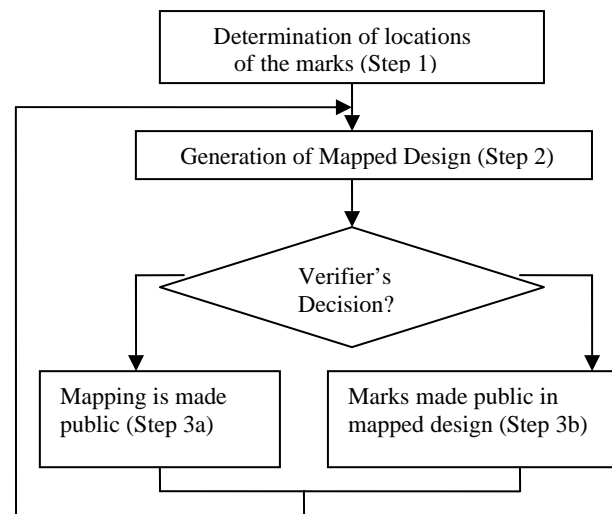


Figure 1. Main Steps of Verify_ZKP

string obtained by encoding IP owner's public information I_{pb} with a function h_r (for example, r shifts of maximal Non-Linear Feedback Shift Register NLFSR [7]). The locations of the marks in the original marked design are selected using a one-way-hash function with certain key K_C as seed. The values of r and K_C are private to the IP owner.

The main steps of the proposed verification technique Verify_ZKP are described below and shown in Figure 1.

Step 1 (Determination of location of the marks):

The IP owner (prover) inputs the values r and K_C to the verification tool. The locations of the marks in M are identified using K_C .

For each of t rounds, Step 2 is executed, and the IP owner (prover) performs one of the two options either Step 3a or Step 3b, as demanded by the buyer (verifier).

Step 2 (Generation of mapped design):

(i) **Location Mapping** — Let L be a set of possible locations of marks, say, mark-holders and C the set of bitstring contents possibly can be a mark. In the original design D_O , let l represents the placement, $l : C \rightarrow L$. Another placement $l' : C \rightarrow L$ is to be generated such that the distance function $d(l(c), l'(c))$, $c \in C$, is randomized. Thus a mapping function g is applied on l such that $g.l = l'$. This mapping conceals the locations of the marks during public verification.

(ii) **Content Encoding** — It conceals from the verifier the exact form of the signature string embedded as public mark, lest the bitstring acts as potential clue for searching the marks in the original design. h_r (r ' shifts of NLFSR) is applied on each $c \in C$ and $h_r(c)$ is placed in $l'(c)$ to generate the mapped design D_M .

If the verifier chooses Step 3a, IP owner discloses the location mapping (g) and the content encoding (h_r) to the buyer to prove that D_M is a mapped version of D_O .

If the verifier chooses Step 3b, IP owner publicly applies $(r + r')$ shifts of NLFSR on I_{pb} to compute $h_{r+r'}$ (I_{pb}). On the other hand, the bitstring associated with $l'(c)$, $c \in M_P$ in the mapped design D_M is made public. As $h_{r+r'}(I_{pb}) = h_r(h_r(I_{pb}))$, the equivalence of these two

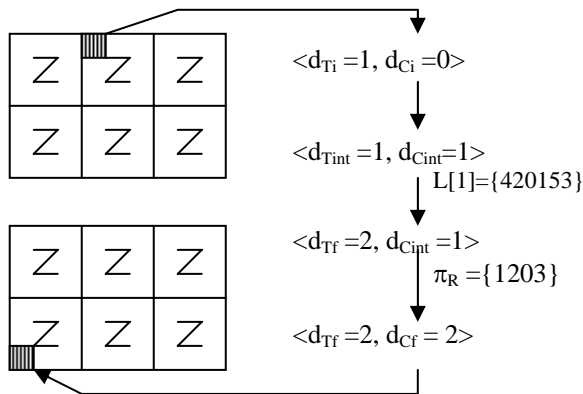


Figure 3. Location Mapping of a CLB in a Design with six tiles, each tile containing four CLBs

$$L_{PT} = \{ l(c) \mid c \notin M \cap l(c) \in L_T \}.$$

Steps for permutation π_R of CLBs mapped in tile T :

1. Select a subset $L_{RT} \subset L_{PT}$ with $|M_{PT}|$ locations for the CLBs in M_{PT} , and randomly arrange those CLBs into the locations in L_{RT} .
2. Compute the set of locations $L_{FT} = L_T - L_{RT}$, which are free for placing the CLBs without any public mark.
3. Permute rest of the CLBs of tile T into $|L_{FT}|$ locations.

The location mapping of a CLB is demonstrated in Figure 3.

Content Encoding: The configuration bitstring of each CLB is encoded using the same function h , used for mark preparation. Here a Non Linear Feedback Register NLFSR is used with primitive characteristic polynomial satisfying the properties of maximum period [7]. This type of NLFSR can produce a pseudo-random binary sequence, called DeBruijn sequence. Selective product terms are only included in the feedback function to balance between the non-linearity of the feedback function and the encryption speed. The configuration bitstring of each CLB is fed to the NLFSR as initial state $A_0 = \langle a_{-1}, a_{-2}, \dots, a_{-s} \rangle$ and the final state $A_{r'}$ after r' shifts be $\langle a_{r'-1}, a_{r'-2}, \dots, a_{r'-s} \rangle$. If $p = \langle p_1, p_2, \dots, p_s \rangle$ be the characteristics polynomial of the NLFSR, the final state is computed using the relation $a_k = \sum_{i=1}^s p_i * a_{k-i} + \sum_{i=1}^s \sum_{j=1}^s p_{ij} * a_{k-i} * a_{k-j} + \dots + p_{12..s} * a_{k-1} * a_{k-2} * \dots * a_{k-s}$ [Figure 4]. The final state $A_{r'}$ constitutes the encoded bitstring.

Content Encoding: Step 2(ii) of Verify ZKP
 Input: (i) D_O (ii) $l(c)$, $c \in M$ (iii) Location Mapping M_L
 Output: Final mapped design D_M
 1: Design a maximal NLFSR with s D-F/Fs, $s = \#$ bits in the LUTs of a CLB.
 2: Select an integer $r' \in [1, 2^s - 1]$ as $\#$ shifts, such that there are no two configuration bitstrings (c, c') of two CLBs satisfying $h_{r'}(c) = c'$ where $c \in M_p$ and $c' \in M$.
 3: For each configuration bitstring $c \in C$ generate encoded bit-string $h_{r'}(c)$.
 4: For each $c \in C$, associate $h_{r'}(c)$ with $l'(c)$ to generate mapped design D_M .

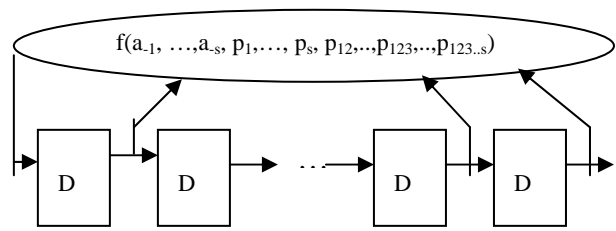


Figure 4. Bitstring Encoding using NLFSR

IV. EXPERIMENTAL RESULTS

The proposed scheme Verify_ZKP for FPGA design is implemented in C on a 1.2 GHz Sun Blade 2000 machine with Sun OS 9. IWLS'05 benchmarks and a number of real world designs are synthesized and placed correspondingly on the smallest possible FPGA devices.

Table I shows the overhead in terms of % increase in the usage of CLBs for both the method in [3] and Verify_ZKP. 256-bit private watermark and 284-bit public watermark consisting of 28-bit header and 256-bit message body (kept longer so that minute alteration in the header causes huge change in the design) are considered for the method in [3]. For Verify_ZKP, private mark (256-bit) is similar to [3], but the public marks only contain 128-bit public information I_{pb} in encoded form (a moderately long header). Message body is not needed in public watermark as the header is encoded and shown in mapped location.

For Verify_ZKP, Table II shows average value ($\mu(CD)$) and standard deviation ($\sigma(CD)$) of Manhattan distance CD of CLBs due to marking over all CLBs and over 15 rounds of interaction. Average value ($\mu(BD)$) and standard deviation ($\sigma(BD)$) of hamming distance BD of bitstrings are shown in Table III. Tables II and III reflect the robustness of public verification achieved through Verify_ZKP. The values of $\mu(CD)$ and $\sigma(CD)$ are the percentage of the corresponding values with respect to the half-perimeter of the corresponding CLB array, whereas the values of $\mu(BD)$ and $\sigma(BD)$ are the percentage of the corresponding values with respect to the length of a bitstring contained in a CLB.

V. CONCLUSIONS

For the first time, the principle of zero-knowledge protocol is effectively applied for secure verification of public marks in FPGA. Properties of Latin Square ensures distinct design mapping in each round. Previously, robustness of public verification is forced through hamming distance between the public marks of distinct companies, whereas, for Verify_ZKP, the more unpredictable the distance of the mapped design from the original one, the more robust is the proposed technique. Experimental results on FPGA benchmarks show that the Manhattan distance of FPGA CLBs and the hamming distance of bitstring have quite high mean and standard deviation values ensuring the strength of Verify_ZKP.

TABLE I
% REDUCTION IN OVERHEAD AS COMPARED TO METHOD IN [3]

Benchmark Circuits	#CLBs in Original Design	Method in [3]		Verify_ZKP	
		#CLBs in Marked Design	% increase in CLB usage	# CLBs in Marked Design	% increase in CLB Usage
SSRAM	51	56	9.804	54	5.882
RSA cipher	131	136	3.817	134	2.290
MD5	305	310	1.639	308	0.983
3DES	422	427	1.185	425	0.711
MIPS R2000	756	773	2.249	768	1.587
DES	875	892	1.943	887	1.371
AES	1792	1797	0.279	1795	0.167
ATR	1876	1893	0.906	1888	0.639
Average			2.728		1.703

TABLE II
ROBUSTNESS OF LOCATION MAPPING OF VERIFY_ZKP

Benchmark Circuits	# CLBs in Marked Design	Size of CLB Array	Computed Over CLBs		Computed Over Rounds	
			$\mu(CD)$	$\sigma(CD)$	$\mu(CD)$	$\sigma(CD)$
SSRAM	54	12 × 16	32.85	17.03	32.97	16.92
RSA cipher	134	12 × 16	32.82	16.98	32.98	16.95
MD5	308	20 × 24	33.17	17.79	33.23	17.43
3DES	425	40 × 48	34.11	18.38	33.58	17.28
MIPS R2000	768	32 × 32	33.52	18.02	33.37	17.78
DES	887	32 × 32	33.38	17.99	33.40	18.74
AES	1795	40 × 48	33.74	18.33	33.60	18.07
ATR	1888	40 × 48	33.72	18.34	33.63	18.04
Average			33.41	17.86	33.35	17.53

TABLE III
ROBUSTNESS OF BITSTRING ENCODING OF VERIFY_ZKP

Benchmark Circuits	# CLBs in Marked Design	Length of bitstrings in a CLB	Computed over CLBs		Computed over Rounds	
			$\mu(BD)$	$\sigma(BD)$	$\mu(BD)$	$\sigma(BD)$
SSRAM	54	128	48.43	22.62	48.37	22.67
RSA Cipher	134	128	48.65	22.73	48.41	22.75
MD5	308	128	48.68	22.87	48.22	22.86
3DES	425	128	48.32	22.92	48.45	22.72
MIPS R2000	768	32	47.85	21.38	47.83	21.50
DES	887	32	47.89	21.46	47.90	21.21
AES	1795	128	48.70	22.93	48.49	22.66
ATR	1888	32	47.91	21.73	47.95	21.59
Average			48.30	22.33	48.20	22.25

REFERENCES

- [1] A. B. Kahng, J. Lach, H. Mangione-Smith, Constraint-Based Watermarking Techniques for Design IP Protection, *IEEE Transactions on CAD/ICAS*, vol 20, No. 10, 2001, pp. 1236-52.
- [2] J. Lach, W. H. Mangione-Smith, M. Potkonjak, Fingerprinting Techniques for Field-Programmable Gate Array Intellectual Property Protection, *IEEE Trans. On CAD/ICAS*, vol 20, No. 10, 2001, pp. 1253-1261.
- [3] G. Qu, Publicly Detectable Techniques for the Protection of Virtual Components, *Proc. of Design Automation Conference*, 2001, pp. 474-479.
- [4] G. Qu, Keyless Public Watermarking for Intellectual Property Authentication, *LNCS 2137*, 2001, pp. 96-111.
- [5] D. Saha, S. Sur-Kolay, Fast Robust Intellectual Property Protection for VLSI Physical design, *Proc. of Intl. Conference on Information Technology*, 2007, pp. 1-6.
- [6] S. Goldwasser, J. C. Lagarias, A. K. Lenstra, K. S. McCurley, A. M. Odlyzko, Cryptology and Computational Number Theory, *Proc. of Symposia in Applied Mathematics*, 1989.
- [7] M. Soriano, Stream Ciphers based on NLFSR, *IEEE Intl. Telecommunications Symposium*, 1998, vol 2, pp 528-533.
- [8] H. Sagan, Space-Filling Curves, Springer-Verlag, 1994.
- [9] J. Denes, A. D. Keedwell, Latin Squares, N Holland, 1991.