Chapter 11
Obfuscated Built-In Self-authentication

Qihang Shi, Kan Xiao, Domenic Forte and Mark M. Tehranipoor

11.1 Introduction

As discussed in Chap. 1, changing economic trends have resulted in a global IC supply chain. For all but a few semiconductor companies, IC fabrication is now being performed by contract foundries and outside the purview of original intellectual property (IP) owners. There are serious concerns about whether trust between an IP owner and such fabs/foundries can be established [1]. A untrusted foundry with malicious intent could conduct a number of attacks including IP piracy [2], IC cloning/overproduction [3, 4], and hardware Trojan insertion [5].

A great deal of research has been performed to address the attacks associated with untrusted foundries. One approach introduced by DARPA [6] is split manufacturing. In this approach, an untrusted foundry manufactures the front-end-of-line (FEOL) part of the IC (the transistors and lower metal layers) and then ships it to a trusted foundry to deposit back-end-of-line (BEOL) layers, which includes the remaining metal layers (see Fig. 11.1). By concealing complete layout information, split manufacturing prevents the untrusted foundry from stealing IP information or committing attacks that require reverse engineering of the design.

Q. Shi (✉)
ECE Department, University of Connecticut, Storrs, CT, USA
e-mail: qihang.shi@engr.uconn.edu

K. Xiao
Intel Corporation, Santa Clara, CA, USA
e-mail: kan.xiao@intel.com

D. Forte · M.M. Tehranipoor
ECE Department, University of Florida, Gainesville, FL, USA
e-mail: dforte@ece.ufl.edu

M.M. Tehranipoor
e-mail: tehranipoor@ece.ufl.edu

D. Forte et al. (eds.), *Hardware Protection through Obfuscation*,
DOI 10.1007/978-3-319-49019-9_11

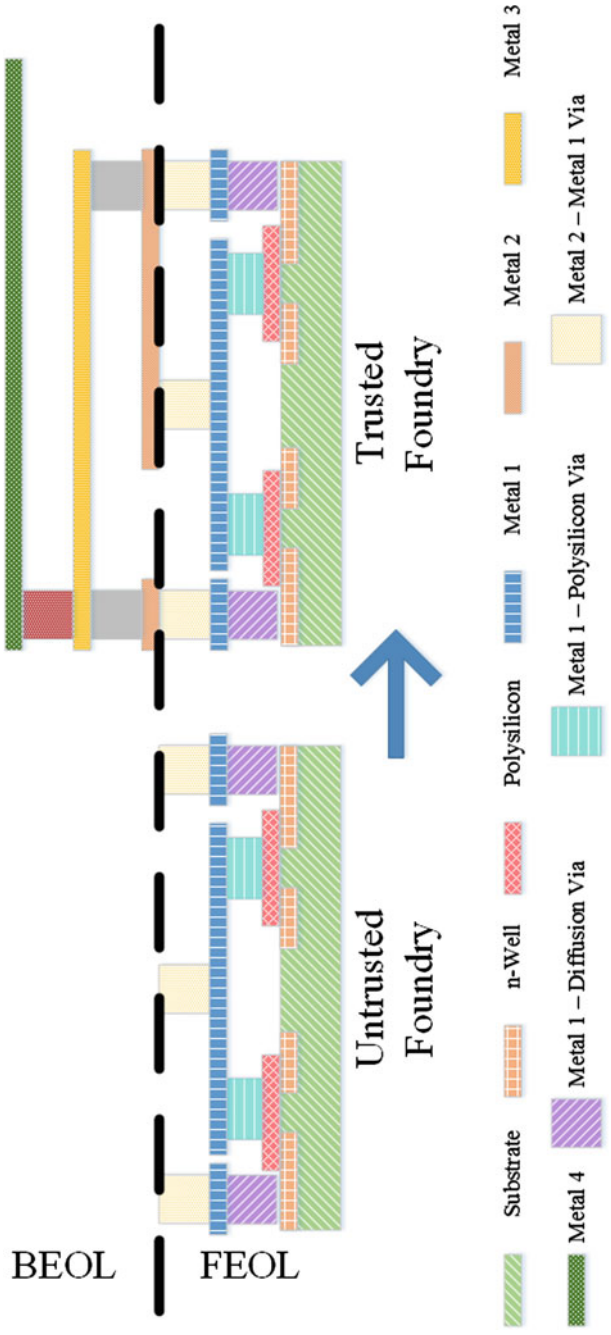


Fig. 11.1 Typical split manufacturing arrangement (assuming split made between Metal 2 and Metal 1 layers)

There has also been a lot of work on protection against the threat of hardware Trojans. Techniques against hardware Trojan insertion can be grouped into two categories, depending on how they address the issue.

- *Detection*: The first category includes Trojan detection techniques, such as functional verification, side-channel signal analysis, or by new front-end design techniques such as design-for-trust [7–15]. Such techniques detect the existence of hardware Trojans by generating a signature of the circuit under test (CUT) and then classifying the CUT with this signature. To perform classification, they require a golden model, i.e., signature of a copy of the same circuit that is known to be free from hardware Trojans. Unfortunately, it remains doubtful whether golden models can be acquired for real-world applications (e.g., commerical off-the-shelf parts). In addition, process variations introduce errors in classification, especially for small, hard-to-detect Trojans.
- *Prevention*: The second category includes hardware Trojan prevention techniques that stop an adversary from inserting a Trojan in the first place and also do not require a golden model. Built-in self-authentication (BISA) is the first proposed technique to prevent hardware Trojan insertion in the circuit layout and mask. By occupying all available spaces for Trojan insertion and detecting malicious removal through built-in self test, BISA is able to deter hardware Trojan insertion without the requirement of golden models and classification errors introduced by process variation.

However, problems remain. Techniques against IP piracy do not usually consider the threat of hardware Trojan insertion. Conversely, techniques against hardware Trojan insertion (such as BISA) do not consider IP theft. Unfortunately, IP piracy and Trojan insertion often go hand-in-hand, with the same adversary capable of pirating the IP as well as inserting a Trojan.

An untrusted foundry is characterized by two attributes: (1) The service of an untrusted foundry is imperative (e.g., due to the high cost associated with advanced nodes). Otherwise, security could be ensured by simply using a trusted foundry and (2) an untrusted foundry cannot be trusted with the security of the intellectual property (IP). Hence, additional measures need to be taken to prevent potential IP piracy. Since both attributes need to be present for split manufacturing to be necessary, we can assume that all untrusted foundries in the adversarial model possess both attributes.

Therefore, in order to ensure the security of fabrication with split manufacturing, we must ensure security against all possible attacks from an untrusted foundry. Due to attribute (1), it is likely that the untrusted foundry is aware of its own criticality, so there is no disincentive for the untrusted foundry to refrain from all possible attacks given its technical capability. At the same time, as a result of attribute (2), the IP owner has no reason to trust the untrusted foundry. It can be reasonable to assume that the untrusted foundries will likely try all attacks in their arsenal, and IP owners will desire an overall solution that can secure their design against all attacks. Therefore, a complete security solution to address the threats from an untrusted foundry is needed. Unfortunately, both split manufacturing and BISA are limited when it comes to comprehensively countering the problem of an untrusted foundry.

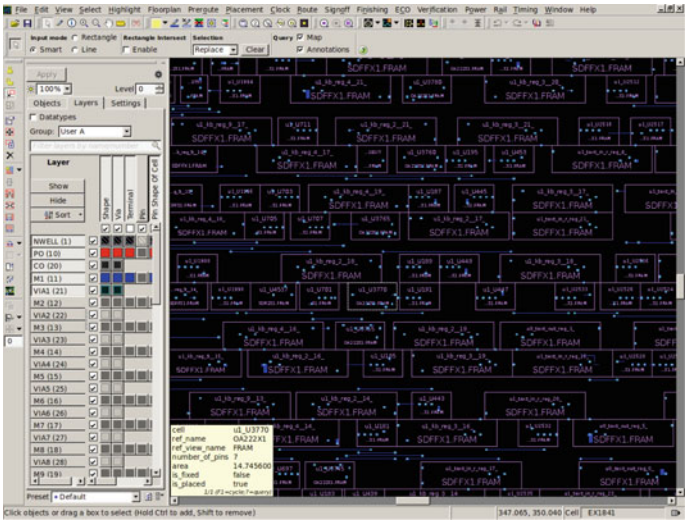
11.1.1 Limitations of Built-In Self-authentication (BISA)

To explain how built-in self-authentication (BISA) works, we need to first establish how normal back-end design of an IC works. Back-end design of an IC is usually in the form of a netlist, i.e., a list of gates and how the nets connect them. This netlist is used to build a layout, which can be used to generate a photolithography mask, which in turn is used to fabricate the IC. The layout phase consists of at least two steps: (i) placement of the gates and (ii) routing of the nets. Normally, during the placement step of the back-end design, gates in the circuit are placed at optimized locations based on density and routability [16]. This leaves so-called *white spaces*, i.e., spaces in the layout that are not filled by standard cells in the layout (see Fig. 11.2a). White spaces have to exist, because gates placed too close to each other will make routing of nets very difficult or impossible. Power dissipation as well as cross talk due to high-frequency gate operations in close vicinity could also generate enough heat and noise in the IC to render it useless.

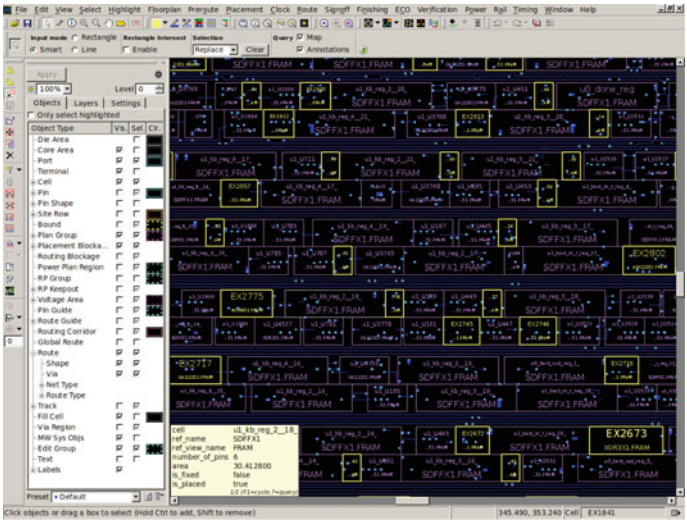
For security-oblivious design purposes, these white spaces are usually filled with *filler cells* to serve as decoupling capacitors and/or extension of power tracks [17]. For such purposes, a filler cell design containing only power tracks, or power tracks and decoupling capacitors, is usually adopted, since they consume less leakage power than standard cells. However, such a simple and unsupervised design also makes these filler cells prone to malicious removal by Trojan inserters in order to make room for hardware Trojans. This is because white spaces are not monitored by any logic. Decoupling capacitors serve performance purposes rather than functional needs. The very reason white spaces exist is because these spaces cannot be occupied with normal functional logic. The problem is that Trojan gates are mostly dormant throughout the host IC's life span. If white spaces or decoupling capacitors are replaced by Trojan gates, it would likely incur a mild level of performance loss (e.g., slight drop in operating frequency). However, the magnitude of a Trojan impact could be so low that designers could simply attribute it to transient conditions. The symptom would be comparable to the case of a mild fever in humans: The patient usually attributes it to stress or other temporary factors, without suspecting any major problem at play.

BISA prevents hardware Trojan insertion by occupying white spaces with testable standard cells instead of non-functional filler cells (see Fig. 11.2b). All inserted BISA cells are organized to form a built-in self test (BIST) circuitry, so that they can be tested to verify that no BISA cells have been removed. BISA is designed so that removal of its member cells will lead to a BIST failure, so that no attempt to make room for hardware Trojans will evade detection. As a technique against Trojan insertion, BISA is highly effective.

Unfortunately, simply securing the design against all hardware Trojan insertion is insufficient at addressing the intended adversary. As we have discussed above, untrusted foundries can and should be expected to commit all kinds of attacks within its capabilities. Unfortunately, the intended attacks BISA is capable of securing against are quite limited in scope. It is unwise to assume an untrusted foundry willing to attempt Trojan insertion will not attempt to commit other attacks, e.g., steal IC



(a) Layout of an AES crypto-core, with unfilled white spaces.



(b) Layout of an AES crypto-core, with white spaces filled with BISA cells.

Fig. 11.2 Layout of an AES crypto-core, with unfilled white spaces

layout in order to perform IP piracy or IC cloning [2, 3]. Therefore, BISA is not a complete solution. Moreover, specific attacks exist for BISA. For example, if the attacker can distinguish BISA cells from original circuitry, it is theoretically possible to perform a "redesign attack" so that BISA detection could be evaded. We shall discuss limitations of BISA in more details in Sect. 11.2.2.

11.1.2 Limitations of Split Manufacturing

Split manufacturing prevents all attacks that require complete knowledge of the whole layout, which also includes attacks against BISA such as identification of BISA cells. However, not all attacks require complete knowledge of the whole layout. One example of these kinds of attacks is untargeted Trojan insertion [18].

A "targeted" hardware Trojan is designed to trigger at certain specified states of the original circuit or to maliciously modify a specific function of the original circuit, or both. Hence, it requires knowledge of the related modules in the original circuitry. In the case where the adversary is an untrusted foundry, this knowledge will also require their locations on the FEOL layout. In a split fabricated IC, at least some nets consist of BEOL interconnects, whose information is thus denied to the untrusted foundry. This will complicate or deter targeted Trojan insertion by making it harder for the untrusted foundry to identify the site that he wants to insert the Trojan trigger or payload. This deterrence is most significant when the split is optimized to maximize the effect of obfuscation, e.g., through the use of wire lifting to maximize security rating k as described earlier in the chapter on 3D IC- based obfuscation. To overcome this obfuscation, the untrusted foundry will have to insert Trojan payloads at all possible sites and/or generate trigger signals using net values from all possible sites, proportional to the design's security rating k. This will cause the Trojan to be proportionally larger and easier to discover and/or trigger.

On the other hand, untargeted Trojan does not require knowledge of the original circuitry [19] and can still pose a threat to split manufactured ICs. Such a Trojan can be designed as long as a sufficiently rare triggering condition can be produced. For example, consider an original circuitry where only front-end-of-line (FEOL) part of the layout is visible, i.e., the portion of the layout visible to an untrusted foundry under split manufacturing. In this scenario, all cells of the original circuitry are visible; therefore, all pins are available for the hardware Trojan. Granted, split manufacturing is effective in denying the untrusted foundry access to backend-of-line (BEOL) information, and the untrusted foundry would not be able to distinguish the functionality the signals at these pins may serve; however, hardware Trojan payloads do not depend on knowing the functionality of the original circuitry, and therefore neither does it require such access to begin with. As long as the Trojan trigger is hard enough to trigger during manufacturing test, the Trojan is capable of evading detection and can cause security concerns.

One way to do this is to choose structurally hard-to-reach nets for the Trojan trigger inputs. Since such nets can be expected to be difficult for manufacture tests to place values at without knowing what functions they serve, it is also unlikely for the manufacture test patterns to trigger the Trojans. This is shown in Fig. 11.3. In this example, D-pins (i.e., input pins) of flip-flops are used to generate the trigger signal. D-pins are a suitable option because they are structurally hard to reach for manufacture test patterns. As is shown in Fig. 11.3, D-pins of flip-flops are at one end of timing paths. During manufacture test, test patterns are fed into the flip-flops with scan paths and launched from the other end of timing paths as shown in the

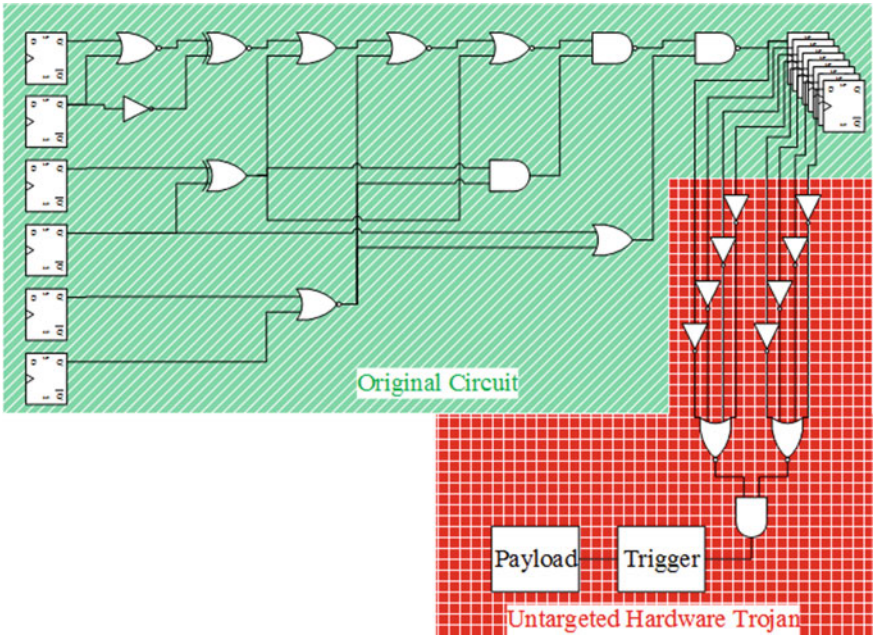


Fig. 11.3 One approach to insert untargeted hardware Trojan into split manufacturing protected layout

figure. This naturally makes D-pins the furthest away from the origin of manufacture test patterns, in terms of number of gates in between. Since each gate in between exponentially increases the number of inputs necessary to control any net (i.e., to place a desired logic state at a given net), this makes D-pins hardest to control from a test coverage point of view. In other words, the number of tests necessary to completely traverse the complete state space of the D-pin nets in order to trigger the Trojan will be too large for manufacture test to implement. Consequently, using D-pins as inputs to generate trigger signal makes the hardware Trojan very hard to discover with manufacture tests.

Another possible way for the proposed hardware Trojan to be discovered during manufacture test is through its impact on path delay. Since D-pin nets are used as inputs to the Trojan's trigger, the trigger adds load capacitance to the net, which adds delay to paths that contain that net. There exists a realistic possibility that this additional delay overhead will change the timing of the affected paths significantly enough to lead to discovery during delay test. Another realistic constraint is the distance between available space for a Trojan gate to be inserted and the D-pin. If the minimum distance is too large, it could also lead to a large resistance on the interconnect between the D-pin net and the trigger gate. This can also add delay, possibly resulting in Trojan detection.

A simple solution to these issues is to add the smallest gate to buffer the trigger input. This is done in the example shown in Fig. 11.3 with inverters. An inverter is the smallest cell in a standard cell library. It can help in minimizing the distance and input capacitance that contributes to delay change to paths that contain D-pin nets. Its weakness is its fan-out load capacity, a quality essential in meeting timing requirement, but this is rather unimportant for inserting hardware Trojans since Trojans usually do not have rigid timing requirements. Therefore, to evaluate the possibility of untargeted Trojan insertion in a layout, one may simply insert filler cells into it, but use an inverter instead of usual filler cell models. This will insert inverters at all available white spaces for it, and a simple geometric search using a reasonably set radius centered at the coordinate of D-pins will yield all possible inverter site candidates that are reasonably close to the D-pins. The layout editor may then be employed to create an interconnect between the inverter inputs and the D-pins to study the impact on delays of paths that ends at those D-pins. Indeed, in one such experiment, we performed with an untargeted Trojan insertion on an open source advanced encryption standard (AES) crypto-core, insertion at 2601 of all 2602 D-pins does not even impact worst-case delays of all paths containing them. In a real implementation where parametric drift of devices due to fabrication can lead to path delay variation of 10 % or more [20], the timing impact will be even less distinguishable.

Untargeted hardware Trojans do not target specific functions of the original circuitry and therefore cannot commit attacks that require knowledge of such functions. Nevertheless, untargeted hardware Trojans are still capable of degrading the performance and/or reliability of manufactured ICs or triggering a denial-of-service (DoS) attack in critical control systems [21]. These are threats that need to be addressed.

Like BISA, split manufacturing has its own share of issues, in addition to the problem of lacking security against untargeted Trojan insertion. As has been introduced in Chap. 10, k-security is an existing security metric of split manufacturing. It evaluates the effectiveness of obfuscation (via denial of BEOL information) by calculating the least number (the security rating k) of mutually indistinguishable gates or nets that exist for any observable gate or net in FEOL. This definition is mathematically sound, but places some rather heavy restrictions on the original circuitry design. For example, unconnected nets are indistinguishable from each other by default, but unconnected gates are not. An inverter is distinguishable from an AND gate and so is an AND gate from another AND gate with twice as much fan-out capacity. As mentioned in Chap. 10, normal synthesis and netlist optimization yield many standard cell models with very few numbers of instances, which will seriously restrict how high k can reach. Consequently, a design that optimizes its k-security rating will have to restrict the standard cell models it uses. Indeed, the authors in [22] restricted the standard cell model count in their experiments between 3 and 7, while an otherwise unconstrained synthesis of an AES core yields 37. This restriction will undoubtedly lead to elevated area and power overhead as well as performance loss. Moreover, white spaces also exist in split fabricated ICs, whose existence and spatial distribution on the FEOL layout could be used to deduct BEOL connections as part of proximity attack [23] (see Chap. 10).

11.1.3 Chapter Overview

While split manufacturing and BISA are excellent techniques, they are still incomplete. An apparent solution is to create a technique that combines the best of both worlds. In this chapter, we term this combined technique as the *obfuscated* built-in self-authentication (OBISA). This chapter provides background information to OBISA, investigation of different ways in which OBISA can be implemented, as well as how protection against both IP theft and hardware Trojan insertion can be better implemented by OBISA than BISA or split manufacturing alone. The rest of this chapter is organized as follows: Sect. 11.2 provides background on BISA and elaborates on existing problems and weakness of BISA that could be improved; Sect. 11.3 reviews split manufacturing techniques introduced in the previous chapter, comments on their relevance to BISA, and investigates possible benefits of their integration with BISA; Sects. 11.4 and 11.5 investigate two possible ways that OBISA can be implemented and provide their implementation flow, their respective strengths, and potential attacks; and finally, Sect. 11.6 concludes this chapter.

11.2 Built-In Self-authentication (BISA)

The purpose of BISA, as we briefly visited in Sect. 11.1, is to prevent hardware Trojan insertion. This objective is achieved by implementing two major features:

1. occupying all white spaces in the layout that could be used for Trojan insertion and
2. ensuring no inserted filler cell for the purpose of preventing Trojan has been removed.

As was discussed in Sect. 11.1, existing design flow already occupies white spaces in the layout with filler cells. That alone is not sufficient to deter malicious Trojan insertion because conventional filler cells are not under any kind of surveillance to prevent them from being removed by an attacker in order to make room for hardware Trojans. Therefore, the second feature is really essential to prevent hardware Trojan insertion. BISA implements this by replacing filler cells with combinational logic cells from the standard cell library and then organizing them into a BIST. Removal of any BISA cell will lead to changes in BIST test signatures and thus detection.

As shown in Fig. 11.4a, BISA consists of three parts: the BISA circuit under test, the test pattern generator (TPG), and the output response analyzer (ORA). The BISA circuit under test is composed of all BISA cells that have been inserted into unused spaces during layout design. In order to increase its stuck-at fault test coverage, the BISA circuit is divided into a number of smaller combinational logic blocks, called BISA blocks as shown in Fig. 11.4a. Each BISA block can be considered as an independent combinational logic block. The TPG generates test vectors that are shared by all BISA blocks. The ORA will process the outputs of all BISA blocks

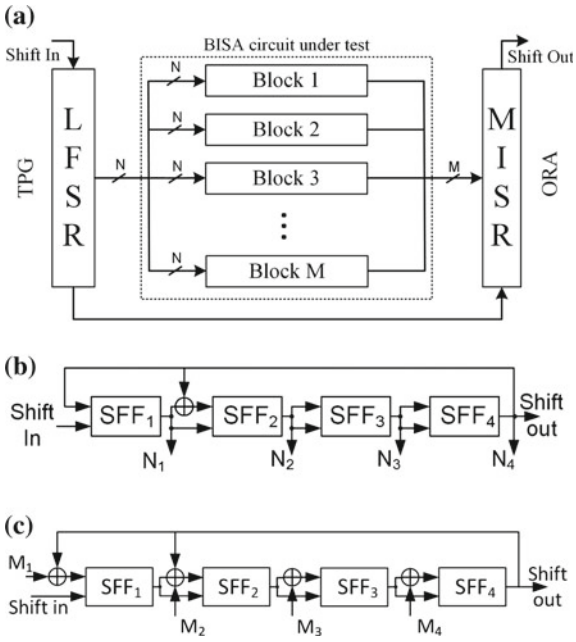


Fig. 11.4 Structure of **a** BISA, **b** four-stage LFSR, and **c** four-stage MISR

and generate a signature. TPG has been implemented with linear feedback shift register (LFSR), while ORA has been implemented with multiple input signature register (MISR) in prior work [24]. Examples of four-stage LFSR and four-stage MISR are shown in Fig. 11.4b, c. They are used in the generation of random vectors and compression of responses into a signature. SFF in the figure represents a scan flip-flop. Other types of TPG and ORA can also be applied [25].

The main advantage of BISA is that it does not require a golden chip/model. Most other researches on addressing the issue of Trojan insertions have focused on the development of:

1. hardware Trojan detection techniques using functional verification and side-channel signal analysis (applied post-silicon) or
2. new design techniques to improve detection capabilities (applied to front-end design) [26].

Most detection approaches need golden chips either fabricated by a trusted foundry or verified to be Trojan-free through reverse engineering, both of which are prohibitively expensive, if not impossible in many scenarios. Since BISA relies on logic testing, process variation is not a factor either, as compared to Trojan detection techniques based on side-channel analysis. As an additional advantage, impact of BISA on original design in terms of area and power is also negligible.

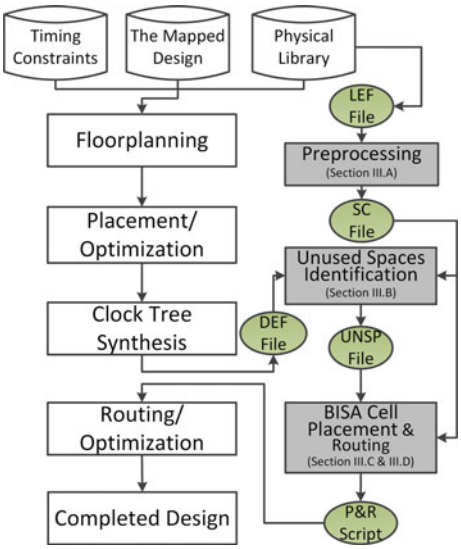


Fig. 11.5 BISA design flow

11.2.1 Implementation Flow

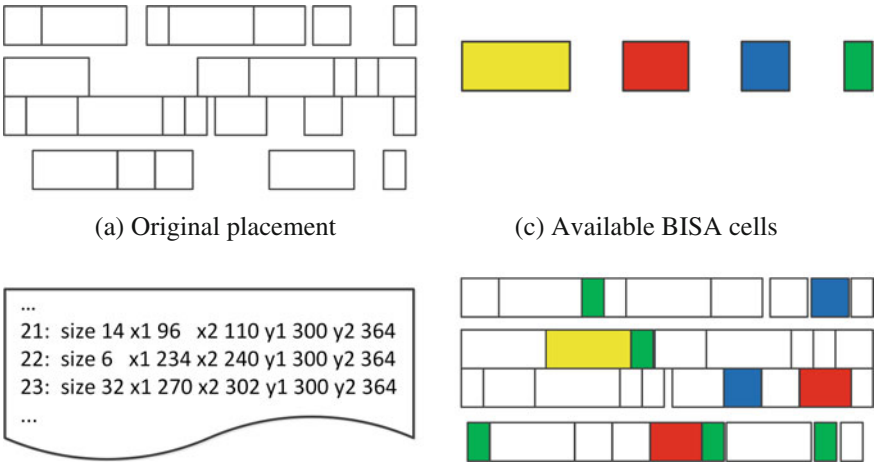
Figure 11.5 shows the BISA design flow and where it fits within the conventional ASIC design flow. The white rectangles in the figure are steps taken in a conventional ASIC design flow, and the gray ones are the additional steps for inserting BISA circuitry.

The first step in BISA design flow is called *preprocessing*, where information such as dimensions of each standard cell, the number of input pins, and the name of a cell is acquired from the standard cell library for use in later steps.

After obtaining the necessary information for all standard cells, BISA cells will be selected from them and marked according to the following criteria:

1. BISA cells must be the minimum-sized cell for every logic function, so they are resistant to a resizing attack by the adversary (see Sect. 11.2.2).
2. The amount of decoupling capacitance the cells can provide and the input count should be considered as well. Fewer inputs help to improve test coverage; therefore, a normalized input count is used here to represent the number of inputs of a standard cell if the same cell has the same area of the minimum-sized cell (e.g., INVx0 in Synopsys 90 nm library).
3. The smallest cell in the library must also be included in order to ensure that no cell can be inserted in any remaining unused space.

The second step is called *unused space identification*, where the BISA flow identifies white spaces by using a matrix to record the state of each point in the layout. Every standard cell placed in the layout will be processed one by one, and eventually,



(a) Original placement (c) Available BISA cells

(b) Unused spaces file (.unsp) (d) Placement after BISA insertion

Fig. 11.6 BISA cell insertion and placement

the matrix reveals the location and size of unused spaces. The matrix is then used to insert BISA cells into these spaces, as shown in Fig. 11.6.

The final step in the flow is to *place and route* BISA cells. Placement solutions can be found and optimized (e.g., dynamic programming algorithm was used in [24]) based on white space identified in the previous step. Optimization of BISA placement is an interesting problem; however, it is not of central importance in BISA. On the other hand, all placed BISA cells need to be connected into a number of combinational BISA circuits (referred to as BISA blocks) to ensure test coverage of the BISA circuit. Test coverage is a key issue for BISA since its security relies on its capability to discover tampering of its constituent cells. A higher test coverage leads to a higher credibility of results from BISA. Several approaches are employed to enhance stuck-at fault test coverage:

- First, create as many BISA blocks as possible to make each BISA block with fewer gates so that higher test coverage is easier to achieve. Since the output of every BISA block will connect to MISR, the number of BISA blocks is determined by the size of MISR. If M is the size of MISR, all placed BISA cells are divided into M groups (BISA blocks).
- Second, redundant gates could deteriorate controllability and observability of the circuit and lower the test coverage significantly, so a tree-structure circuit is constructed to eliminate redundant gates, as shown in Fig. 11.7. If every input is independent of other inputs in a tree-structure circuit, every net is controllable and observable, so the theoretical test coverage of stuck-at fault is 100 %. Here, a tree-structure BISA block is constructed according to the sequence of cells in a block set.

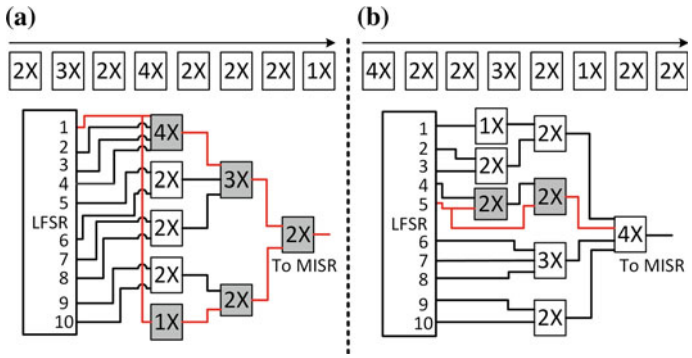


Fig. 11.7 Routing a BISA block

Figure 11.7 shows that two different sequences lead to two different tree-structure circuits. The first gate becomes the root of the tree-structure circuitry, i.e., it is on the top (first) level of tree. The outputs of the next x cells (x being the number of inputs of the root cell) are connected to its inputs as its children cells, on the second level. The same is repeated on the third level to connect new cells to cells on the second level. Cells are sequentially connected to cells on upper levels until all of them are processed, as shown in Fig. 11.7a, b. After complete routing in each block, all inputs of each block should connect to the LFSR sequentially to avoid sharing of inputs. In the end, the M bit outputs from M BISA blocks connect to a MISR with size of M.

11.2.2 Possible Attacks Against BISA

To attack BISA, an attacker would have to find a way to remove enough cells to make room for his Trojan insertion, without triggering detection by BISA. Depending on targets and methods used in this removal, several possible strategies exist to attack BISA:

1. attack TPG or ORA of BISA,
2. directly remove cells from BISA or original circuitry. This is known as a *removal attack*,
3. Replace BISA or original circuitry with a smaller functionally equivalent circuit. This is known as a *redesign attack*. In particular, if standard cells of greater fan-out are replaced with their equivalent counterparts of lower fan-out, this is known as a *resizing attack*.

Of the three possible attacks against BISA, attacking TPG or ORA is the least likely to succeed. BISA uses pseudo-random pattern to perform BIST, which makes it very easy to increase pattern count and consequently very difficult for the attacker to make sure all responses of the modified TPG and/or ORA will stay the same for arbitrarily many patterns. Similarly, direct removal of BISA cells is unlikely

to succeed as they are covered by BISA test coverage during BISA insertion. It is indeed possible to remove cells from original design as long as they do not serve crucial functions. However, design optimization and test coverage will minimize this opportunity for the attacker.

The attack that is most likely to succeed against BISA is the redesign attack. Both BISA and the original circuitry can be targeted in this attack. Redesign attack on original circuitry is restricted by manufacture test as well as other detection-based anti-Trojan approaches. If attackers redesign the original layout for Trojan insertion, moving gate locations and altering wire interconnections will result in significant changes in the electrical parameters, such as power and path delay. These can be detected much more easily by delay-based and power-based techniques [27–36].

It is more likely for the removal attack to succeed against BISA cells since the BISA cells cannot be expected to meet a uniform timing constraint: Their insertion has to prioritize area occupation. The attacker may try to first reverse engineer BISA circuitry—a monumental effort, but not impossible—and then perform logic optimization, hoping to remove redundant BISA cells. It is possible to further secure BISA cells by performing this optimization on BISA design to prevent this particular attack. However, the attacker can also choose to design a custom cell functionally equivalent to several BISA cell at the cost of fan-out and/or delay in order to make room for Trojan insertion. Prevention of such an attack would require anticipation of all possible custom cell designs that are functionally equivalent to any combination of BISA cells. That is not likely feasible except for very small BISA circuitry. Therefore, this attack, called the *custom cell attack*, is also not a likely threat to BISA security.

11.2.3 Limitations of BISA

BISA adversarial model is more or less limited to Trojan attacks after the back-end design. This is because Trojan attacks on the original circuitry can be expected to be detected by other techniques such as functional and delay tests. This leaves untrusted foundry as the most likely adversary to BISA.

One valid limitation of BISA is its inability to prevent IP piracy or IC cloning—a task perfect for split manufacturing to tackle. Another small limitation is that due to the possibility of resizing attacks (see Sect. 11.2.2), all BISA cells have to be of the smallest variant in area among standard cells of the same function, which might make it easier for the attacker to identify them.

11.3 Combining BISA with Split Manufacturing

In light of the respective limitations of BISA and split manufacturing, it makes sense to combine them so that benefits of both techniques can be reaped. We henceforth term the combined technique as obfuscated BISA (OBISA).

The most apparent advantage of the resulting technique is the security against untargeted hardware Trojan insertion, as well as security against IP piracy and IC cloning, both of which are primary strengths of BISA and split manufacturing, respectively. Combining with split manufacturing can also make the resulting OBISA technique secure against redesign attack, since the attacker must first identify which existing cells are connected together before designing a functionally equivalent circuit to replace these existing cells. This will be much harder if the designer lifts the wires that connect them to BEOL, so that BISA structure becomes indistinguishable from the original circuitry.

The obfuscation effect from split manufacturing can further enhance OBISA beyond protection against redesign attack. As mentioned previously in Sects. 11.2.2 and 11.2.3, conventional BISA requires functional and delay tests as well as detection-based anti-Trojan techniques for the security of the original circuitry against redesigning. Although this does not make BISA insecure, detection-based anti-Trojan techniques do rely on golden models for effectiveness. Reliance on these techniques erodes BISA's advantage of not requiring a golden model. Combining with split manufacturing makes the threat of redesign attack much less of a problem due to security of BEOL information and therefore reduces the necessity of using detection based anti-Trojan techniques (which often require a golden model that is not always available). In addition, obfuscation also deters reverse engineering. A relaxed threat from reverse engineering could allow relaxation of other limitations that was not possible with BISA alone, e.g., the requirement of only using the smallest standard cells may not be necessary if the designer can be reasonably confident that OBISA cells will not be identified.

On the other hand, obfuscation in OBISA could also benefit from BISA insertion, owing to additional cells and FEOL interconnects that BISA insertion introduces to the layout. Since the purpose of split manufacturing is to hide BEOL information, most theorized attacks and security metrics (see Chap. 10) define split manufacturing security as anonymity of broken interconnects in FEOL layout [22, 23, 37]: In other words, even and uniform distribution of FEOL features help split manufacturing security. Additional cells and interconnects introduced by BISA circuitry can be very helpful here, because they can be used to compensate rare gate models and interconnect types so that signatures of original circuitry can be hidden. Without such additional obfuscating material, cells of rare gate models have to be banned during synthesis optimization of original circuitry [22], leading to performance loss. Further, proximity attack based on FEOL-observable distribution of gates as well as white spaces could also be foiled by occupying white spaces and compensating spatial distribution of gate types with BISA cells.

To summarize, a combined OBISA technique could derive advantages from both split manufacturing and BISA, as shown in Fig. 11.8.

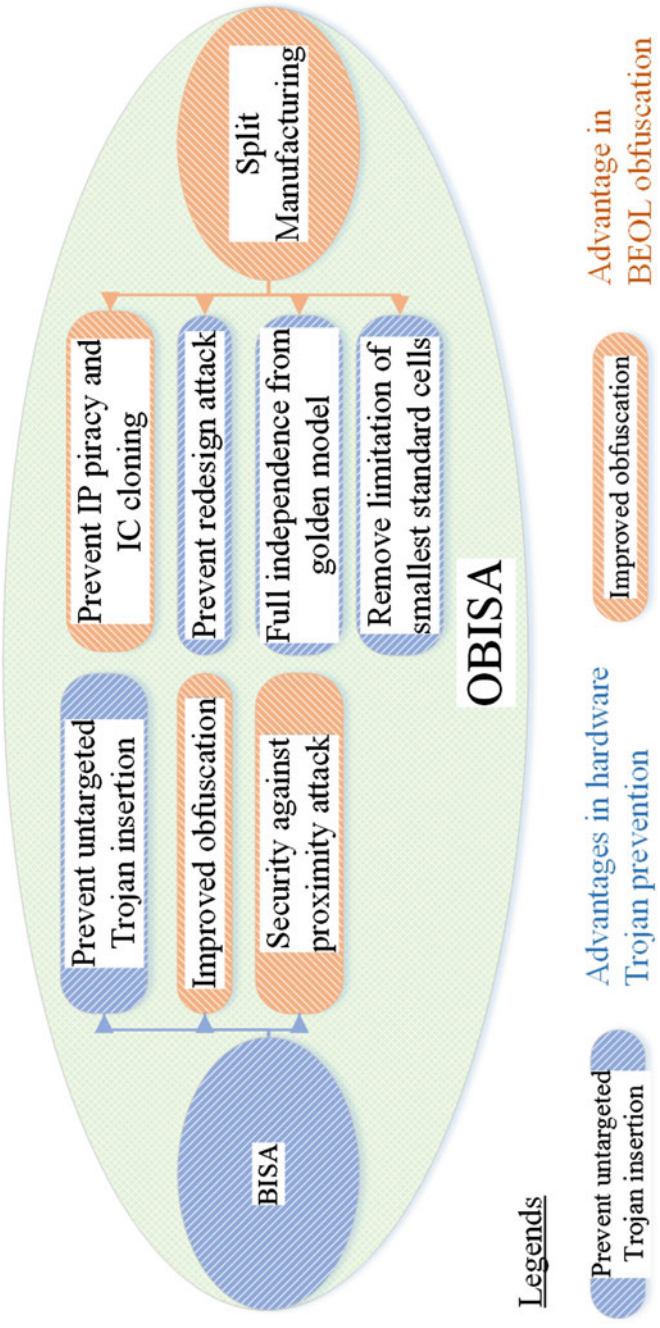


Fig. 11.8 Possible advantages of obfuscated BISA (OBISA) technique

11.3.1 Trade-Off Between BEOL Security and Computational Cost

Split manufacturing techniques, as discussed in Chaps. 10 and 12, come in a number of different implementations. It can be implemented simply by separating FEOL from BEOL at a certain layer without any modification to the design flow; alternatively, wires and/or cell placements in FEOL can be modified to avoid information leakage [23, 37]. In the most secure form of implementation, a list of wires to be elevated to BEOL fabrication can be optimized to obtain a mathematically provable metric of obfuscation [22]. Unfortunately, this is also computationally the most complicated. Generally speaking, there is a trade-off between security and computational cost among split manufacturing techniques.

Depending on how much complexity is dedicated to the split manufacturing side of the technique, OBISA can have different implementation approaches. In this chapter, we introduce two sample approaches:

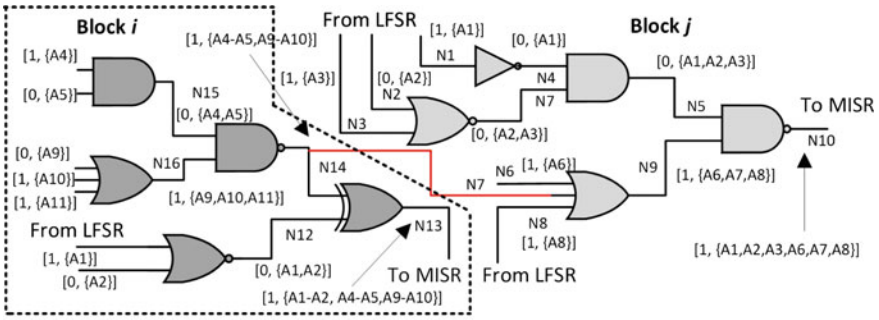
1. Approach A assumes minimum computational cost is dedicated to split manufacturing (i.e., assume split manufacturing is simply implemented by separating FEOL from BEOL at a certain layer) and
2. Approach B assumes maximum level of security is desired (i.e., wire lifting—as was introduced in Chap. 10—is optimized using the notion of k-security).

11.4 Approach A: Obfuscated Connection

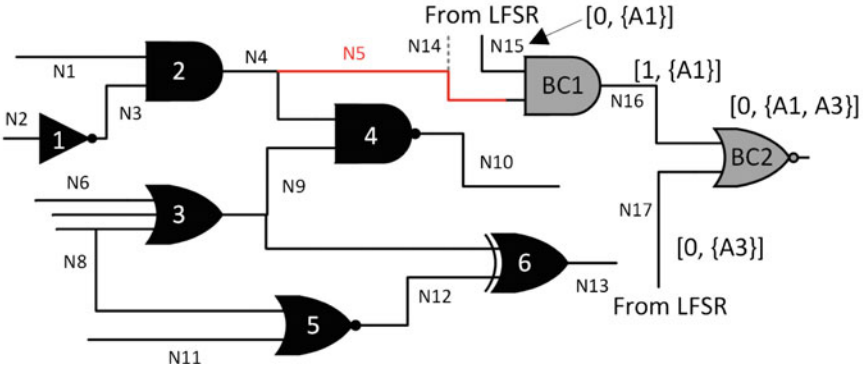
As has been discussed in Sect. 11.3.1, there is a trade-off between security and computational cost among split manufacturing techniques. There is also a trade-off between security and fabrication difficulty, in terms of which layer is used to split the design between FEOL and BEOL. Generally speaking, splitting at higher layers would lead to easier fabrication, higher yield, and lower requirements on the technical capability of the trusted foundry, but would likely leak more interconnect information to the FEOL and thereby the untrusted foundry. From an industrial point of view, a higher split layer is more desirable. In this approach, we assume a split layer at or higher than M3 [18].

To maintain the security of this approach despite reduced obfuscation (due to splitting at higher metal layers), modifications are performed based on the classic BISA structure (which we introduced earlier in the chapter). Specifically, two new types of connections are introduced (see below), and critical wires are lifted to BEOL.

1. The *inter-BISA-block fan-outs*: This refers to an input of a OBISA block being driven by a net in another OBISA block. By doing this, the typical tree-like structure of OBISA blocks can be broken, so that it will become more complicated for an attacker to identify OBISA cells.



(a) A fan-out is made between two OBISA blocks.



(b) An obfuscation connection is made.

Fig. 11.9 Two new types of connections to improve obfuscation

2. The *obfuscation connection (OC)*: This refers to an input of a OBISA block being driven by a net in the original circuitry. By doing this, logic cones in original circuitry are obfuscated with OBISA cells, and identifiable logic patterns are broken.

Adding *inter-OBISA-block fan-outs* (henceforth called "fan-outs") could potentially produce redundant gates and thereby lower controllability of gates. To avoid this, fan-outs can be created using following the rules:

- The fan-out is created between a net in one block i and an input pin of another block j ($i \neq j$) and
- The net in block i and the root output in block j have no common related inputs from OBISA LFSR.

If these two conditions are satisfied, the net and the input pin can form a candidate pair for a fan-out. Figure 11.9a shows an example of the fan-out creation. The net N14 in OBISA block i has completely different related inputs of LFSR from the root output N10 in OBISAv block j, so N14 can have a fan-out to connect to any input in

OBISA block j. In Fig. 11.9a, the pin for the net N7 is selected. Note that a fan-out cannot be made on the net N13, because the net N13 and net N10 share related inputs of LFSR, A1 and A2.

The creation of *obfuscation connections* involves two issues. First, activity from the original circuitry must not propagate into OBISA. Otherwise, it could cause unnecessary power consumption. This is ensured by choosing the right kind of gates that have the same controlling value as the LFSR's idle states so that they can isolate OBISA from the original circuitry when OBISA is idle. As shown in Fig. 11.9b, cells BC1 and BC2 are both gated in idle state. BC1 is selected since it is a leaf cell in the tree-structure OBISA block. The other issue that needs consideration is that it will inevitably add capacitive load to the original circuitry net it is attached to. The added capacitance could potentially cause paths to fail in the original circuitry. Thus, we must select target nets in the original circuitry for the obfuscation connection very carefully to avoid timing violations. One way to do this is to perform static timing analysis (STA) prior to the creation of obfuscation connections and only choose nets whose worst-case paths are faster than the critical path by a margin.

A final step of this approach is to perform wire lifting to restrict FEOL interconnect information. The main problem with wire lifting is that finding the best solution requires high computational cost. Since Approach A is geared toward minimizing computational cost, wire lifting in this approach is limited to security critical block key to OBISA functions, for example, the mode select net, the feedback nets in LFSR/MISR, and nets connecting flip-flops in LFSR/MISR.

11.4.1 Implementation Flow

Figure 11.10 presents the design implementation flow of Approach A. The flow fits within the conventional ASIC design flow and is compatible with current commercial physical design tools. OBISA insertion procedure begins after clock tree synthesis. At that point, the whole original circuit has been placed and no more cells will be added in conventional flow (the most left column in Fig. 11.10). The unused spaces would be identified in DEF file, and various standard cells are inserted depending on size of each unused space. Once all unused spaces are filled with OBISA cells, all OBISA cells in each geometric region will be connected to construct an OBISA block. These steps as shown in the middle column were developed in [24, 38]. New steps, as shown in the third column in Fig. 11.10, are introduced to strengthen obfuscation, including fan-out creation, adding obfuscation connections, and lifting secure-critical paths within OBISA. After the OBISA process, the flow resumes the procedures in the conventional design flow. The physical design tool will perform routing for the entire design including original circuit and OBISA circuit. All constraints for the original design can be taken care of by the physical design tool during routing process. Once the timing and sign-off of the design are successful, the last step involves the generation of a GDSII format of the design for final tape-out.

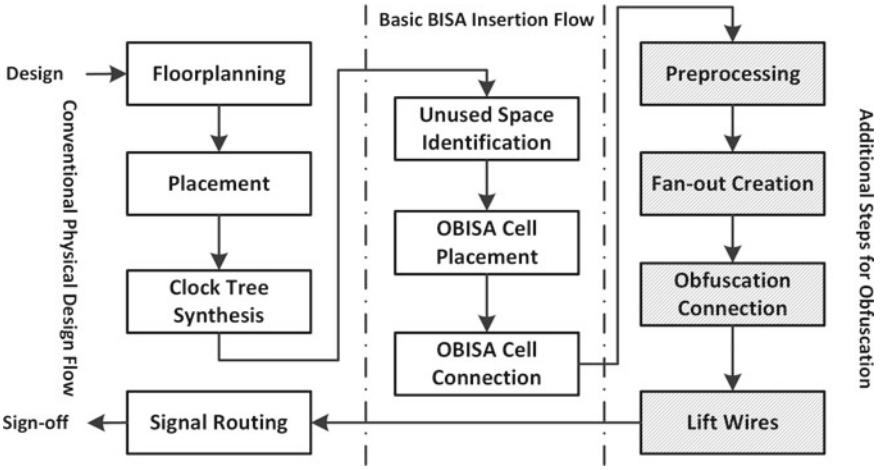


Fig. 11.10 Implementation flow of OBISA technique, Approach A

11.4.2 Design Issues and Trade-Offs

The main trade-off of Approach A results from the two new types of connections it adds to the original BISA design. Both fan-outs and obfuscation connections modify BISA's built-in self test architecture, which could lead to redundant gates, redundant nets, and loss of controllability. These translate into test coverage loss, which means that "blind corners" of OBISA gates are created, whose removal cannot be detected by OBISA. As long as reverse engineering of the OBISA structure can be prevented, this will not be a security issue, but a lowered test coverage nevertheless hurts the credibility of the authentication result. Comparatively, fan-outs between OBISA blocks are easier to handle as they can be made to be fully controllable, while the obfuscation connections will cause more severe loss of test coverage, since their value comes from and is subjected to the status of the original circuitry. However, this can be remedied by performing multiple tests, each starting from an independent state of the original circuitry. Indeed, results in [18] show that even for obfuscation connections which are as many as 25 % of all OBISA inputs, 100,000 random patterns yield just above 60 % coverage, while 10,000 random patterns applied from 10 iterations of the original circuitry can boost the test coverage to almost 100 %.

One optional feature of Approach A is it can choose to use only minimum-sized cells for OBISA structure to keep its security against resizing attacks or dropping that requirement for better obfuscation between OBISA cells and original circuitry cells. This is due to the fact that Approach A does not perform optimized wire lifting as is done in Approach B, nor does it need to introduce compensation OBISA cells that may not have minimum sizes, as we shall see in Approach B. In this regard, Approach A is closer to the classic BISA design.

11.4.3 Potential Attacks

One possible attack against Approach A of OBISA technique is redesign attack. Owing to a higher split layer, it is likely that many small-scale logic blocks such as adders, decoders, and finite-state machines can be identified and optimized by an untrusted foundry. Wire lifting could prevent this. However, large-scale wire lifting would not be possible under our assumption that Approach A is supposed to serve as an example of OBISA technique with minimum computational cost dedicated to the split manufacturing side. Obfuscation connections and inter-OBISA-block fan-outs could help in reducing such signatures. Unfortunately, without a metric dedicated to computational complexity, it is hard to say how much this could help.

11.5 Approach B: OBISA with Wire Lifting

In Sect. 11.4, we discussed an approach to implement OBISA with minimum computational cost. In this section, we discuss the possibility on the other end of the cost-security trade-off axis, which is how maximum security could be achieved with large-scale wire lifting.

To achieve maximum security, it is necessary to first define security. In this approach, we use the k-security definition as was introduced in [22]. k-security has been discussed in more detail in Chap. 10, so we are only giving a brief revisit of the idea here. Consider an IC secured with split manufacturing. Its netlist can be modeled as a graph, where each of its gates is represented by a vertex and each interconnect by a number of edges connecting such vertices. Assuming limited number of custom cells, all models of the gates can be represented by coloring the vertices. Now, remember that we are considering an IC secured with split manufacturing. The FEOL part of its layout contains netlist information of all the gates and a subset of all interconnects. The corresponding graph of the FEOL part, compared to the graph of the complete netlist, will look like the second graph with some of its edges hidden away. Those hidden edges correspond to interconnects reserved to the BEOL part of the IC.

An example of this process is shown in Fig. 11.11. Shown on top is a netlist of a full adder, where black lines mark interconnects visible in FEOL layout, while gray lines mark interconnects in BEOL layout. If we further assume FEOL layout of the full adder splits at its input and output pins, the FEOL layout of the full adder is represented by the graph below.

From the figure, we notice it is impossible to distinguish the two XOR gates (represented by vertices shaded in red slash) in FEOL layout. According to k-security, XOR gates in this full adder have a $k = 2$ security. If the same can be said for all other gate models, k-security metric dictates the FEOL layout has at least $k = 2$ security. Unfortunately, as we can see from the figure, it is impossible for the full adder to

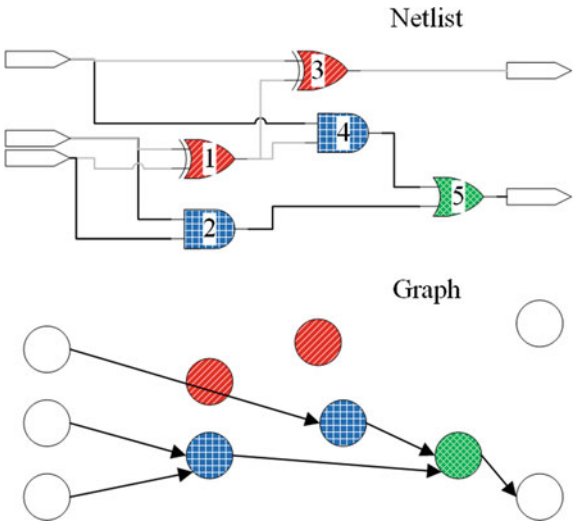


Fig. 11.11 Example: netlist and graph of a split manufactured full adder

reach $k = 2$, even if we lift all edges to BEOL, simply because it has only one OR gate. For this full adder, any optimization of wire-lifting solution is futile.

There are a few ways to address this issue. In [22], only 3 to 7 gate models are allowed during design synthesis, in order to prevent rare gate models from restricting wire-lifting optimization. From a designer's point of view, however, this approach seriously impacts the performance of the original circuitry and could cause serious overhead in area and power.

However, an OBISA technique that performs wire lifting does not have to submit to this restriction, because the number of instances of rare gate models can be compensated with OBISA cells. For example, an AES crypto-core netlist after unconstrained synthesis and optimization has more than 26,000 gates. Among them, 20 gate models have less than 100 instances. Meanwhile, simple BISA insertion into its layout at a normal 0.7 utilization ratio typically yields about 5,000 BISA cells. In other words, OBISA cell count under typical utilization ratio is more than sufficient to compensate rare gate models to 100 instances, a number much higher than what existing wire-lifting algorithm will likely be able to work at within realistic processing time. In [22], the highest reported k is 48 and was only achieved on a much smaller benchmark circuit (c432 from ISCAS, gate count 147).

An example of this advantage is shown in Fig. 11.12. The same full adder is used as shown in Fig. 11.11. In this example, OBISA cells and interconnects are added, as shown in dashed lines. We can see from the example how the bottleneck in previous example—the single OR gate—is compensated with OBISA cells. In the shown wire-lifting example, $k = 2$ security is reached. If we consider a more extreme solution, for example, lifting all wires to BEOL, at maximum, the layout could reach $k = 4$ security rating.

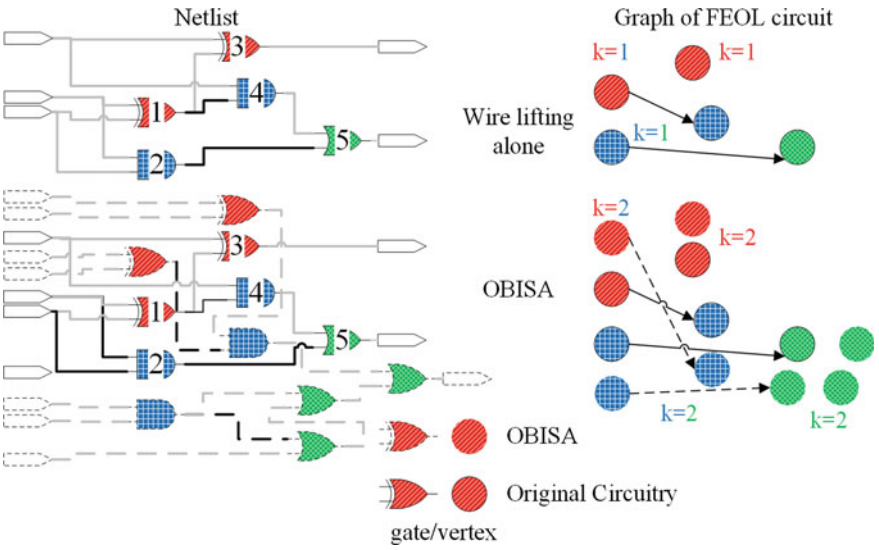


Fig. 11.12 Example: With OBISA insertion, the same full adder can benefit from wire-lifting optimization to achieve higher k-security rating

We can also see from the example that to reach a high security rating k, a large percentage of wires has to be lifted to BEOL for almost every gate. One advantage of this result is that most logic blocks will unlikely be distinguishable in the resulting FEOL layout. This will greatly deter IP piracy as well as redesign attacks, making this approach much more secure. On the other hand, this will also result in a lot of vias having to be matched between FEOL and BEOL, making fabrication more complicated.

Another inference from this example is that the previous limitation (to only use the smallest standard cell models for BISA cells to prevent resizing attacks) must be dropped. This is because it is unlikely for all rare gate models to be of the smallest size. However, as long as obfuscation due to wire lifting holds (i.e., the circuit has a high k-security rating), the attacker will not be able to distinguish OBISA cells from original circuitry cells. Resizing original circuitry cells runs the risk of being discovered by a simple delay test, and this will likely be a sufficient deterrent against resizing attacks.

11.5.1 Implementation Flow

The implementation flow of the OBISA technique (Approach B) is shown in Fig. 11.13. As shown in the diagram, boxes shaded with blue slashes represent procedures already existing in the BISA flow, while boxes shaded with red crosses repre-

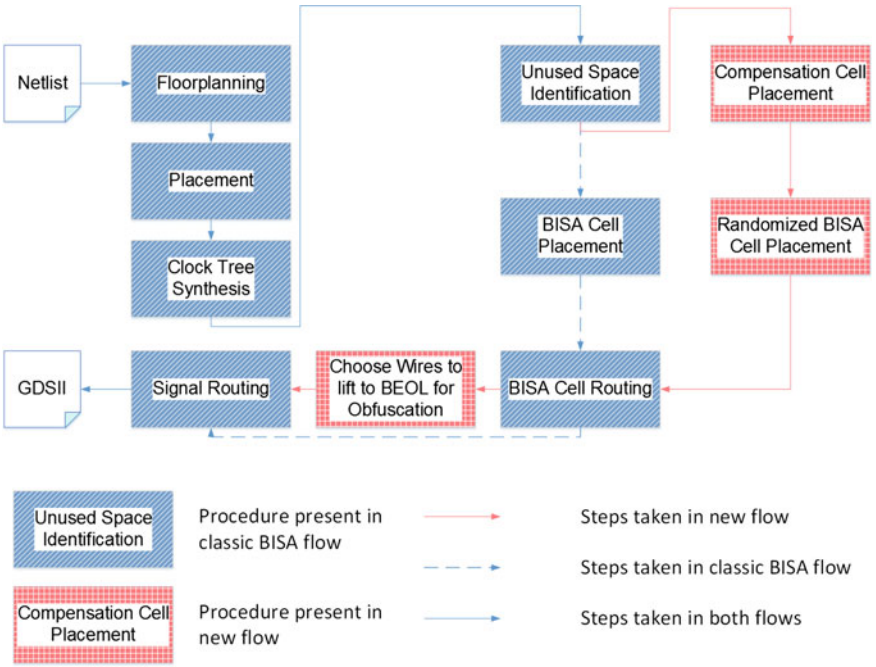


Fig. 11.13 Implementation flow of OBISA technique, Approach B

sent new procedures in this approach. This technique departs from classic BISA after unused space identification: Instead of performing BISA cell placement, cells of the rare gate models are placed first to compensate gate model distribution. After these cells are inserted, placement of BISA cells with random gate models is performed to fill remaining white spaces. After that, normal BISA cell routing is performed. Before signal routing, an optimized wire-lifting solution is found for the complete layout. Wire lifting can then be performed with the help of layout editor, for example, by simply elevating routed interconnects to BEOL metal layers. The rest of the design flow does not differ from existing back-end design flow.

11.5.2 Design Issues and Trade-Offs

The main issue with this approach is cost. Searching for optimal wire-lifting solution is computationally costly, and lifting a sizable percentage of wires to BEOL results in cost in yield loss, elevated requirement on trusted foundry, etc. Between the two costs, another trade-off exists: Obviously, if we choose to lift all wires to BEOL, computational cost will be minimal, security will be maximum, but fabrication cost will be astronomical. The dilemma here is that computational complexity of wire-

lifting solution optimization is a Sharp-P problem, because it consists of a greedy algorithm that exhaustively traverses and verifies the entire solution space, where each step is a Boolean satisfiability problem—a known NP-hard problem. Meanwhile, fabrication cost is not exactly easy to accommodate either. Unless an improvement on at least one of both problems emerges, the cost of Approach B is likely caught in between rock and hard places.

11.5.3 Potential Attacks

One possible attack is based on spatial distribution of cells in FEOL layout. Although k-security metric is defined as at least k mutually indistinguishable instances of each gate model, it goes without saying that this does not account for where those instances are located on FEOL layout. For example, it is certainly logical to assume that a NAND gate very close to a memory cell array is more likely to be a part of the load-store unit than the execution unit. The problem with this kind of information leakage is that little has been established on how the attacker can utilize it. Further, although it is still unclear how the attacker could utilize this information, methods to restrict this leakage of information nevertheless exist. In [22], it is proposed that performing layout design *after* wire lifting could anonymize the layout and prevent any information leakage. Unfortunately, this will likely make back-end optimization of timing a nightmare. Besides, this is not applicable in OBISA since BISA insertion has to come after placement. Still, OBISA in Approach B can partition the complete layout into a number of smaller sublayouts and perform wire lifting on each of them so that wire lifting in each sublayout is relevant to cells in close vicinity. This technique may limit the maximum achievable security rating, but it makes it possible to parallelize the wire-lifting algorithm.

11.6 Conclusion

In this chapter, we first reviewed both BISA and split manufacturing techniques in terms of their adversarial models and pointed out their common adversary, the untrusted foundry. We also explained why the untrusted foundry cannot be trusted to not try attacks that are beyond the scope of a BISA-only design flow (IP piracy) or split manufacturing-only approach (Trojan insertion). We then provided a detailed background for the BISA technique and showed how a combination of both techniques, which we termed as "OBISA," could be expected to be effective against both kinds of attacks. We then investigated two possible approaches to implement the OBISA technique depending on the trade-offs between security and computational/fabrication cost. We provided details on their implementation, discussed the design trade-offs involved, introduced their respective strengths and weaknesses, and theorized how either approach could be attacked.

References

1. Guin U, Forte D, Tehranipoor M (2013) Anti-counterfeit techniques: from design to resign. In: 14th international workshop on microprocessor test and verification, pp 89–94. IEEE
2. Tehranipoor MM, Guin U, Forte D (2015) Counterfeit integrated circuits. Springer, Switzerland, pp 15–36
3. Guin U, Shi Q, Forte D, Tehranipoor MM (2016) Fortis: a comprehensive solution for establishing forward trust for protecting ips and ics. ACM Trans Des Autom Electron Syst (TODAES) 21(4):63
4. Guin U (2016) Establishment of trust and integrity in modern supply chain from design to resign
5. Xiao K (2015) Techniques for improving security and trustworthiness of integrated circuits
6. IARPA Trusted Integrated Circuits (TIC) program announcement. http://www.fbo.gov
7. Salmani H, Tehranipoor M, Plusquellic J (2012) A novel technique for improving hardware trojan detection and reducing trojan activation time. IEEE Trans Very Large Scale Integr (VLSI) Syst 20(1):112–125
8. Li J, Lach J (2008) At-speed delay characterization for ic authentication and trojan horse detection. In: IEEE international workshop on hardware-oriented security and trust, 2008. HOST 2008. IEEE, pp 8–14
9. Jin Y, Kupp N, Makris Y (2010) Dftt: design for trojan test. In: 2010 17th IEEE international conference on electronics, circuits, and systems (ICECS). IEEE, pp 1168–1171
10. Rajendran J, Jyothi V, Sinanoglu O, Karri R (2011) Design and analysis of ring oscillator based design-for-trust technique. In: 29th VLSI Test Symposium. IEEE, pp 105–110
11. Salmani H, Tehranipoor M (2012) Layout-aware switching activity localization to enhance hardware trojan detection. IEEE Trans Inf Forensics Secur 7(1):76–87
12. Chakraborty RS, Bhunia S (2009) Security against hardware trojan through a novel application of design obfuscation. In: Proceedings of the 2009 international conference on computer-aided design. ACM, pp 113–116
13. Banga M, Hsiao MS (2011) Odette: a non-scan design-for-test methodology for trojan detection in ics. In: 2011 IEEE international symposium on hardware-oriented security and trust (HOST). IEEE, pp 18–23
14. Chakraborty RS, Bhunia S (2009) Harpoon: an obfuscation-based soc design methodology for hardware protection. IEEE Trans Comput Aided Des Integr Circ Syst 28(10):1493–1502
15. Rajendran J, Pino Y, Sinanoglu O, Karri R (2012) Security analysis of logic obfuscation. In: Proceedings of the 49th annual design automation conference. ACM, pp 83–89
16. Yang X, Choi B-K, Sarrafzadeh M (2003) Routability-driven white space allocation for fixed-die standard-cell placement. IEEE Trans Comput Aided Des Integr Circ Syst 22(4):410–419
17. Charlebois S, Dunn P, Rohrbaugh G (2008) Method of optimizing customizable filler cells in an integrated circuit physical design process, 28 October 2008, uS Patent 7,444,609. https://www.google.com/patents/US7444609
18. Xiao K, Forte D, Tehranipoor MM (2015) Efficient and secure split manufacturing via obfuscated built-in self-authentication. In: 2015 IEEE international symposium on hardware oriented security and trust (HOST). IEEE, pp 14–19
19. Xiao K, Forte D, Jin Y, Karri R, Bhunia S, Tehranipoor M (2016) Hardware trojans: lessons learned after one decade of research. ACM Trans Des Autom Electron Syst 22(1):6:1–6:23. http://doi.acm.org/10.1145/2906147
20. Shi Q, Tehranipoor M, Wang X, Winemberg L (2014) On-chip sensor selection for effective speed-binning. In: 2014 IEEE 57th international midwest symposium on circuits and systems (MWSCAS). IEEE, pp 1073–1076
21. Turk RJ et al (2005) Cyber incidents involving control systems. Idaho National Engineering and Environmental Laboratory
22. Imeson F, Emtenan A, Garg S, Tripunitara M (2013) Securing computer hardware using 3d integrated circuit (ic) technology and split manufacturing for obfuscation. In: Presented as part of the 22nd USENIX security symposium (USENIX Security 13), pp 495–510

23. Jagasivamani M, Gadfort P, Sika M, Bajura M, Fritze M (2014) Split-fabrication obfuscation: metrics and techniques. In: 2014 IEEE international symposium on hardware-oriented security and trust (HOST). IEEE, pp 7–12
24. Xiao K, Forte D, Tehranipoor M (2014) A novel built-in self-authentication technique to prevent inserting hardware trojans. IEEE Trans Comput Aided Des Integr Circ Syst 33(12):1778–1791
25. Bushnell M, Agrawal VD (2000) Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits, vol 17. Springer Science & Business Media, New York
26. Jha S, Jha SK (2008) Randomization based probabilistic approach to detect trojan circuits. In: 11th IEEE high assurance systems engineering symposium, 2008. HASE 2008. IEEE, pp 117–124
27. Wang X, Tehranipoor M, Plusquellic J (2008) Detecting malicious inclusions in secure hardware: challenges and solutions. In: IEEE international workshop on hardware-oriented security and trust, 2008. HOST 2008. IEEE, pp 15–19
28. Agrawal D, Baktir S, Karakoyunlu D, Rohatgi P, Sunar B (2007) Trojan detection using ic fingerprinting. In: 2007 IEEE symposium on security and privacy (SP'07). IEEE, pp 296–310
29. Narasimhan S, Wang X, Du D, Chakraborty RS, Bhunia S (2011) Tesr: a robust temporal self-referencing approach for hardware trojan detection. In: 2011 IEEE international symposium on hardware-oriented security and trust (HOST). IEEE, pp 71–74
30. Zhang J, Yu H, Xu Q (2012) Htoutlier: hardware trojan detection with side-channel signature outlier identification. In: 2012 IEEE international symposium on hardware-oriented security and trust (HOST). IEEE, pp 55–58
31. Wei S, Meguerdichian S, Potkonjak M (2010) Gate-level characterization: foundations and hardware security applications. In: Proceedings of the 47th design automation conference. ACM, pp 222–227
32. Aarestad J, Acharyya D, Rad R, Plusquellic J (2010) Detecting trojans through leakage current analysis using multiple supply pad s. IEEE Trans Inf Forensics Secur 5(4):893–904
33. Alkabani Y, Koushanfar F (2009) Consistency-based characterization for ic trojan detection. In: Proceedings of the 2009 international conference on computer-aided design. ACM, pp 123–127
34. Jin Y, Makris Y (2008) Hardware trojan detection using path delay fingerprint. In: IEEE international workshop on hardware-oriented security and trust, 2008. HOST 2008. IEEE, pp 51–57
35. Xiao K, Zhang X, Tehranipoor M (2013) A clock sweeping technique for detecting hardware trojans impacting circuits delay. IEEE Des Test 30(2):26–34
36. Cha B, Gupta SK (2013) Trojan detection via delay measurements: a new approach to select paths and vectors to maximize effectiveness and minimize cost. In: Design, automation & test in Europe conference & exhibition (DATE). IEEE, pp 1265–1270
37. Rajendran JJ, Sinanoglu O, Karri R (2013) Is split manufacturing secure? In: Proceedings of the conference on design, automation and test in Europe. EDA Consortium, 2013, pp 1259–1264
38. Xiao K, Tehranipoor M (2013) Bisa: built-in self-authentication for preventing hardware trojan insertion. In: 2013 IEEE international symposium on hardware-oriented security and trust (HOST). IEEE, pp 45–50