Chapter 4
Gate Camouflaging-Based Obfuscation

Xueyan Wang, Mingze Gao, Qiang Zhou, Yici Cai and Gang Qu

4.1 Circuit Camouflaging with Configurable Gates

One of the greatest threats to VLSI design intellectual property (IP) is reverse engineering [1]. ***Reverse engineering*** (RE) is the process of extracting the IP and design information from the target product and reproducing the product [2, 3]. Motivations of RE vary from the paranoia of the Cold War, through commercial piracy, to competitive intelligence, and courts of patent law. The targets of RE include systems as large as an aircraft or as small as a microchip, programming codes, a pill of medical drug, or any sort of IPs [2]. In the semiconductor sector, RE has become a powerful tool for IP piracy where the attacker analyzes a design and reproduces it with no or much less investment in research and development. These low cost illegitimate products bring security vulnerabilities to critical commercial and military systems, or they can be sold at a much lower price, giving them an unfair competitive edge against the authenticated products.

The popular digital circuit watermarking and fingerprinting techniques [1] are passive IP protection schemes because they do not prevent RE from happening or make it more difficult. Watermark and fingerprint can be embedded into the IP to make each instance of the IP unique. When necessary, they can be revealed to show the authorship or ownership of the IP and identify the parties that misuse the IP. Although it is hard or impossible to completely remove the watermark and fingerprint, RE attackers can still extract valuable information from the IP and reproduce the IP sillegally. The existence of watermark and fingerprint in the IP can deter RE attacks, but will not increase the complexity of RE.

X. Wang · Q. Zhou · Y. Cai
Tsinghua University, Beijing, People's Republic of China

M. Gao · G. Qu (✉)
University of Maryland, College Park, MD, USA
e-mail: gangqu@umd.edu

D. Forte et al. (eds.), *Hardware Protection through Obfuscation*,
DOI 10.1007/978-3-319-49019-9_4

Table 4.1 The configurable CMOS cell in Fig. 4.1 can perform three different functions: NAND, NOR, or XOR, based on different true and dummy contact combinations [7]

Function	Contacts	
	TRUE	DUMMY
NAND	2, 4, 6, 8, 11, 12, 16, 17	1, 3, 5, 7, 9, 10, 13, 14, 15, 18, 19
NOR	2, 5, 6, 11, 12, 18, 19	1, 3, 4, 7, 8, 9, 10, 13, 14, 15, 16, 17
XOR	1, 3, 4, 7, 9, 10, 12, 13, 14, 15, 18, 19	2, 5, 6, 8, 11, 16, 17

Gate camouflaging techniques have emerged as an effective countermeasure for RE attacks [4–7]. These techniques rely on the general belief that RE technology is normally 2–3 generations behind the latest CMOS design technology. That is, certain CMOS design features cannot be completely reverse engineered until several years later. For example, some logic cells can be configured to perform different functionalities while maintaining an identical look to RE attackers. In circuit camouflaging, conventional logic gates are intentionally replaced by these configurable CMOS cells to thwart RE attacks.

4.1.1 Configurable CMOS Cells

One popular approach to construct configurable CMOS cells is to use the true and dummy contacts [4, 5]. A true contact spans the dielectric between two adjacent layers and represents an electrical connection, while a dummy contact has a gap in the middle thus fakes the connection between layers. Figure 4.1 shows an example where 19 contacts are utilized in the configurable CMOS cell [7]. As demonstrated in Table 4.1, with different combinations of true and dummy contacts, the camouflaging gate can perform three different logic functions: NAND, NOR, or XOR. For instance, when contacts 2, 4, 6, 8, 11, 12, 16, 17 are true and contacts 1, 3, 5, 7, 9, 10, 13, 14, 15, 18, 19 are dummy, the camouflaged CMOS cell performs the functionality of a NAND gate.

When an attacker performs the top-down image processing-based RE attack, he is unable to detect whether a contact is true or dummy, because from the top view of the chip, the true and dummy contacts appear identical even under most powerful optical and electrical microscopes. Therefore, this configurable CMOS cell will appear the same look to the attacker regardless of the functionality it implements. Without knowing the functionalities of the camouflaged gates, the attacker will fail to reverse engineering the IP. Of course, the attacker can guess and try all the different possible configurations, which will increase the complexity of the RE attack. This also implies that against such brute force attack, the more camouflaged gates we have in the IP and the more functionalities a camouflaged gate can achieve, the more difficult it will be for attackers to recover the IP.

4.1.2 Circuit Camouflaging Technique

The circuit camouflaging technique proposed in [7] obfuscates a circuit by disguising the functionality of selective XOR, NAND, or NOR gate behind the configurable CMOS cell illustrated in Fig. 4.1. That is, each camouflaged gate will have the same appearance but three possible functionalities to an RE attacker. In the circuit shown in Fig. 4.2, two logic gates have been replaced by the configurable CMOS cells C1 and C2. Even when an RE attacker has successfully recovered the layout of the circuit, he cannot rebuild the circuit unless he knows the types of the two camouflaged cells C1 and C2.

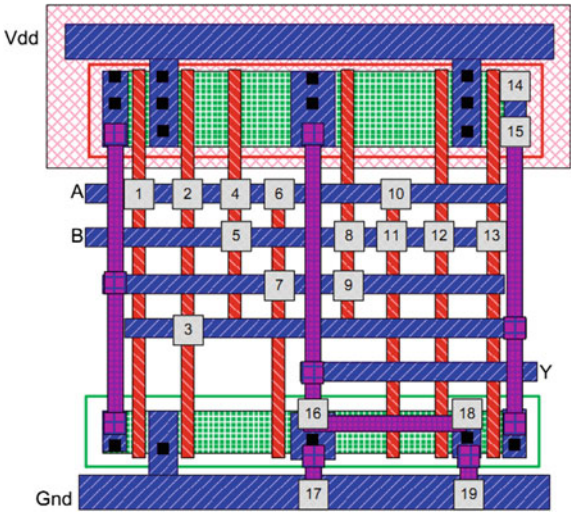

Fig. 4.1 A configurable CMOS cell with 19 contacts that can be configured to be either true or dummy [7]

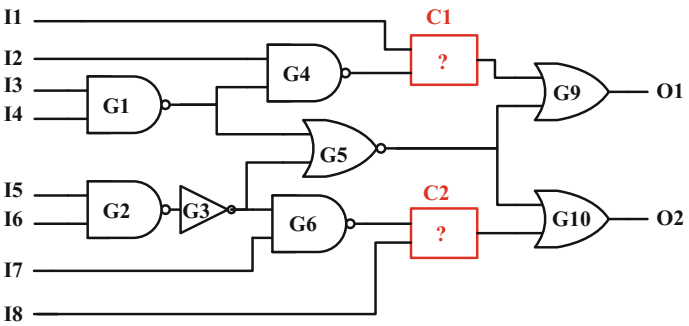

Fig. 4.2 A circuit with two camouflaged gates C1 and C2, which can be resolved individually by VLSI testing-based attack

Intuitively, the attacker has to guess $3^2 = 9$ possibly combinations, because each of the two camouflaged cells can be an XOR, NAND, or NOR gate. Moreover, since the attacker can only give input values to the circuit's primary inputs (PI) and observe the corresponding primary output (PO) values to verify whether a guess is correct or not. This seems to make attacker's job very challenging. Unfortunately, this is not the case.

In the above example, based on the fact that (i) the output of a camouflaged gate under input '00' can differentiate {XOR} from {NAND, NOR} (XOR outputs 0, while both NAND and NOR output 1), and (ii) the output under input '01' or '10' can differentiate {NAND} and {NOR} (NAND outputs 1, while NOR outputs 0), the attacker can apply the input pattern '010XXXXX' (X represents do not care values) at the PIs, this justifies the camouflaged gate C1's inputs as '00', and sensitize C1's output to PO O1. If O1 is 0, the functionality of C1 is resolved to be XOR. Otherwise, when O1 is 1, C1 will be either NAND or NOR. The attacker will then apply input pattern '110XXXXX' at PIs to justify C1's inputs as '10' and sensitize C1's output to O1. If O1 is 0, C1 is resolved to be NOR, otherwise C1 is resolved to be NAND.

This is known as ***VLSI testing-based attack***, where the attacker resolves a camouflaged gate's functionality by justifying the gate's inputs to certain values from the circuit's PIs then sensitizing the gate's corresponding output to PO to observe from a functional IC. In such attack, it is not necessary to obtain the entire truth table to resolve a camouflaged gate, a couple of selective input–output pairs will be sufficient. Thus, it is a very effective way to attack circuit camouflaging.

4.1.3 Enhanced Circuit Camouflaging

In the above example, we see that randomly selecting gates to camouflage is vulnerable to the VLSI testing-based attack. This can be fixed by judiciously selecting candidate gates to camouflage such that these gates will be interfered and cannot be revealed one by one [7].

Two camouflaged gates are ***interfered*** if one gate lies on a path between the other gate and an output, or the outputs of these two gates go into the same gate. Clearly, in Fig. 4.3, the three camouflaged gates C1, C2, and C3 are interfered. If gate G4 is also camouflaged, it will not interfere with C3, but it will interfere with C2 because their outputs meet at gate G6.

On the other hand, when camouflaged gates do not have any path in the circuit interfering with other camouflaged gates, they are called ***isolated***. Isolated camouflaged gates can be resolved independently by VLSI testing-based attack as we have seen in Fig. 4.2. However, this will not be the case when camouflaged gates are interfered.

For example, in Fig. 4.3, none of the camouflaged gates C1, C2, and C3 can be resolved by VLSI testing-based attack individually: C1's output cannot be observed from any of the POs without resolving C2 and C3 first; C3's inputs cannot be controlled before C1 and C2 are resolved; both the controllability of C2's inputs and the observability of its output rely on the functionalities of C1 and C3.

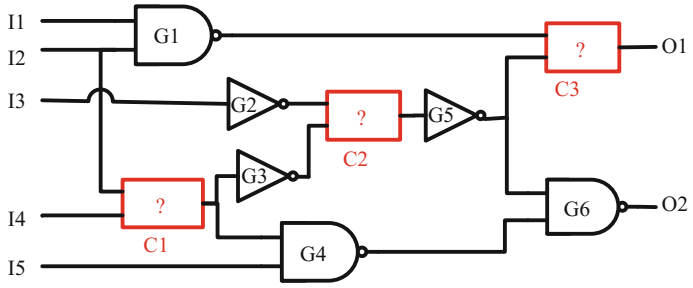

Fig. 4.3 A camouflaged circuit where the three camouflaged gates C1, C2, and C3 are interfered

In [7], it is argued that this will force the attackers to brute force search all the possible functionality combinations of these camouflaged gates. Specifically, for each possible combination, the attacker will simulate input patterns at PIs of the circuit to get the corresponding outputs at POs and compare them with an unpackaged/functional circuit. If they are not the same, the guess is incorrect and the attacker will check the next possible combination. Considering that each camouflaged gate has 3 possible functionalities, the needed brute force efforts will be 3^3. When the circuit has N selective camouflaged gates, the complexity will be 3^N. This exponential complexity leads to the claim that when there are sufficient number of interfered camouflaged gates, the circuit camouflaging is secure [7]. To end this section, we demonstrate by an example that this claim is not accurate, which motivates us to propose more accurate metrics to define the security of circuit camouflaging.

4.1.4 Defeating the Enhanced Circuit Camouflaging

Figure 4.4 is a sub-circuit of the secure camouflaged circuit in Fig. 4.3, where the two camouflaged gates C1 and C2 are clearly interfered with each other. We now show how both camouflaged gates can be revealed with no more than four input–output pairs.

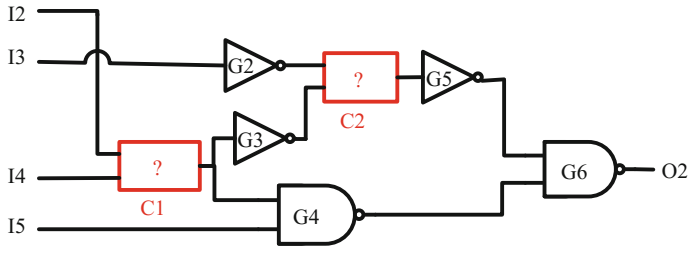

Fig. 4.4 A sub-circuit of the secure camouflaged circuit in Fig. 4.3

We consider the output O2 as a function of inputs I2, I3, I4, and I5 and denote it by O2(I2, I3, I4, I5). That is, when input values are I2 = 1, I3 = 1, I4 = 0, and I5 = 0, for example, the output value can be written as O2(1,1,0,0). Here is how an attack can reveal C1 and C2 easily:

(1) Apply (0, 1, 0, 1) as the input values for (I2, I3, I4, I5);
(2) If O2(0, 1, 0, 1) = 0 : apply (0, 1, 1, 0) as the input;
(3) if O2(0, 1, 1, 0) = 1 : apply (0, 0, 1, 0) as the input;
(4) If O2(0, 1, 0, 1) = 1 : apply (0, 1, 1, 1) as the input;
(5) apply (0, 1, 0, 0) as the input;
(6) if O2(0, 1, 0, 0) = 1 : apply (0, 0, 0, 0) as the input;

In step (1), we apply (0,1,0,1) as input to the circuit. If C1 is either NAND or NOR, C1(0,0) = 1; so G4(1,1) = 0 and we should be able to observe O2 = 1 regardless of the output of G5. Therefore, we conclude that if O2(0,1,0,1) =0, C1 must be XOR. Next, in order to reveal C2, we apply (0,1,1,0) in step (2). C1(1,1) = XOR (1,1) = 0, hence C2 will have both its input as 1 and it will output 0 only if C2 is XOR; otherwise, we need another input pattern to determine whether C2 is NOR or NAND, this can be done by for example using (0,0,1,0) as in step (3). Similar analysis can be done for steps (4)–(6) when C1 is either an NOR or an NAND.

Once we fully resolve the two camouflaged gates C1 and C2 in Fig. 4.4 (by applying no more than four input patterns), it will be trivial to resolve the last camouflaged gate C3 in Fig. 4.3 (with no more than two input patterns). Note that this takes us no more than six input–output pairs, much less than trying 3^3 possible combinations with multiple input–output pairs for each combination. We are able to do this because that the circuit can be partitioned to smaller sub-circuits and the set of {XOR, NAND, NOR} can be effectively distinguished. Next, we will elaborate an attack against circuit camouflaging based on circuit partitioning and then discuss several countermeasures to this attack.

4.2 Circuit Partition-Based Attack

It is believed that the enhanced IC camouflaging is secure because of the high brute force complexity that is exponential to the number of camouflaged gates [7]. However, this is just secure against the naïve brute force search. Like many other studies in security literature, a new type of attack would break a system that was previously proven secure.

As we have seen from the last example, an intelligent attacker does not need to resolve all the camouflaged gates together even though they are interfered. Instead, he may first partition the circuit to sub-circuits whose functions can be tested individu-

ally from a functional IC, and then perform a brute force search for the functionalities of the camouflaged gates in each sub-circuit individually. He can of course use some other smarter approaches that leverage the input–output difference of the potential gate types of the camouflaged gates.

The key idea of the circuit partition-based attack is to leverage the divide and conquer methodology to partition camouflaged gates into multiple sub-circuits, then target each sub-circuit individually. The benefit of circuit partition is breaking down the original large interference circles of camouflaged gates to multiple small interference circles in order to reduce the brute force complexity. Before elaborating circuit partition-based attack, we first give the definition:

Definition 1 The Maximum FanIn-Cone rooted at a primary output Z is defined as $MFIC_Z = \{G_i \mid \text{gate } G_i \text{ belongs to the circuit and there exists a path } G_i \rightarrow Z\}$, which is the set of all the gates whose outputs will directly or indirectly feed into the gate that generate output Z.

The word maximum in $MFIC_Z$ indicates that all the gates that will impact the value of output Z should be included. In another word, if a gate does not belong to $MFIC_Z$ for some output Z, then we will not be able to observe from Z any changes on that gate. Notice that in this chapter, unless it is specified otherwise, we will use $MFIC_Z$ for both the maximum fanin-cone rooted at the primary output Z and the corresponding sub-circuit that includes all these logic gates. For example, in the circuit shown in Fig. 4.3, we have $MFIC_{O1} = \{C1, C2, C3, G1, G2, G3, G5\}$, and $MFIC_{O2} = \{C1, C2, G2, G3, G4, G5, G6\}$, where the latter is the sub-circuit shown in Fig. 4.4.

Accordingly, MFIC's function, as a sub-circuit, can be studied by directly feeding the PIs of the MFIC and observing the corresponding output. A camouflaged gate, like other logic gates, may belong to multiple MFICs. For instance, five logic gates, including C1 and C2, belong to both $MFIC_{O1}$ and $MFIC_{O2}$ in the above example. Thus, when an attacker applies brute force attack to resolve the camouflaged gates, he can start with the MFIC that has the fewest number of camouflaged gates. In the above example, after he resolves C1 and C2 from $MFIC_{O2}$, $MFIC_{O1}$ will have only one camouflaged gate C3 left to solve. This greedy approach is the basic idea behind the following smart circuit partition-based attack shown in Algorithm 1.

In line 1, we partition the circuit into MFICs, which can be done with standard algorithm. In the rest of the algorithm, we iteratively resolve the camouflaged gates in one MFIC at a time. We greedily choose MFIC with the minimum number of unresolved camouflaged gates to apply brute force attack (lines 2–3) (for example in Fig. 4.3, there are three camouflaged gates in $MFIC_{O1}$ and two camouflaged gates in $MFIC_{O2}$, so we will start with $MFIC_{O2}$). This selection ensures that brute force efforts in current iteration can be minimized (line 7). The obfuscated netlist will then be updated by replacing the resolved camouflaged gates with the corresponding logic gates they implement (line 8). When there are multiple eligible MFICs with the same minimum number of camouflaged gates, the algorithm will choose the one that minimizes the maximum number of camouflaged gate number in the remaining MFICs (lines 4–6). The while loop in lines 9–12 checks whether there exist relevant

unresolvable camouflaged gates that will become resolvable when the obfuscated netlist is updated (in Fig. 4.3, C3 will become resolvable after C1 and C2 are resolved in $MFIC_{O2}$). If there is any, we resolve it by the VLSI testing principles-based attack [7].

ALGORITHM 1. Smart Circuit Partition based Attack

Input: Camouflaged Netlist, Functional IC.
Output: Original Netlist.
1: partition the circuit to MFICs;
2: **while** *there exist unresolved camouflaged gates in the netlist*
3: find MFIC(s) with minimum unresolvable camouflaged gates;
4: **while** *there is more than one MFIC eligible*
5: select the one minimizes next maximum camouflaged gates number;
6: **end**
7: brute force search possible functionality combinations;
8: update netlist;
9: **while** *there are unresolvable camouflaged gates become resolvable*
10: resolve them;
11: update netlist;
12: **end**
13: **end**
14: **return** the resolved netlist.

Suppose that there are N camouflaged gates in the netlist and each camouflaged gate can implement any one of the three logic gates {XOR, NAND, or NOR}. When an RE attacker does not have any more information, there will be 3^N possible functionality combinations to enumerate. However, the circuit partition-based attack algorithm in Algorithm.1 clearly shows that the attacker can resolve all the camouflaged gates much more efficiently. Let n_i be the number of camouflaged gates in the MFIC we selected during the i-th iteration (line 3), the brute force search process will search 3^{n_i} cases in the worst case. Let r_i be the number of camouflaged gates that become resolvable (lines 9–12) during the same iteration after the i-th MFIC is resolved, we have $\Sigma_{i=1,2,\ldots}(n_i + r_i) = N$. Since it takes only two PI patterns to resolve each of the r_i camouflaged gates in line 10 [7], it becomes clear that the complexity of the algorithm in Algorithm 1 is dominated by $3^{n_{max}}$, where $n_{max} = \max\{n_i\}$, the largest number of camouflaged gates in the same MFIC that need to be resolved simultaneously. In real design, n_{max} normally is much smaller than N and does not increase with N. Our experiments on benchmark circuits indicate that n_{max} is usually small (less than 10) [8].

Theorem 1 *The security of a camouflaged circuit against the circuit partition-based attack is determined by n_{max}, the largest number of camouflaged gates in the same MFIC that need to be resolved simultaneously.*

4.3 Mitigating the Circuit Partition-Based Attack

From Theorem 1, we see that it is crucial to have a large n_{max}, which means that we want to keep the camouflaged gates together such that the attacker cannot partition them into multiple sub-circuits and resolve separately. A gate classification method can help us to select the to-be camouflaged gates for this purpose.

Definition 2 For a gate G, $MFICS_G$ is the set of $MFIC_{PO}$ that G belongs to. Formally, $MFICS_G = \{MFIC_{PO_i} \mid PO_i$ is a primary output and $G \in MFIC_{PO_i}\}$.

To compute $MFICS_G$, we can first compute $MFIC_{PO}$ for all the primary outputs, then construct each of the $MFICS_G$ by examining which $MFIC_{PO}$ gate G belongs to. For example, for the circuit in Fig. 4.5, we have

$MFIC_{O1} = \{G1, G2, G3, G5\}$,
$MFIC_{O2} = \{G2, G3, G4, G6\}$,
thus
$MFICS_{G1} = MFICS_{G5} = \{MFIC_{O1}\}$,
$MFICS_{G2} = MFICS_{G3} = \{MFIC_{O1}, MFIC_{O2}\}$,
$MFICS_{G4} = MFICS_{G6} = \{MFIC_{O2}\}$.

Theorem 2 *MFICS is an equivalent relation and thus we can partition the circuit by putting the gates with the same MFICS into the same equivalent class. That is, gates G1, G2, ..., Gn are partitioned to the same class if and only if* $MFICS_{G1} = MFICS_{G2} = \cdots = MFICS_{Gn}$.

In the above example, the circuit will be partitioned into three equivalent classes: {G1, G5}, {G2, G3}, and {G4, G6}. Notice that gates in the same equivalent class cannot be partitioned further. This is an important feature for the following practical gate selection method that mitigates circuit partition-based attack.

As a reverse engineering attacker can only assign values to the PIs and observe the corresponding POs from an unpackaged functional IC, $MFIC_{PO}$ will be the minimum sub-circuit whose function can be tested by the attacker. Therefore, if we select gates to obfuscate from the same equivalent class, for any $MFIC_{PO_i}$ of the circuit that the attacker can attack, either all of the camouflaged gates belong to it, or none of them belongs to it. Thus the attacker will not be able to partition the camouflaged gates

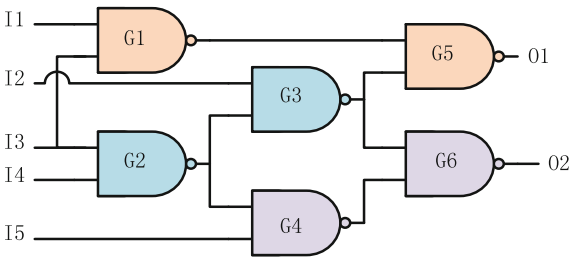

Fig. 4.5 A circuit example for gate classification. Gates in the same class, {G1, G5}, {G2, G3}, and {G4, G6}, are marked with the same color

into multiple sub-circuits to perform attacks individually. This criterion should be added to the enhanced circuit camouflaging method in [7] for better security.

4.4 Multiplexer-Based Circuit Obfuscation

Recall that the more functionalities a camouflaged gate can achieve, the more difficult it will be for attackers to resolve the camouflaged gate. As we have demonstrated above, when the configurable CMOS cell can only implement the functionality of XOR, NAND, or NOR gate, there will be two major security concerns. First, it restricts the selection of the gates to be camouflaged to only these three types of gates, limiting the value of n_{max}. Second, such camouflaged gate can be easily resolved by applying two distinct input patterns (as shown in the examples above). The multiplexer-based circuit obfuscation method [9, 10] solves this problem.

A 4×1 multiplexer (MUX) has four data lines {X1, X2, X3, X4}, two selection bits {A, B} and one output line S that comes from one of the data lines determined by the value of the selection bits A and B. Specifically, the output can be expressed as

$$S = A'B'X1 + A'BX2 + AB'X3 + ABX4$$

By assigning proper values to the data lines, a 4×1 MUX can implement any 2-input logic function (see Table 4.2). For example, when X1 = 0, X2 = 1, X3 = 1, and X4 = 0, the MUX becomes XOR.

In multiplexer-based circuit obfuscation, special designed MUX is utilized as the configurable logic unit to replace conventional gates. Programmable camouflage connector is used to configure the functionality of configurable logic unit. Similar to the contacts used in [7], the programmable camouflage connector can be programmed to be either a connection or an isolation, while appears to be physically identical under optical or electron microscopy.

More specifically, as shown in Fig. 4.6, the selection lines A and B and output line S of the multiplexer act as the inputs and output of the configurable logic unit, respectively. Each input line X_i (i = 1,2,3,4) is connected to Vdd and Vss by two

Table 4.2 A 4 × 1 MUX can implement all the 16 2-input Boolean functions

Function Number	X1	X2	X3	X4	Logic expression of S
1	0	0	0	1	AB
2	0	0	1	0	$A\overline{B}$
3	0	0	1	1	A + 0· B=A
4	0	1	0	0	$\bar{A}B$
5	0	1	0	1	0·A + B=B
6	0	1	1	0	A ⊕ B
7	0	1	1	1	A+B
8	1	0	0	0	$\bar{A} \cdot \bar{B} = \overline{A + B}$
9	1	0	0	1	A ⊙ B
10	1	0	1	0	$\bar{B}$
11	1	0	1	1	$A + \overline{B}$
12	1	1	0	0	$\bar{A}$
13	1	1	0	1	$\bar{A} + B$
14	1	1	1	0	$\overline{AB}$
15	1	1	1	1	const 1
16	0	0	0	0	const 0

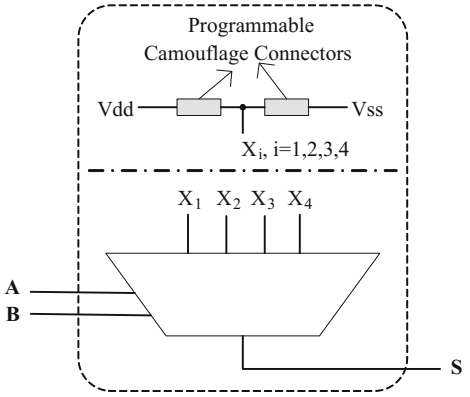

Fig. 4.6 Each input line of the MUX is connected to two programmable camouflage connectors, one is configured to be a connection and the other to be an isolation

camouflage connectors, but only one is programmed to be a connection, the other one is programmed to be an isolation. $X_i = 1$ when the camouflage connector that connected to Vss is configured as a connection, and $X_i = 0$ when the camouflage connector that connected to Vdd is configured as a connection.

Algorithm 2 shows a heuristic algorithm to perform circuit obfuscation with such MUXs. As analyzed in the previous section, we select gates from the same equivalent class so they cannot be resolved individually. The algorithm makes the gate that generates the PO, called outGates, of the MFICS of an equivalence class to appear as black boxes by obfuscating the outGates with MUXs, and blocking at least one input of the MUX (lines 4–9). Then desired number of gates are iteratively selected from the class to obfuscate, following the principle of minimizing performance overhead (lines 10–18). When there are more than one eligible gate class, the algorithm will get different versions of obfuscated circuits (line 19) and will return the one that results in minimum performance overhead (line 21).

ALGORITHM 2. Circuit Obfuscation with Multiplexers

Input: Original circuit *G*, to be obfuscated gate number *N*.
Output: Obfuscated circuit *G′*.
1: do gate classification by the *MFICS* gates belong to;
2: **for** *each class C with no less than M gates*
3: *currentG* ← *G*
4: **for** *each* $MFIC_{PO} \in$ *class C's MFICS*
5: obfuscate the *outGate* of the $MFIC_{PO}$;
6: **if** *none of the MUX's inputs is blocked*
7: obfuscate the MUX's one input gate whose *MFICS* is a subset of *C*'s *MFICS*;
8: **end**
9: **end**
10: **while** *obfuscatedGates* < *N*
11: select $gate_i$ with smallest *maxPathDelay* from unobfuscated gates in *C*;
12: **if** *there is an inverter* $gate_{INV}$ *before any input of* $gate_i$ *and* $gate_{INV} \in C$
13: merge $gate_{INV}$ and $gate_i$ as a logic block;
14: obfuscate the logic block with one MUX;
15: **else**
16: obfuscate $gate_i$;
17: **end**
18: **end**
19: save current obfuscated circuit *currentG*;
20: **end**
21: **return** obfuscated circuit with smallest performance overhead as *G′*.

4.5 Conclusions

In this chapter, we study the popular gate camouflaging-based obfuscation with focus on its security analysis. We use the circuit partition-based attack to demonstrate that the selection of camouflaged gates is crucial to increase the attack complexity of reverse engineering. We show that mitigation methods such as smart gate selection and the use of multiplexer can help to secure gate camouflaging against reverse engineering. It is our belief that both 'spear' and 'shield' need to be developed in the war against attackers.

To make this promising circuit camouflaging technique practical in thwarting reverse engineering attacks, there still exist many challenges. Perhaps the most significant one is that the overhead in applying CMOS camouflaging gates can be rather high in terms of circuit timing, power consumption, and area, especially when a high level of protection is needed. How to reduce the overhead incurred by circuit camouflaging would continue to be an urgent need. The second challenge is the development of countermeasures against the newly proposed and powerful de-obfuscation attacks based on SAT solver. Such attacks can effectively exclude incorrect functionality combinations of the camouflaged gates, successfully bypassing the exponential complexity of brute force. Although it cannot ensure to be effective in all circumstances, it has already posed a serious threat to many circuit camouflaging scenarios. Finally, it will be interesting to study intrinsic reconfigurable properties of emerging devices and how they can be utilized for circuit camouflaging.

Acknowledgements Mingze Gao and Gang Qu were supported in part by AFOSR MURI under award number FA9550-14-1-0351.

References

1. Qu G, Potkonjak M (2003) Intellectual property protection in VLSI designs: theory and practice. Kluwer Academic Publishers, Dordrecht. ISBN 1-4020-7320-8
2. Torrance R, James D (2011) The state-of-the-art in semiconductor reverse engineering. In: Proceedings of the ACM/IEEE design automation conference (DAC), pp 333–338
3. Quadir SE, Chen J, Forte D et al (2016) A survey on chip to system reverse engineering[J]. ACM J Emerg Technol Comput Syst (JETC) 13(1):6
4. Chow L, Baukus J, Clark W (2002) Integrated circuits protected against reverse engineering and method for fabricating the same using an apparent metal contact line terminating on field oxide. US Patent 20020096776
5. Chow L, Baukus J, Wang B, Cocchi R (2012) Camouflaging a standard cell based integrated circuit, US Patent 8151235
6. Cocchi RP, Baukus JP, Chow LW, Wang BJ (2014) Circuit camouflage integration for hardware ip protection. In: Proceedings of the 51st annual design automation conference (DAC 14), New York, NY, USA. ACM, pp 153:1–153:5
7. Rajendran J, Sam M, Sinanoglu O, Karri R (2013) Security analysis of integrated circuit camouflaging. In: Proceedings of the ACM conference on computer and communications security (CCS), pp 709–720

8. Wang X, Zhou Q, Cai Y et al (2016) Is the Secure IC camouflaging really secure. In: Proceedings of the IEEE international symposium on circuits and systems (ISCAS), pp 1710–1713
9. Liu B, Wang B (2014) Embedded reconfigurable logic for ASIC design obfuscation against supply chain attacks. In: Proceedings of the design automation and test in Europe (DATE). IEEE, pp 1–6
10. Wang X, Jia X, Zhou Q et al (2016) Secure and low-overhead circuit obfuscation technique with multiplexers. In: Proceedings of the ACM great lakes symposium on VLSI, pp 133–136