Interlocking Obfuscation for Anti-Tamper Hardware

Avinash R. Desai, Michael S. Hsiao, Chao Wang, Leyla Nazhandali and Simin Hall
Department of Electrical and Computer Engineering
Virginia Tech
Blacksburg, Virginia, 24060
{aviraj,mhsiao}@vt.edu

ABSTRACT

Tampering and Reverse Engineering of a chip to extract the hardware Intellectual Property (IP) core or to inject malicious alterations is a major concern. Digital systems susceptible to tampering are of immense concern to defense organizations. First, offshore chip manufacturing allows the design secrets of the IP cores to be transparent to the foundry and other entities along the production chain. Second, small malicious modifications to the design may not be detectable after fabrication without anti-tamper mechanisms. Some techniques have been developed in the past to improve the defense against such attacks but they tend to fall prey to the increasing power of the attacker. We present a new way to protect against tampering by a clever obfuscation of the design, which can be unlocked with a specific, dynamic path traversal. Hence, the functional mode of the controller is hidden with the help of obfuscated states, and the functional mode is made operational only on the formation of a specific interlocked Code-Word during state transition. No comparator is needed as the obfuscation is embedded within the transition function of the state machine itself. A side benefit is that any small alteration will be magnified via the obfuscated design. In other words, an alteration to the design will manifest itself as a large difference in the circuit's functionality. Experimental results on an Advanced Encryption Standard (AES) circuit from the open-source IP-cores suite suggest that the proposed method provides better active defense mechanisms against attacks with nominal (7.8%) area overhead.

1. INTRODUCTION

The protection of sensitive information in a device is generally considered as a responsibility of the user. This information can be of critical importance for defense organizations, especially if the device falls in the hands of an adversary. The adversary seeks to extract as much sensitive information he/she can have from the device with the help of sophisticated techniques. An adversary may also be interested in learning about an enemy's (or competitor's) latest design by stealing or capturing one or more prototypes/functional devices and dismantling it. To make things worse, in recent years, outsourcing of manufacturing and chip-fabrication requires revealing the design IP to external entities, creating many opportunities for IP infringements, counterfeiting, piracy, and/or insertions of malicious alterations. The problem is exacerbated by contracting the offshore foundries to lower the labor and manufacturing costs. Attacks are thus possible at major entities in the production and supply chains during third party manufacturing. Without proper anti-tamper mechanisms, chips can be reverse-engineered to extract the important IP within the chips. Pirated chips can then be sold at a very low cost. In the same way, insertion of malicious hardware (e.g., Trojans) by the untrusted manufacturer may be easy without anti-tamper features. Once inserted, the Trojan may be extremely difficult to detect, thereby compromising security. Additional threats such as cloning, counterfeiting, reverse-engineering, or re-marking of Integrated Circuits (IC) are possible when there is lack of protection of the design. The estimated U.S. sales losses due to copyright piracy in 2004 is approximately $12.54 billion in total [1] with a significant contribution coming from hardware IPs.

Many techniques have been proposed to protect the circuit at different levels, including both active and passive methods. But with the increase in both strength and sophistication of the techniques used by the adversary, existing methods may not be strong enough. Our goal is to make anti-tamper easy to implement, yet offer a strong protection, of the design.

The proposed method is based on a new hardware obfuscation technique that hides the hardware from the attacker with an interlocked Code-word in the transition function. The methodology is implemented completely in the Register Transfer Level (RTL) design such that a tight bonding between the core logic and the protection circuitry is achieved. Two stages are defined in this method: the entry mode and the functional mode. Both modes are hidden from the adversary with the help of obfuscated states and a dynamic, interlocked Code-Word. The Code-Word is not stored anywhere on chip but is formed dynamically during the entry mode. Code-Word is integrated into the transition logic such that no comparator is used. Existence of any comparator compromises security since the adversary can use the comparator to his/her benefit. Irrespective of the value of the Code-Word, the functional mode is always entered. This differentiates our method from existing methods where an invalid key disallows entry to the functional mode. However, in our design, the behavior in the functional mode

CSIIRW '12 Oct 30 - Nov 01 2012, Oak Ridge, TN, USA

depends on the correct value of the Code-Word. By the manner of our formulation, the core and obfuscation logic are not separable as in existing methods where there is a clear distinction of the modes of operation. This results in a more secure design. Experimental results show that the proposed approach is an effective method against tampering with low area overhead.

2. CLASSIFICATIONS FOR ANTI-TAMPER

Existing defense mechanisms can be classified in two major categories: (a) Passive Techniques and (b) Active Techniques. Passive Techniques such as watermarking are those in which the circuit does not prevent the user from using it in the functional mode. But there are certain characteristics and properties in the circuit which help the user to prove the copyrights of the design and hence can file a case against the adversary for counterfeiting and/or tampering. Active Techniques are those in which the circuit has built-in capability to protect itself against tampering. In this case the circuit has embedded protection hardware so that unauthenticated persons cannot have full access to the circuit. Encryption, hardware metering [6] and obfuscation are placed in this category.

2.1 Reverse Engineering Tools and Methods

Reverse engineering is the process through which one can obtain the details of the circuit for a given IC. An adversary can use a range of techniques to identify the circuit objective and underlying structure using a small set of known characteristics collected through combotronics and/or using an old version of similar kind of chip. Techniques used in reverse engineering can be generally classified as Black Box Testing [8], White Box Testing [8] and Side Channel Analysis (SCA) [5]. In white box analysis, the complete circuit is available to the user/attacker. Creating gate library and focusing on areas dense in XOR is one of the methods proven helpful for reverse engineering for cryptographic algorithms [8].

2.2 Requirements of Effective Anti-Tamper

The adversary has some of the limitations on his side too. Knowing these limitations can help us strengthen the circuit's defense mechanisms. For example, complete functionality of the circuit is unknown to the adversary. The adversary also has limited resources such as time, money, equipment, personnel, etc. As the circuit is large, the adversary will try to attack the protection mechanism with the help of software tools like test pattern generators. We will need to devise techniques that can withstand such attacks.

In addition, obfuscated hardware used for protection should be placed such that it is very hard to differentiate it from the core logic of the function. Furthermore, the anti-tamper mechanisms should not change the specifications of the chip. Finally, the complexity of designing the protection mechanism should be sufficiently low to be feasible.

2.3 Related Work

HARPOON [3] is a method where the emphasis of protection hardware to be kept close to the IP core is made. The circuit is divided into an obfuscated mode and a normal mode. Circuit has to traverse obfuscated mode to reach normal mode. Different induction points are used to drive the circuit to faulty output. The drawback of this method is that it does not offer protection in the normal mode.

Hardware IP Protection during Evaluation using Embedded Sequential Trojan [7] is based on obfuscation of states similar to HARPOON [3]. However, in this work, the implementation of obfuscated states is such that once on missing the sequence of inputs, the core logic goes in a sequence of states and goes on looping in this sequence. After a certain number of incorrect inputs, the IP core goes into extended states which activate certain hardware to force error in the output. The protection mechanism does not act in the functional mode and in the obfuscated mode the circuit loops in the faulty states only.

RTL Hardware IP Protection Using Key-Based Control and Data Flow Obfuscation [4] aims at protecting the hardware by making changes in the data flow and control flow of the original circuit. The circuit reaches the proper functional mode on application of input sequence, i.e., an expected key. If an incorrect sequence is entered, the control and data flow of the circuit are altered with the help of alterations dataflow graph. Thus, the circuit is not stuck at any state but moves in a sequence of states producing an incorrect output. Another advantage is that the circuit is protected even in the functional mode. Although this method provides a great level of security, it fails to provide a dynamic nature to the circuit for different set of inputs provided. In addition, the use of conditions(comparators) weakens the method.

3. METHODOLOGY

Our proposed method is a low-complexity anti-tamper design at the RTL. Similar to previous methods, it implements obfuscated hardware elements for protection. The hardware can be unlocked only with a correct sequence of keys that is given to the appropriate user. However, by looking at the internals of the circuit, it is nearly impossible to separate the anti-tamper logic from the rest of the circuit or to derive the key. The strength of the protection can be increased by the expansion of states, which is done by increasing the number of present state elements in a Finite State Machine (FSM) in core logic. The overall functionality of the circuit is likewise divided into two modes: Entry mode (obfuscated) and Functional mode.

A Code-Word encodes the path executed by the circuit from the inputs applied in the entry mode in order to reach the functional mode. In each stage of the entry mode, Code-Word is gradually formed. Code-Word is used for protection of the circuit in the functional mode, but note that the expected value of the Code-Word is not stored anywhere on chip. Instead, the Code-Word is integrated into the transition functions of the state machine. For example, the transition function in the functional mode is modified and the

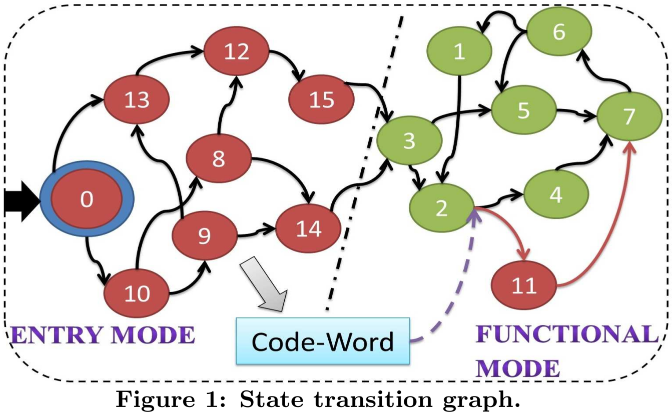

Figure 1: State transition graph.

next state for a number of states is a function of both the present state and Code-Word.

Therefore, the value of the obtained Code-Word plays a key part in determining the transition relation in the functional mode. The two modes of operation are integrated and not easily separable. Figure 1 shows the state transition graph of a modified circuit. The states shown in red are the invalid states used in the entry mode and to force faulty path in functional mode. The valid states in the functional mode are shown in green. Figure 1 also shows the embedding of the Code-Word to compute the correct next state in state “2”.

With such a setup, entry mode starts with a fixed initial state. The correct Code-Word is formed only when the path traversed is as desired (via a correct input sequence). Irrespective of the value of the Code-Word, the functional mode is always entered. This differentiates our method from existing methods where an invalid key disallows entry to the functional mode. However, the behavior in the functional mode depends on the correct value of the Code-Word. The implementation of the Code-Word logic is interlocked with the original transition function and is protected from the adversary by increasing the interaction with the FSM state elements. Black holes and Gray holes [6] can also be formed in the Entry mode to confuse the attacker if the designer wishes to do so. Black holes are a set of states in which the circuit loops once it goes inside. Gray holes are set of states in which there is only one pattern when applied takes you out of the loop of these states.

In the functional mode, the nodes with important computations and assignments are reached only with the correct value obtained in the Code-Word. In other words, the value of the Code-Word is not needed to compute the correct next state for every present state. If the adversary reaches a valid state in the functional mode, he/she is unaware of the next state and also the dependence on the Code-Word. So even if adversary reaches the functional mode via a different input sequence, the next states reached may be different as the value of the Code-Word is different. The functional mode of the circuit adapts a different form with any change in Code-Word. Hence the circuit behaves in a dynamic way according to different values of Code-Word formed.

Figure 2 shows a RTL transformation of the code for the transition graph in Figure 1. The Figure 2a is the original design and Figure 2b shows the modified design. Modified design shows the formation of Code-Word in entry mode marked in blue. Code marked in red shows the usage of Code-Word in state “0010” of functional mode. Note that in this example, a variable “code_word” is used for clarification purposes. In a real design if code has to be given to untrusted entity, one can easily replace the variable name with an obfuscated name.

3.1 Design Flow

The complete implementation is at the RTL level and can be modeled using any Hardware Description language. For demonstration of the idea, the proposed method was implemented using designs from OpenCores [2].

Once the cores are obtained, the first step is to identify an FSM which is the heart and controls most of the assignments and computations. The states of this FSM are expanded and classified into extended states and functional states. State encoding for the functional states is kept as required for the design. Entry mode state transition graph is designed such that it contains loops of states and the correct path is difficult to figure out. Functional mode is modified and the value of the Code-Word to be formed is calculated as per the function and the state encoding. Size of the Code-Word depends on the area overhead allowed by the designer/user. Longer Code-Words imply that the circuit can sustain higher levels of brute force attack. The computations and assignments in the entry mode graph are chosen such that the Code-Word changes for any change in the input sequence and also state transition. Finally, a set of inputs is then obtained to form the key to make the circuit functional.

(a) Original Design

```
case STATE  is
        when "011" => if(in1 = ‘1’) then STATE <= "101";
                else STATE <= "010"; end if;
        when "010" =>           STATE <= "100";
        when "101" =>           STATE <= "111";
        when "111" =>      STATE <= "110";
        when "110" => if(in1 = ‘1’) then STATE <= "001";
                else STATE <= "101"; end if;
        when "001" =>           STATE <= "010";
        when "100" =>           STATE <= "111";
        when others =>           null;
end case;
```

(b) Modified Design

```
case STATE  is
  when "0000" => if(in1 = ‘1’) then STATE <= "1101";
      code_word <="0011";
    else STATE <= "1010"; code_word <="0110"; end if;
  when "1010" => if(in1 = ‘1’) then STATE <= "1001";
      code_word <= STATE or code_word;
    else STATE<="1000"; code_word<=STATE and code_word; end if;
  when "1000" => if(in1 = ‘1’) then STATE <= "1100";
      code_word <= STATE(3) & code_word(2 downto 0);
    else STATE <= "1110"; code_word <= not code_word;end if;
  when "1001" => if(in1 = ‘1’) then STATE <= "1110";
      code_word <= STATE(3 downto 1) & in1;
    else STATE <= "1101"; end if;
  when "1110" => STATE <= "0011";
      code_word <= in1 & code_word(3downto1);
  when "1101" => STATE <= "1100";
      code_word <= code_word(1 downto 0) & STATE (3 downto 2);
  when "1100" => STATE <= "1111";
      code_word<= code_word(3)&(not in1) & code_word(1downto0);
  when "1111" => STATE <= "0011";
      code_word <= (not in1) & code_word(2 downto 0);
  when "1011" => STATE <= "0111";
  when "0011" => if(in1 = ‘1’) then STATE <= "0101";
    else STATE <= "0010"; end if;
  when "0010" =>        STATE <= STATE xor code_word;
  when "0101" =>        STATE <= "0111";
  when "0111" =>  STATE <= "0110";
  when "0110" =>        if(in1 = ‘1’) then STATE <= "0001";
    else STATE <= "0101"; end if;
  when "0001" =>        STATE <= "0010";
  when "0100" =>        STATE <= "0111";
  when others =>        null;
end case;
```

Figure 2: Example obfuscated RTL design.

4. IMPLEMENTATION AND RESULTS

For implementation purposes, an open source core from Open Cores [2] was taken. Circuit chosen for test was an AES design. The following changes were made in the design:

(1) Only the core FSM of the AES design was padded up with three more bits thus raising it from 5 to 8. Now the number of states available in the extended state is 256-32 = 224.

(2) Code-Word was chosen to be of size 48 bits. For the state transition in the entry mode, 64 inputs (out of 130 inputs present in the design) were used to reduce the hardware overhead. Design was implemented in Virtex 7, and simulations were performed using Xilinx ISE.

Implementation of the Code-Word based control in the functional mode helps to achieve dynamic nature of the circuit to various incorrect input sequences. It is very difficult to distinguish between the Code-Word, the FSM state elements and other state elements in the whole circuit. Due to implementation in the RTL level, the components protecting circuit can be said to be highly obfuscated.

One of the worst possible scenarios is when the adversary has a previous working model of a chip. The designer has launched a new model which has improved performance metric. In this case the adversary knows circuit completely. The adversary can perform known answer tests to verify whether he/she has unlocked the IC or not.

Consider that the adversary has obtained information of other state elements in circuit except the main FSM and the Code-Word as we have not altered the other parts of the circuit. The best way to attack the chip is to separate the Code-Word (48 state elements) and core FSM (8 State Elements). The adversary knows the assignments made in the states of the original circuit. So, it would be best for him to find the main state elements and then the valid states accordingly. Since there are 56 state elements (8 + 48) to be searched for the number of combinations to find state elements is

$$\sum_{k=1}^{56}\binom{56}{k}$$

where, ‘k’ is all possible sizes of Code-Word

Next, he/she will try to find a state in which the assignments performed are similar to the original circuit assignments. The adversary has to identify the obfuscated 32 functional states.

No. of states in the old circuit(state elements 5) = n = 32

Number of computations required for comparison for predicting states with similar assignments= n. Thus the total number of iterations involved is given by

$$\left(\sum_{k=1}^{56}\binom{56}{k}\right)*(n)=2.30e18$$

To find the next state, the best possible way is to find the next state in the original circuit and map it to the state in new circuit. These comparisons have been done in an earlier step so there is no increase in computations. In calculating this case, we have considered all the best possible scenarios in terms of the attacker. But still the number of computations required is 2.3e18, which is enormous.

In practice, the number of computations required would be larger than this, especially if the adversary does not have any other circuit to compare with. At each stage of the combination he/she will have to guess the value of the Code-word which requires $2^{(56-k)}$ number of iterations. In that case, the total number of iterations will be

$$\sum_{k=1}^{56}\left(\binom{56}{k}*2^{(56-k)}\right)=5.233e26$$

Table I shows the hardware overhead in the implementation of these protection mechanisms. For the AES design, the area overhead of our method is 7.8%, compared to a recent technique [4] that needed 8.6%. The strength of our method provides a higher level of protection, in which 5.233e26 combinations is needed, while an average of 7.13e15 as needed in [4] using similar calculations. Average values are considered here as the implemented designs are different.

Table 1: RESULTS

Design	Area Overhead		Number of Combinations Required		
	Present Method	Average Overhead in [4]	Present Method		Average in [4]
			(min)	(max)	
AES	7.8%	8.6%	2.3e18	5.233e26	7.13e15

5. CONCLUSION AND FUTURE WORK

A new interlocking obfuscation technique is proposed for active defense mechanisms of the circuit against tampering. The area overhead for implemented design is less than 8%. The circuit response to incorrect sequences is different due to the dynamic Code-Word formed. As the Code-Word is embedded within the transition function itself, it creates a new dynamic nature to the circuit behavior making reverse engineering more difficult. Our method suggests bypassing of the important states with extended states to add up to the confusion of the adversary. The specifications of the chip are not altered and is identical to original specifications once user is authenticated. Inputs and outputs are kept the same. As a side benefit any minor change in the circuit by third party gets magnified due to the obfuscated design.

Static algorithms for detecting the correct path to functional mode may yield certain results but the method ensures that only one path in the entry mode when traversed helps unlock the circuit. Higher levels of protection can be achieved if one allows for a larger area overhead. Thus, there is a tradeoff between area overhead and level of protection achieved. The proposed method can be implemented directly at the RTL level; thus, any HDL can model this type of circuit making it language and platform independent.

In the future, other methods can be used to decrease the area overhead. The use of PUFs can provide high dynamic nature to circuit output, hence a higher level of obfuscation.

6. REFERENCES

[1] Foreign infringement of intellectuaal property rights implications on selected U.S. industries. http://www.usitc.gov/publications/332/working_papers/id_14_100505.pdf.

[2] Open Cores. http://www.opencores.org.

[3] Chakraborty, R., and Bhunia, S. HARPOON: An Obfuscation-Based SoC Design Methodology for Hardware Protection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2009*.

[4] Chakraborty, R., and Bhunia, S. RTL Hardware IP Protection Using Key-Based Control and Data Flow Obfuscation. In *23rd International Conference on VLSI Design, 2010. VLSID '10.*, pp. 405 –410.

[5] Fan, J., Guo, X., De Mulder, E., Schaumont, P., Preneel, B., and Verbauwhede, I. State-of-the-art of secure ECC implementations: a survey on known side-channel attacks and countermeasures. In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST),2010.*

[6] Koushanfar, F. Provably Secure Active IC Metering Techniques for Piracy Avoidance and Digital Rights Management. *IEEE Transactions on Information Forensics and Security,* (feb. 2012).

[7] Narasimhan, S. and Chakraborty, R. and Bhunia, S. Hardware IP Protection During Evaluation Using Embedded Sequential Trojan. *IEEE Design Test of Computers,* (2011).

[8] Porter, R., Stone, S., Kim, Y., McDonald, J., and Starman, L. Dynamic Polymorphic Reconfiguration for anti-tamper circuits. In *International Conference onField Programmable Logic and Applications, 2009. FPL 2009.* (2009).