A comprehensive hardware/software infrastructure for IP cores design protection

Brice Colombier, Lilian Bossuet, Ugo Mureddu
Univ Lyon, UJM-Saint-Etienne, CNRS Laboratoire Hubert Curien UMR 5516
F-42023, Saint-Étienne, France
{b.colombier, lilian.bossuet, ugo.mureddu}@univ-st-etienne.fr

David Hély
Univ. Grenoble Alpes, LCIS
F-26000, Valence - France
david.hely@lcis.grenoble-inp.fr

***Abstract*—Core-based design, which is widely used nowadays due to the high complexity of electronic systems, comes with specific threats against design data. Cases of intellectual property infringement and illegal copying have risen in the last decade. To fight this threat, must be aware of how many instantiations of an IP core have been carried out. Based on this, illegal copies can be detected and precise metering is achieved. To work toward this goal, we propose a comprehensive hardware/software infrastructure that allows a designer to modify an IP core to make it remotely activable later on when it is implemented on an FPGA. We focus on industrial applicability and ease of integration. On the one hand, hardware implementation on FPGA focuses on achieving a medium level of security at reduced cost. On the other hand, the software side aims at computational efficiency and industrial applicability for smooth integration into EDA tools.**

I. Introduction

A design-and-reuse paradigm has been widely adopted by the microelectronics industry in the last decades. Indeed, due to the increasing complexity of integrated circuits, independent functional modules must be developed separately before being integrated in a large, complex system. These functional modules, called Intellectual Property (IP) cores, are provided by designers to system integrators. However, once the IP core has been sold, the original designer has no way of knowing how many times the IP core is actually instantiated. This situation prohibits metering [1] and has two consequences. First, the designer cannot know if illegal copies of the IP core have been carried out. Second, it inhibits the development of interesting licensing possibilities such as evaluation or premium modes, that could only be built on top of precise metering.

In order to fight against these threats, an IP core integrated into an FPGA should be modified to make it remotely activable in a secure manner. This is only possible by combining multiple features into a complete IP protection scheme. One requirement is to uniquely identify IP core instances, using a Physical Unclonable Function (PUF) for instance [2]. The IP core must also be unusable before activation. The FSM [3], the communication bus [4] or the combinational logic [5] can be modified to this end. Finally, a cryptographic core is employed for security.

However, very few works deal with the actual integration of all these components into a complete, industrially usable system. We make a proposal here to take these considerations into account, with a focus on usability, low resources occupancy and integration into EDA tools. We provide a detailed workflow that could be easily integrated into existing EDA tools and standard design flow. Implementation results show that the logic resources cost of implementing the overall IP protection scheme is manageable and compatible with real-life designs. The code associated with this article is fully open-source[1].

Section II proposes a threat model before deriving the requirements for an IP protection scheme. Section III gives an overview of our proposal, integrating both software and hardware components, described in Sections IV and V respectively. Experimental results obtained from FPGA implementation are given in Section VI. Section VII concludes the article.

II. Objectives and related work

A. Threat model

1) Attacker model: When an attacker carries out illegal copying, the objective is to instantiate an IP core more times than originally agreed with the IP core designer. From the attacker's point of view, instantiating the IP core as a black box is enough. Thus, in this frame, the attacker is not interested in acquiring knowledge about the internals of the IP core.

We assume that the attacker can legally obtain a copy of the IP core. The attacker also has all the technical resources that it takes to instantiate the IP core in a correct manner.

2) Defender model: The aim of the defender is to prevent the attacker from instantiating the IP core as a black box without having to report it to the designer. In addition, the defender must be able to control, not only know, how many of these instances actually operate. The fact to control how many instances of an IP core are running is called *metering* [1].

When protecting design data, the designer's objective is not to obtain absolute security and resistance against all sorts of attackers. Instead, illegal actions must be made prohibitively expensive The data protection scheme would then deter attackers if it is more costly to perform illegal copying than to develop the IP core in-house or purchase it legally.

The cost for the defender must be limited as well. Indeed, to foster industrial adoption, a realistic technique must be proposed and the cost to implement the IP protection system must be lower than the financial losses associated to illegal copying suffered by the designer. This cost is essentially due

[1] https://gitlab.univ-st-etienne.fr/b.colombier/demonstrator

to extra logic resources occupied on the hardware side, that should be as lightweight as possible.

B. High-level requirements

Having a threat model allows to define a set of high-level requirements for the IP protection scheme, which should be:

- **Easy to use** by a legitimate user.
- **Hard to circumvent** by an attacker.
- **Lightweight** in logic resources to limit the cost.
- **Efficient** at making illegal copies unusable.
- **Stealthy** so that legitimate copies operate normally.
- **Integrable** in a standard design flow.
- **Universal** and applicable to all sorts of IP cores.

C. Combination of solutions

Some proposals have been made in the literature that associate several components into an IP protection scheme. In [6], the activation inputs of a logic masking module are controlled by the response obtained from a PUF. This response is compared to a value stored in memory, fed by the system integrator. Following the principles of public key cryptography, the system integrator obtained the PUF response from the designer after it has been encrypted on chip by the designer's public key and decrypted by the designer with his private key. In [7], a locking FSM is used to control the activation inputs of a logic masking module. The transitions between the extra states of the boosted FSM depend on a PUF response. In [8], a boosted FSM in integrated with a PUF, so that the start-up state depends on the PUF response. This makes the set transitions to the original start-up state dependent on the PUF response. Therefore, the set of transitions is device-specific and is a condition to unlock the IP core. Finally, in [9], a key is common to all instances of the IP core. This key is compared to one stored in a non-volatile memory and the result of this comparison triggers the blowing of specific anti-fuses located in the input-output blocks. These associations of previously described solutions are summarised in Table I.

TABLE I: Association of solutions for complete IP protection.

	[6]	[7]	[8]	[9]
Identification of IP core				Global identifier
Identification of instances	PUF	PUF & identifier	PUF	
Degraded operation	logic masking	FSM locking & logic masking	FSM locking	Anti-fuse locking
Cryptography primitive	Elliptic curve			

Only one of these solutions [6] relies on a proper cryptography primitive for security. All other solutions are hence not secure. However, an elliptic-curve core, as used in [6], offers strong security guarantees but is very costly to implement. Such long term security primitives are not suited for our case. Instead, mid-term security, typically achieved with 80-bit symmetric keys, is a better trade-off for security at limited cost.

III. Overview of our proposal

Our proposal for an IP protection module is shown in Figure 1. Individual components are detailed next. The typical use case is coming in three distinct steps. At **design phase**, the IP core is modified so that it can be remotely activated. The module shown in Figure 1 is added. At the **enrolment phase**, a reference response is obtained from the PUF and stored on a server. This is intrinsic to each device and allows to identify it uniquely. Later on, during the **activation phase**, the PUF response is generated again and the errors are corrected between the response on the server and the one stored on the device. This PUF response is then used to encrypt an activation word (AW) on the server, that is fed to the device. The lightweight block cipher decrypts it internally, using the PUF response as a symmetric key. The decrypted AW is then fed to the modifications done at the combinational logic level, so that the circuit operates correctly.

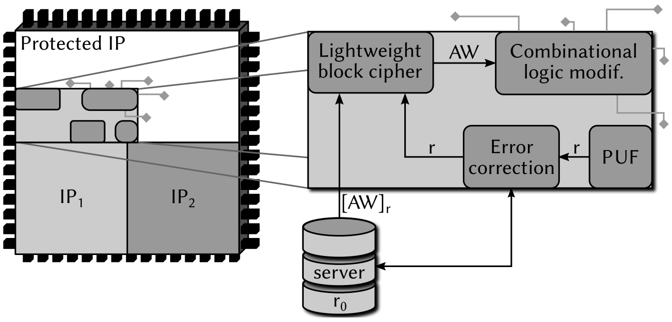


Fig. 1: Schematic of the IP protection module

IV. Software components

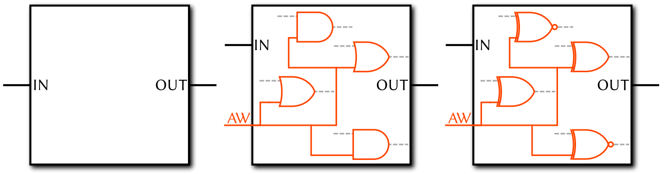
The software components described below are meant to be integrated into the EDA tools. They consist in modifying the IP core so that it does not operate properly before activation. Since we want the overall IP protection scheme to be as general as possible, we chose to carry out the aforementioned modifications at the combinational level. Extra inputs are then added to the IP core, that must be driven by the correct AW to get the IP core to operate normally. Moreover, for smooth integration into the EDA tools, the computational complexity of these modifications should be limited to make them usable. We investigated combinational logic locking [10] and logic masking [5]. Once the netlist has been modified, an associated AW is obtained and must be stored.

A. Combinational logic locking

Combinational logic locking [10] consists in modifying the combinational logic of an IP core so that its outputs can be controllably forced to a fixed logic value. It is implemented by first converting the IP core netlist into a directed acyclic graph, before identifying the sequences of nodes that propagate a fixed logic value to an output. The computational complexity of this method is limited and very large netlists can be processed. The logic resources overhead brought by inserting the extra logic gates necessary for combinational logic locking is 2.9% on average. Thus it comes at reasonable cost for the designer.

B. *Logic masking*

Logic masking [5] aims at inserting XOR/XNOR gates at specific locations in the netlist so that the internal state can be controllably altered if an incorrect value is fed to these gates. Depending on the size of the netlist to analyse, in order to remain computationally feasible, the node selection heuristic can be different. For small netlists of up to 5 000 gates, fault-analysis can be used [11]. However, it quickly becomes computationally impractical. For very large netlists of more than 100 000 gates, random selection, as originally proposed in [5], is the only method that has reasonable runtime. For netlists of intermediate size, recently proposed centrality indicators [12] can be used. The more masking gates are inserted, the more efficient the logic masking is. The designer can then pick the associated logic resources overhead. Figure 2 summarises how an IP core, shown in Figure 2a is modified by logic locking (see Figure 2b) or by logic masking (see Figure 2c).



(a) Original IP core (b) Locked IP core (c) Masked IP core

Fig. 2: Modifications of combinational logic for IP protection with additional inputs and logic gates in orange.

V. Hardware components

After the IP core has been modified by combinational logic locking or logic masking, it cannot be directly used as is. Indeed, exposing the activation input directly makes the modified IP core subject to attacks aiming at recovering the AW [13]. Moreover, in this case, the AW is the same for every instance of the IP core, preventing metering. Therefore, extra hardware components must be added as well.

A. *Mandatory components*

1) Lightweight block cipher: In order to ensure a sufficient level of security, a cryptographic block is essential. A lightweight block cipher can be used to this end, like PRESENT [14]. In our IP protection scheme, the block cipher is used to decrypt the AW internally and feed it to the logic locking/masking module. The symmetric key, owned both by the server and the device, is the PUF response.

2) Physical Unclonable Function: A PUF is integrated in our proposal as well so that individual instances can be identified. We chose to integrate a PUF based on Transient-Effect Ring Oscillators (TERO) [15]. The TERO-PUF exhibits low error-rate and is feasible on multiple FPGA families [16]. However, just like any PUF, its responses are noisy and cannot be used directly as a symmetric key for a lightweight block cipher. An error-correction module is then necessary.

3) Error-correction module: A reliable identifier from a PUF response can be obtained with error-correcting codes [17]. However, those require a large amount of logic resources. A recent solution is to use the `CASCADE` key reconciliation protocol to correct the errors found in PUF responses [18]. It can accommodate the error rate observed for the TERO-PUF.

B. *Optional components*

Besides the mandatory components described above, some extra modules could be added to enhance the security or bring additional features, at the cost of using more logic resources.

1) Hash function: After the error correction has been carried out on the PUF response, some bits of information have been leaked. They prevent the PUF response to be used directly as a cryptographic key. Typically, a large PUF response is requested to account for the leakage induced by the error correction step. Moreover, the responses obtained from the PUF do not have the statistical properties to be used as a cryptographically safe encryption key. Considering the relaxed security requirement of the use case we consider here, this might not be an issue. However, when considering to be in the random oracle model, a hash function can be used to derive a cryptographic key from the PUF response, like in [19] where the SPONGENT lightweight hash function is used. It is up to the designer to decide if the application requires the security brought by the hash function for the extra cost induced in logic resources.

2) Watermark: A small watermark can be integrated in the IP core, so that the designer can claim ownership of the IP core. Typically these watermarks are better detected by side-channel, since this is the less constraining way of verifying it. For example, the watermark can be detected in the power consumption [20] or by electromagnetic analysis [21].

VI. Implementation results

We implemented a proof of concept of our IP protection scheme on Intel Cyclone V and Microsemi SmartFusion 2 FPGAs. Implementation results are given in Table II.

TABLE II: Device-side implementation of the IP protection module

	Intel Cyclone V		Microsemi SF2	
	6-LUTs	DFFs	4-LUTs	DFFs
TERO-PUF	4841	160	2258	158
Response shift register	0	128	0	128
Communication	321	2560	2664	2478
IP protection module	**444**	**357**	**1030**	**376**
MUX indexes 128x7:7	301	0	595	0
MUX response bits 128:1	37	0	85	0
One time pad	128	0	128	0
AW storage	0	128	0	128
`CASCADE` module	1	1	1	1
Controller	104	90	101	69
Total	5606	2949	5746	2803

The TERO-PUF is the most expensive primitive in terms of logic resources. However, this heavy cost is due to the lack of flexibility offered by FPGAs when it comes to implement PUFs, and is hence very platform-specific. Similarly, the cost of the communication system could be greatly reduced as well. The logic resources strictly occupied by the IP protection module are shown in bold in Table II. In our case, we chose a one-time pad for implementing the decryption of the AW on the device. These results show that the overall module is lightweight and would come at low additional cost for a designer.

A. Security analysis

In order to activate and IP core obtained illegally, an attacker would need to encrypt the correct AW with the instance-specific PUF response. After enrolment, the PUF response is usually made inaccessible by blowing a fuse on the device. Even if the attacker owns one unlocked IP core, the AW cannot be obtained without the PUF response. The properties of the CASCADE protocol used to correct the errors of PUF responses are such that recovering the PUF response from the parity informations is hard. Detailed security analysis and countermeasures can be found in [18]. The physical security of the PUF responses depends on the implementation but the responses are usually declared as inaccessible without destroying the device.

B. Possible improvements

We chose to use large multiplexers to direct the PUF indexes to the parity computation module used by the CASCADE protocol to make our implementation as universal as possible. For some cases, if RAM is available, it can reduce the amount of logic resources used. Indeed, the PUF responses and the indexes can be stored in RAM and selected directly.

For communication, we chose to send the largest possible frames at once between the server and the device. This increases the cost in logic resources but reduces the execution time of the CASCADE protocol used to correct the errors in PUF responses. Again, the designer could chose a different trade-off.

VII. Conclusion

In this article, we proposed an architecture that combines a PUF, an interactive error-correction module, a lightweight block cipher and a logic locking/masking module into a complete IP protection scheme. We highlighted the flexibility of this association of solutions and showed where are the trade-offs that the designer could make. The implementation of the overall flow shows that it is easily usable in an industrial context and realistically integrated into the design flow at reasonable cost.

Acknowledgements

The work presented in this paper was realized in the frame of the SALWARE project number ANR-13-JS03-0003 supported by the French "Agence Nationale de la Recherche" and by the French "Fondation de Recherche pour l'Aéronautique et l'Espace", funding for this project was also provided by a grant from "La Région Rhône-Alpes".

References

[1] F. Koushanfar, "Integrated circuits metering for piracy protection and digital rights management: An overview," in *Great Lakes Symposium on VLSI*, Lausanne, Switzerland., May 2011, pp. 449–454.

[2] B. Gassend, D. E. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *ACM Conference on Computer and Communications Security*, Washington, DS, USA, Nov. 2002, pp. 148–160.

[3] Y. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security," in *USENIX Security*, Boston MA, USA, Aug. 2007, pp. 291–306.

[4] J. A. Roy, F. Koushanfar, and I. Markov, "Protecting bus-based hardware IP by secret sharing," in *45th Design Automation Conference*, Anaheim CA, USA, Jun. 2008, pp. 846–851.

[5] ——, "Epic: Ending piracy of integrated circuits," in *Design, Automation and Test in Europe*, Munich, Germany, Mar. 2008, pp. 1069–1074.

[6] J. Huang and J. Lach, "IC activation and user authentication for security-sensitive systems," in *IEEE International Workshop on Hardware-Oriented Security and Trust*, Anaheim CA, USA, Jun. 2008, pp. 76–80.

[7] R. S. Chakraborty and S. Bhunia, "HARPOON: an obfuscation-based soc design methodology for hardware protection," *IEEE Transations on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1493–1502, 2009.

[8] F. Koushanfar, "Provably secure active IC metering techniques for piracy avoidance and digital rights management," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 51–63, 2012.

[9] A. Basak, Y. Zheng, and S. Bhunia, "Active defense against counterfeiting attacks through robust antifuse-based on-chip locks," in *IEEE VLSI Test Symposium*, Napa CA, USA, Apr. 2014, pp. 1–6.

[10] B. Colombier, L. Bossuet, and D. Hély, "Reversible denial-of-service by locking gates insertion for IP cores design protection," in *IEEE Computer Society Annual Symposium on VLSI*, Montpellier, France, Jul. 2015, pp. 210–215.

[11] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 410–424, 2015.

[12] B. Colombier, L. Bossuet, and D. Hély, "Centrality indicators for efficient and scalable logic masking," in *IEEE Computer Society Annual Symposium on VLSI*, Bochum, Germany: IEEE, Jul. 2017, pp. 98–103.

[13] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *IEEE International Symposium on Hardware-Oriented Security and Trust*, Washington, DC, USA, May 2015, pp. 137–143.

[14] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: an ultra-lightweight block cipher," in *International Workshop on Cryptographic Hardware and Embedded Systems*, Vienna, Austria, Sep. 2007, pp. 450–466.

[15] L. Bossuet, X. T. Ngo, Z. Cherif, and V. Fischer, "A PUF based on transient effect ring oscillator and insensitive to locking phenomenon," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 1, pp. 30–36, 2014.

[16] A. Cherkaoui, L. Bossuet, and C. Marchand, "Design, evaluation and optimization of physical unclonable functions based on transient effect ring oscillators," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1291–1305, 2016.

[17] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede, "Helper data algorithms for PUF-based key generation: Overview and analysis," *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 889–902, 2015.

[18] B. Colombier, L. Bossuet, D. Hély, and V. Fischer, "Key reconciliation protocols for error correction of silicon PUF responses," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1988–2002, Aug. 2017.

[19] R. Maes, A. V. Herrewege, and I. Verbauwhede, "PUFKY: A fully functional PUF-based cryptographic key generator," in *International Workshop on Cryptographic Hardware and Embedded Systems*, vol. 7428, Leuven, Belgium, Sep. 2012, pp. 302–319.

[20] D. Ziener and J. Teich, "Power signature watermarking of IP cores for FPGAs," *Journal of Signal Processing Systems*, vol. 51, no. 1, pp. 123–136, 2008.

[21] L. Bossuet, P. Bayon, and V. Fischer, "An ultra-lightweight transmitter for contactless rapid identification of embedded IP in FPGA," *Embedded Systems Letters*, vol. 7, no. 4, pp. 97–100, 2015.