Hardware Protection and Authentication Through Netlist Level Obfuscation

Rajat Subhra Chakraborty
Dept. of Electrical Engineering and Computer Science
Case Western Reserve University
Cleveland, Ohio 44106, USA
e-mail: rsc22@case.edu

Swarup Bhunia
Dept. of Electrical Engineering and Computer Science
Case Western Reserve University
Cleveland, Ohio 44106, USA
e-mail: skb21@case.edu

Abstract—**Hardware Intellectual Property (IP) cores have emerged as an integral part of modern System–on–Chip (SoC) designs. However, IP vendors are facing major challenges to protect hardware IPs and to prevent revenue loss due to IP piracy. In this paper, we propose a novel design methodology for hardware IP protection and authentication using netlist level authentication. The proposed methodology can be integrated in the SoC design and manufacturing flow to provide hardware protection to the IP vendors, the chip designer, and the system designer. Simulation results on ISCAS–89 benchmark circuits show that we can achieve high levels of security through a well–formulated obfuscation scheme at less than 10% area overhead under delay constraint.**

Index Terms — **Design for security, IP piracy, hardware authentication, hardware obfuscation, hardware protection.**

I. Introduction

The increase in SoC design complexity in recent years has made reuse–based design using hardware *Intellectual Property* (IP) cores extremely common [3]. These IP cores are in the form of synthesizable Register Transfer Level (RTL) descriptions in Hardware Description Languages (HDLs), or synthesized gate-level netlists. It is quite common to have SoC designs where multiple IPs from different IP vendors are integrated by the chip designer.

Unfortunately, recent trends of IP piracy and reverse–engineering efforts to produce counterfeit ICs have raised serious concerns among the IC design community [1, 2]. It can take several forms – (a) IPs being used without paying the requisite fees to the IP vendor [3]; (b) design houses illegally selling IPs obtained from IP vendors to other parties; (c) fabrication houses manufacturing and selling illegal copies of a design without paying royalty fees to the design house [4], or (d) companies performing post–silicon reverse-engineering to derive the GDS–II database of an IC to manufacture illegal "clones" [5]. Whatever might be its form, IP piracy affects the IP vendors, chip design houses as well as system manufacturers adversely by depriving them of their revenue and market–share.

In this paper we develop a low–overhead, piracy–proof design flow for SoCs that benefit the IP vendors, the chip designers, the system designers and end users of SoC based products. Our approach is based on obfuscation of the design at the gate–level. This is by modification of few selected nodes of the design and the state transition function, followed by a resynthesis of the design. The objective of the modification process is to enforce undesired logic values at the primary outputs and state element inputs unless a sequence of patterns appears in the primary inputs. The enabling pattern or key can vary from one instance of the IP to another. The key effectively embeds authentication capabilities to the design, which helps to establish the legal source of the design in case of litigation, all the while ensuring that the degradation in the user experience is minimal. As far as we know, this is the first work that explores gate–level obfuscation as a method to embed anti–piracy features in the design flow, and ensures both obfuscation and authentication capabilities to all parties associated with it.

The rest of the paper is organized as follows. Section II presents previous work in this field, and indicates the salient features of this work. In Section III, we describe the low–overhead hardware obfuscation scheme in details, and develop the *ObfusFlow* (**Ob**fuscated Design **Flow**) design methodology based on the proposed scheme. In Section IV we present simulation results for a set of benchmark circuits that show the effectiveness of the proposed hardware obfuscation scheme. We conclude in Section V.

II. Background and Salient Features

The issue of IP piracy and ways to mitigate it have been researched extensively in recent years. Existing solutions to this problem can be broadly classified into two main categories: a) Obfuscation, and b) Authentication.

Obfuscation based IP Protection: The problem of *Design Obfuscation* can be viewed from multiple perspectives [6, 7]. In [6], the solution proposed is to modify the HDL (RTL) source code by removing the comments and changing the internal net–names following a simple string–substitution strategy. This is done in such a way that the functionality of the system remains unchanged, but the logic description becomes incomprehensible. In [7], the design source–code text file is encrypted using an advanced encryption technique such as AES, and then decrypted at the user end (using the decryption key provided by the IP vendor). However, these approaches do not modify the hardware structure and functionality and thus cannot provide hardware protection at all levels of design flow.

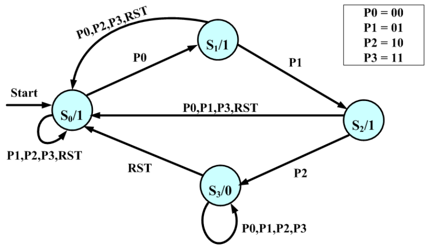
Authentication based IP Protection: Several "Digital Authentication" techniques have been proposed that aim to protect either the rights of the IP vendor [9–11], or that of the chip designer [4, 12–14]. Most of the approaches directed towards benefiting the IP vendor aims to embed a *Digital Watermark* in the design, which helps in authenticating the design at a later stage. Since this digital watermark (or signature) cannot be removed from the IP, it is easy to prove an illegal use of such a component in litigation. Some approaches [4, 14] propose to use a "lock–and–key" approach, such that unless a particular input pattern (termed a "key") is provided, the circuit will not operate in its intended mode. However, these approaches do not obfuscate the design and hence, cannot prevent reverse engineering the design and unintended use of the IP.

Salient Features of the Proposed Approach: Our approach is different from the approaches proposed so far in the sense that *we focus on changing the functionality and structure of the IP core* (by modifying the gate–level netlist), so that it *both obfuscates the design and embeds authentication features in it*. Moreover, these features are present in the design at every stage of the design flow, and thus the rights of all concerned (the IP vendor, the design house, and the system designer) are preserved, and all of them play an active role in making the design flow a secure one. The IP is protected from unauthorized manufacturing by the fact that the system designer depends on input from the chip designer to use the IC. Consequently, the manufacturing house cannot simply manufacture and sell un–authorized copies of the IC without the knowledge of the design house. In addition, it is transparent to the end user who has the assurance of using a product that has gone through a secure design flow. We prefer obfuscation at gate–level because:

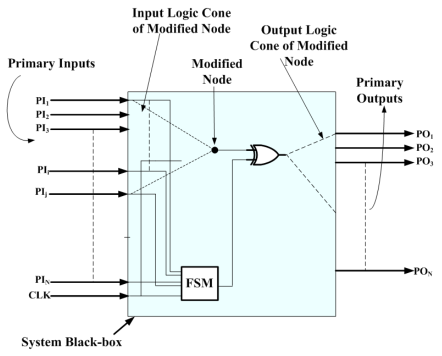
1) Obfuscation of the RTL level description of a circuit is substantially more challenging than the gate–level description. The characteristic lucidity and terseness of a high–level, behavioral description of a circuit makes any design modification much more visible than one performed at the structural level (such as gate–level).
2) In many cases the IP is transferred in the form of a gate–level design by the IP vendor, and hence the RTL description of the design is not available [8].

III. The *ObfusFlow* Design Methodology

The proposed obfuscation approach is essentially one of *obfuscating the functionality of the IP core by structural modifications*. The structural modification is realized by modifying selected internal circuit nodes and the state transition function (assuming a sequential circuit) in a manner that forces undesired values at internal nodes and state element inputs, unless a specific sequence of patterns appear at the primary inputs that enables normal operation. After the modification, the circuit is resynthesized to hide obvious change in the logic structure. To modify the state transition function, we propose to insert a simple Finite State Machine (FSM) (with low hardware overhead) in the IP. The inserted FSM has the



(a) FSM State Diagram



(b) Node Modification

Fig. 1. An example of netlist level obfuscation in a sequential circuit. The key step to accomplish the obfuscation goal is to select a set of nodes in the circuit and modify them to provide wrong outputs until the circuit reaches a pre–defined state on application of a sequence of patterns (referred to as "*key*") at the primary inputs.

primary inputs of the circuit as its inputs, (beside the clock and reset signals), and has one output. The FSM defines two distinct modes of operation of the IP core: *normal* and *obfuscated*. At the start of operations, the FSM is reset to its initial state (with output at logic-1) and then, depending on the applied input it goes through a state transition sequence. Only on receiving N particular input sequences in order, it goes to a state where its output becomes logic–0, and it stays in that state until reset to the initial state. Fig. 1(a) shows the state diagram of such a FSM with $N = 3$ required patterns in the enabling sequence and $N + 1 = 4$ states.

The output of this FSM is XOR–ed with a few selected nodes of the circuit. Thus, initially when the FSM output is at logic–1, the logic values at the modified nodes are inverted and the circuit behaves differently compared to the unmodified circuit. However, once the output of the FSM settles to logic–0, the circuit is restored to its original behavior. If there are M primary inputs (other than the clock, reset and power supply), and if N state transitions are required to make the FSM output logic–0, the probability that a random application of N consecutive vectors at the primary inputs would be able to bring the IP to its *normal* mode is $\frac{1}{2^{MN}}$, which is exponentially small depending on the product $M{\cdot}N$. For example, if $M = 16$ and $N = 16$, $P \sim 10^{-77}$. Hence, we can assume that it is practically infeasible to reverse engineer the correct functionality of the IP by functional simulation.

Choosing the Optimal Set of Nodes: The proposed design modification scheme should provide *maximum robustness* to

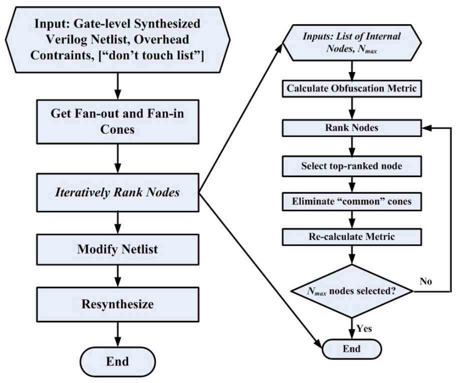

Input: Gate-level Synthesized Verilog Netlist, Overhead Contraints, ["don't touch list"]
Get Fan-out and Fan-in Cones
Iteratively Rank Nodes
Modify Netlist
Resynthesize
End
Inputs: List of Internal Nodes, N_{max}
Calculate Obfuscation Metric
Rank Nodes
Select top-ranked node
Eliminate "common" cones
Re-calculate Metric
N_{max} nodes selected?
No
Yes
End

Fig. 2. Proposed Design Flow and Iterative Ranking Algorithm

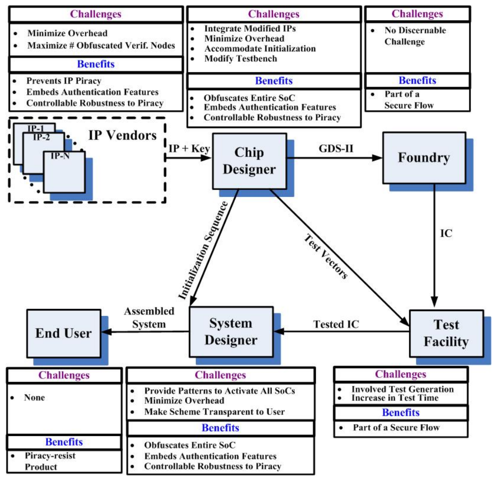


Fig. 3. Challenges and Benefits of the *ObfusFlow* Design Methodology

reverse–engineering approaches at *minimal hardware overhead*. To achieve this goal under given constraints, we need to choose an *optimal set of nodes* to be modified, so that maximum obfuscation is achieved at minimal overhead. We followed a ranking procedure where nodes with larger fan–out and fan–in cones are preferred. This is because large fan–out and fan–in cones imply the modified node affects a large number of internal nodes and primary outputs.

Iterative Ranking Algorithm: Once the metric has been calculated for the nodes in a gate–level design, the nodes are ranked following a *iterative ranking algorithm*. The ranking is a *multi–pass* procedure, with the metric for each node being dynamically modified based on the selection of the nodes in the last iteration. The algorithm takes into account the overlap of the fan–out cones of the nodes which have been already selected, and eliminates them from the fan–out cones of the remaining nodes. On the completion of each iteration, the top ranking node among the remaining nodes is selected, so that selection of N_{max} nodes would take N_{max} iterations. In this way, as the iterations progress, the nodes with more non–overlapping fan–out cones are assigned higher weight.

Design Flow based on Hardware Obfuscation: The IP vendor applies the hardware obfuscation scheme to create the modified gate–level netlist and re-synthesizes it into a flattened netlist. This re-synthesis helps to structurally obfuscate the netlist from the original unmodified one. Note that the same technique can be adopted by a design house which designs its own IPs. Fig. 2 shows the IP obfuscation design flow adopted by the IP vendor. An optional *don't touch* list of nodes (e.g. nodes on the critical path) can be provided at the start of the design flow, which are not to be modified.

The IP vendor then supplies the modified IP to the design house, along with the activating sequence. The design house receives one or multiple IPs from IP vendors, and then integrates them on chip. To activate different IPs, the designer needs to include a low-overhead "controller module" in the SoC, that will steer different initialization sequences to the different IP blocks. This *controller module* has an integrated FSM which determines the steering of the correct input sequences in correct order to a specific IP block. The designer must also modify the test–benches accordingly to perform block–level or chip–level logic simulations.

The manufacturing house manufactures the SoC from the design provided by the design house, and the test engineer performs post-manufacturing testing using the set of test vectors provided by the designer. The tested ICs are passed to the system designer along with different initialization sequences from the design house.

The system designer integrates the different ICs in his board-level design, and arranges to apply the initialization patterns during "booting" or similar initialization phase. Thus, the initialization patterns for the different SoCs need to be stored in Read Only Memory (ROM). In most SoCs composed of multiple IPs, several initialization cycles are typically needed at start–up to get into the "steady–stream" state, which requires accomplishing certain tasks such as initialization of specific registers [15]. The system designer can easily utilize this inherent latency to hide the effect of the initialization sequences from the end user. The system designer in turn benefits from the fact that the board is unusable until the correct initialization patterns have been stored in the ROM. Finally, this secure system is used in the consumer product, which provides the end-user with the assurance that the components have gone through a secure and piracy-proof design flow. Fig. 3 shows the challenges and benefits of the design flow from the perspectives of different parties associated with the flow.

Authentication Features of the Design Flow: To embed authentication features in the design, the IP vendor can design the IP with different activation sequences for designs supplied to different design houses. This will help to trace back to the source from where the IP was illegally leaked in case of litigation.

IV. Results

In this section we present simulation results to show the effectiveness of the proposed hardware obfuscation methodology for a set of ISCAS–89 benchmark circuits [16]. The benchmark circuits were synthesized using Synopsys Design

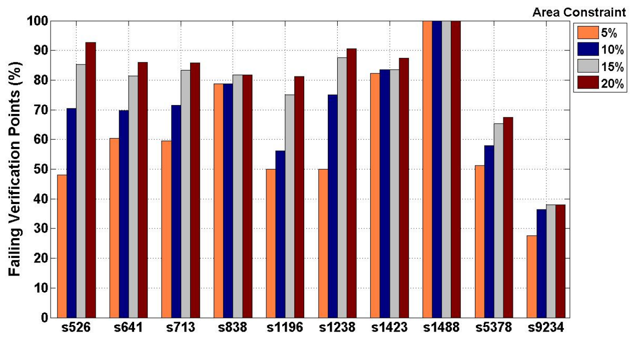


Fig. 4. Verification Failures for ISCAS–89 Circuits

TABLE I
DESIGN OVERHEAD (%) WITH THE PROPOSED OBFUSCATION SCHEME

Overheads(%)	**5%** area constraint			**10%** area constraint		
Circuit	**Area**	**Delay**	**Power**	**Area**	**Delay**	**Power**
s526	3.64	0.00	7.06	8.12	0.00	16.17
s641	4.96	-3.66	8.20	9.74	-3.66	13.90
s713	4.06	-3.66	7.34	9.07	-3.66	14.69
s838	2.20	-2.39	6.92	9.00	-8.57	9.76
s1196	3.96	0.00	6.04	6.06	0.00	16.09
s1238	4.52	-0.42	6.52	9.99	-0.42	9.97
s1423	4.70	-0.78	8.02	9.61	-2.64	14.91
s1488	3.27	-2.79	3.13	8.65	-0.93	8.33
s5378	4.34	0.00	8.91	9.87	0.00	13.80
s9234	4.74	0.00	5.80	8.82	3.60	12.37
Avg.	4.04	-1.37	6.79	8.89	-1.63	12.99

Compiler with optimization parameters set for minimum area and mapped to a LEDA 250nm standard cell library. The flow was developed using the TCL scripting language, and was directly integrated in the Design Compiler environment. Synopsys *Formality* was used for formal verification between the original and the modified design. The verification points considered by Formality constituted of the inputs of state elements (e.g. flip-flops) and primary outputs.

A simple four–state FSM was designed for each of the benchmarks, and integrated into them. The maximum number of modifiable nodes N_{max} for each benchmark circuit was determined considering four different area constraints (5%, 10%, 15% and 20%). In all cases the number of modified nodes was less than 13% of the total number of nodes. The benchmarks were then subjected to the hardware obfuscation flow. A prediction of the number of verification failures between the original and the modified design was obtained from the tool. This value was obtained by analyzing the fan–in logic of verification points and determining how many of them are affected by these modifications.

Fig. 4 shows the verification failure percentage of state elements and primary outputs of the benchmarks obtained. It was found the observed failure matches closely with the predicted failure, with the average prediction error less than 5%. This was in spite of the fact that the resynthesized design differed considerably in logic structure from the original design. Table I shows the design overheads incurred for different benchmark circuits, for 5% and 10% area constraints overhead. From the table, it is evident that the actual area overheads were smaller than the imposed constraints in all cases, while the timing overhead was negative, i.e., the timing constraint was met with positive slack in most cases. The power overhead was within acceptable limits in all cases.

V. CONCLUSION

We have presented a netlist–level hardware obfuscation based anti–piracy design flow for SoCs. It involves active participation of the IP vendor, the IC designer and the system designer, and helps to preserve the rights of all of them. The scheme is based on modification of the gate–level netlist of a pre–synthesized IP core, followed by re–synthesis to obtain maximum functional and structural obfuscation at minimal overhead. Simulation results with a set of ISCAS–89 benchmark circuits show that this scheme is capable of providing predictable and high levels of design obfuscation at nominal area overhead under delay constraint. Future work would involve further reduction in design overhead and extension of the proposed obfuscation scheme to register transfer level level circuit descriptions.

REFERENCES

[1] "VSI Alliance - IP Protection Development Working Group. The value and management of intellectual assets, 2000." [Online]. http://vsi.org/documents/datasheets/TOCIPPWP210.pdf
[2] R. Colin Johson, "Antipiracy scheme aims protect chip makers". [Online]. Available: http://www.eetimes.com
[3] A.B. Kahng, et al, "Constraint-based Watermarking Techniques for Design IP Protection, *IEEE TCAD*, vol. 20, no. 10, pp. 1236-1252, Oct. 2001.
[4] F. Koushanfar and G. Qu, "Hardware Metering", *Proc. DAC*, 2001.
[5] D.C. Musker, "Protecting and Exploiting Intellectual Property in Electronics", *Proc IBC Conferences*, 1998.
[6] "Thicket^{TM} Family of Source Code Obfuscators". [Online]. http://www.semdesigns.com
[7] T. Batra, "Methodology for protection and Licensing of HDL IP". [Online]. http://www.us.design-reuse.com/news/?id=12745\&print=yes
[8] "Designware USB Solutions". [Online]. http://www.synopsys.com/products/designware/usb_solutions.html
[9] E. Castillo, et al, "IPP@HDL: Efficient Intellectual Property Protection Scheme for IP Cores", *IEEE TVLSI*,vol. 15, no. 5, pp. 578-590, May 2007.
[10] I. Cox, M. Miller, and J. Bloom, "Digital Watermarking: Principles and Practice", *San Mateo,CA: Morgan Kaufmann*, 2001.
[11] A.B. Kahng, et al, "Watermarking techniques for intellectual property protection", *Proc. DAC*, 1998
[12] F. Koushanfar and M. Potkonjak, "CAD-based Security, Cryptography, and Digital Rights Management", *Proc. DAC*, 2007.
[13] Y. Alkabani, F. Koushanfar and M. Potkonjak, "Remote Activation of ICs for Piracy Prevention and Digital Right Management", *Proc. ICCAD*, 2007.
[14] J.A. Roy, F. Koushanfar and I.L. Markov, "EPIC: Ending Piracy of Integrated Circuits", *Proc. DATE*, 2008.
[15] W.A. Moore and P.A. Kayfes, "US Patent 7213142 – System and method to initialize registers with an EEPROM stored boot sequence (2007)". [Online]. http://www.patentstorm.us/patents/7213142/description.html
[16] [Online] http://www.fm.vslib.cz/~kes/asic/iscas/