Hardware & Embedded Security

Prof Daniele Rossi


Via G. Caruso 16, room B-1-03


daniele.rossi1@unipi.it


050 221 7611

Design for Hardware Trust

Hardware Trojans Detection Methodologies

Lecture 5-Part 2 - DR

Brief Outline

- **Design for hardware trust: Hardware Trojans Detection Methodologies**
  - Trojan Activation Method
  - Rare event removal: idea is not have events that are way too rare, so it is more difficult to insert evil hoses.
  - Design obfuscation: make life difficult for attackers: alter state diagram that describes our system, add non-functional states to make reverse engineering harder
  - Built-In Self-Authentication
    ↳ Insertion of additional circuitry that doesn't change sys layout and functionality, so that if those cells have been manipulated, I can find it by testing exactly them.

3

Design for Hardware Trust

- Since detecting Trojan is extremely challenging, additional **design for hardware trust** approaches are proposed to
  - **Improve hardware Trojan detection methods**
    - Improve sensitivity to side channel analysis → **Trojan Activation Methods**
    - **Rare event removal**
  - **Prevent hardware Trojan insertion**
    - **Design obfuscation**
    - **Built-In Self-Authentication**

4

SECTION III.
Probabilistic Signature of a Circuit

In this section, we show how to obtain a probabilistic signature of a Boolean circuit. Without loss of generality, let us assume that the circuit has a set of inputs I and a single output O. For circuits with more than one output, we can consider each output and its fan-in cone as a separate circuit. We find a probability distribution, P, of the input assignment such that the random application of input from the distribution leads to a unique probability of the output of the circuit being 1. Then, distinguishing CUT with its design is done by finding the probability of the output being 1 by applying inputs randomly from the probability distribution P. If the CUT is infected with a trojan, then the probability of logic 1 at the output of CUT and original circuit will be different.

Trojan Activation Methods

- Trojan activation strategies can **accelerate the Trojan detection process**, and in some cases have been **combined with power analysis** during implementation
- If a portion of the Trojan circuitry is activated, the Trojan circuit **will consume more dynamic power**, which will further help differentiate the power traces of Trojan-inserted and Trojan-free circuits

& Power consumption sign.

- Some techniques rely on the application of **test patterns** to generate a **signature** (this case, consisting of a specific set of logic patterns) and compare it with that generated by an *IC under authentication* (IUA) (***Region-Free Trojan Activation***, e.g., [5] and [6]):
  - If there are difference in the outputs, then a Trojan is present
- **Question**: what limitations do you see in this kind of techniques?

5

Trojan Activation Methods

- Other techniques aim at **magnifying the difference** between the IUA and the genuine design in some circuit characteristics (for instance, power consumption): ***Region-Aware Trojan Activation*** (e.g., [7])
- First stage: **circuit partitioning** → a region-aware pattern helps identify the potential Trojan insertion regions → To detect a Trojan, the **activity** within a portion of the circuit **is increased** while the activity for the rest of the circuit is simultaneously minimized
- Second stage: **activity magnification** → new test patterns concentrating on the identified regions are applied to **magnify the disparity** between the original and Trojan-inserted circuits

6

Obj: combine orthogonal approaches so that their effects of detection combine.
So power analysis approach together with region free hogan as our idea.

If I do circuit partitioning (ind. functional areas of circuit, especially ones with known hs) and use test vectors to stimulate those parts with a X mark as possible, while the other circuit portions are stimulated the least amount as possible, you can find what you are looking for.

Trojan Activation Methods

- Example [7]: the power profile of the genuine circuit is computed first, and mainly targets dynamic power
- The **dynamic power** for an IC is proportional to the operating **frequency *f***, **switching capacitance C**, **switching activity** α, and supply voltage $\mathbf{V_{DD}}$:

$$P_{dyn-av} = \alpha_{0\to1} C_{Load} V_{DD}^2 \, f_{ck}$$

- Partition rational: Considering the fact that Trojans are mute spectators for most part of the operational cycle of the circuitry → Trojans are most likely to be associated with those signals related to the **circuit state elements** (i.e., FFs), rather than primary inputs

internal nodes

state variables are going to likely be triggers, while we apply tests from primary inputs.

7

Trojan Activation Methods

- Once we have identified the **regions of interest**, we attempt to create an **activity peak** on a **per-region** basis
- For this, we simulate the circuit with vectors that **maximize the switching activity** within the **region of interest** while simultaneously **minimizing the switching activity** for the **rest of the circuit** (1)
- Thus, if in-region activity and out-region activity represent the amount of switching activity for the gates within the region of interest and for the rest of the circuit respectively; then our **objective function** is defined by:

$$F = \max(in.region\ activity - out.region\ activity)$$

(1) This combined with the probing at specific points for power analysis, we maximise probability of detection.

9

Trojan Activation Methods

- The behaviour of a Trojan is **detectable** only if the difference in the activity of the Trojan-infected chip and the genuine chip (without Trojan) is **above the process variation**

→ the Trojan is most detectable when the **power consumed in the genuine circuit is kept low** → a small region is exercised while keeping the rest at low activity

10

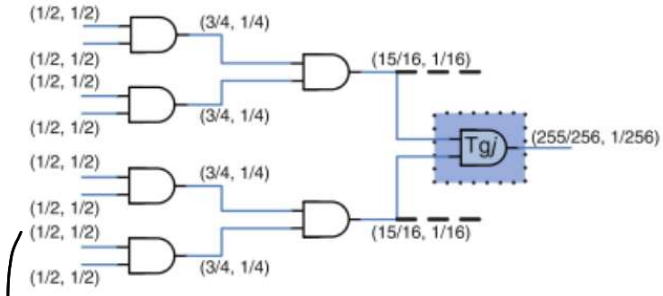
Trojs will activate on rare events, of course. This makes functional tests more ineffective.
What you can do: identified bits that correspond to rare events that can be used for trg activation,
I can modify circuit so that during testing I can increase occurrence number of those events that under normal working conditions would be very rare.

Rare Event Removal

- Intelligent attackers will choose low-frequency events to trigger the inserted Trojans
- **Improving controllability or observability** can make rare events scarce, thereby facilitating detecting Trojans inside the design
  - Design for Trojan test: inserting **probing points**
  - Inserting **dummy scan flip-flops**

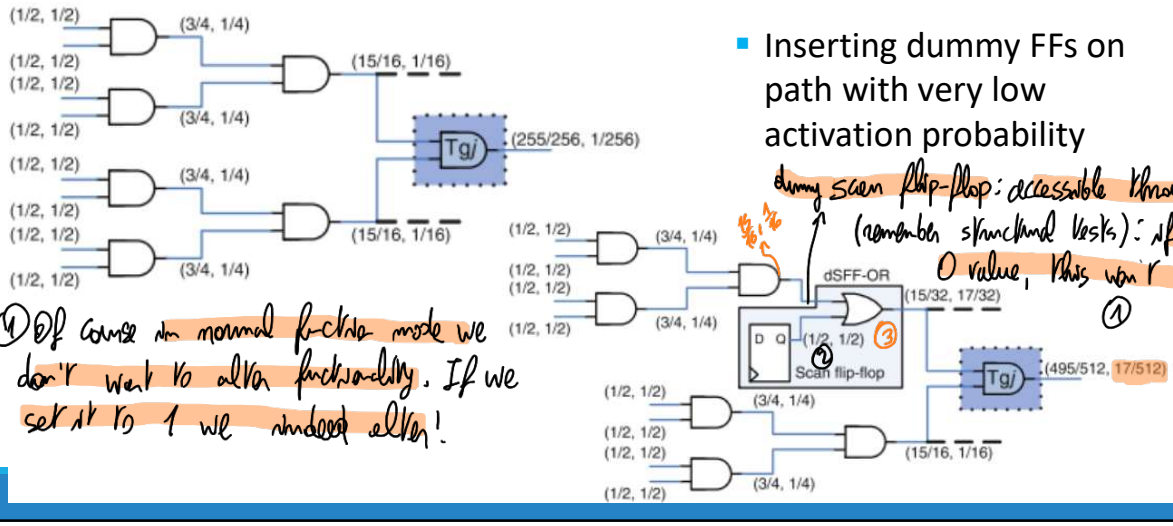
12

Increasing Probability of Partial/Full Activation



inputs have a 50% prob. of having 0/1. Of course, this and tree produces a 1 with 1/256 probability. We want a solution that allow us to mitigate this unbalance during testing.

Increasing Probability of Partial/Full Activation



- Inserting dummy FFs on path with very low activation probability

dummy scan flip-flop: accessible through scan chain (remember structural tests): if we scan a 0 value, this won't change functionality. ①

① Of course in normal functional mode we don't want to alter functionality. If we set it to 1 we indeed alter!

② During test phase, probability of having 1 or 0 is ½ - [ASSUMPTION] [we choose them]

③ To get probabilities for OR: $P_1(0) = P_{1,input1}(0) \cdot P_{1,input2}(0)$.

This increases probability to activate eventual trojan. We increased controllability for that node (we can also increase observability)

Increasing Probability of Partial/Full Activation

- Dummy scan flip-flops are inserted to control hard-to-excite nodes
- Usage:
  - Full activation: increase controllability
  - Power-based: generate switching activities
  - Delay-based: activate more paths to improve coverage

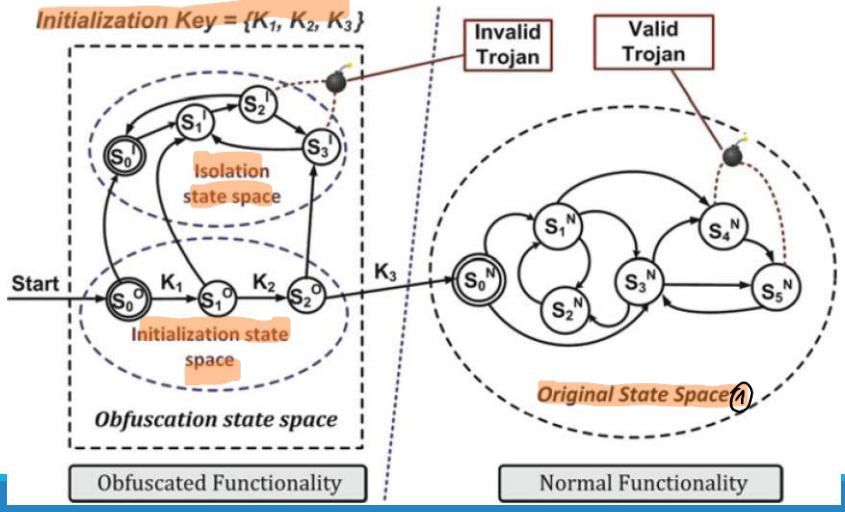
15

Trojan Prevention: Design Obfuscation

- The objective is **deterring attackers** from inserting Trojans inside the design
- Design obfuscation means that a design will be **transformed** into another one which is **functionally equivalent** to the original, but in which it is much **harder for attackers to obtain complete understanding** of the internal logic, making reverse engineering much more difficult to perform
- It **obfuscates** the **state transition function** to add an **obfuscated mode** on top of the original functionality (called **normal mode**)

17

Design Obfuscation



- Specified pattern is able to guide the circuit into its normal mode
- The **transition arc K3** is the only way the design can enter normal operation mode from the obfuscated mode

18

① But I start from an unreachable state space. To reach original one I need to apply a sequence of inputs to reach the ①. So if I don't have them, I end up in an isolated state space and remain there. If attacker insert a change that interact with isolation state space it won't affect functionality. Of course attacker is still able to insert trojan on ①, but harder.

Trojan Prevention: Built-In Self-Authentication

- To prevent Trojan insertion during physical design, **built-in self-authentication (BISA) technique** can be applied: all **unused spaces** in a circuit layout are filled with functional standard cells instead of non-functional filler cells
- **BISA** can **test the functionality of all functional filler cells** automatically with low overhead → BISA is able to prevent hardware Trojan insertion
- BISA's impact on the original circuit is negligible because there is no connection between the BISA circuit and the original circuit, and they do not work simultaneously

19

- When you design your system, in particular physical design, I know they are built so that we cannot put everything too close together (a need to do routing, and also close things might electrically interfere badly). What do I do with unused area? From electrical PoV a need to put something that creates physical and electrical continuity in the chip. Those are filler cells: non functional ones that are physically needed. So what do we do? We insert functional cells that in the original circuit are not used

So empty area is filled with those filler cells that can be exploited by attackers. So instead of filler cells a insert functional cells that don't interact with functionality of circuit, and the a test them to see if they have been modified or not. And those cells are realized in similar way of actual functional cells. So they are extremely difficult to detect by attacker. And since we test them, modifying them will be detected by our tests.
NOTE: BISA cells are combinational! Easier to test!

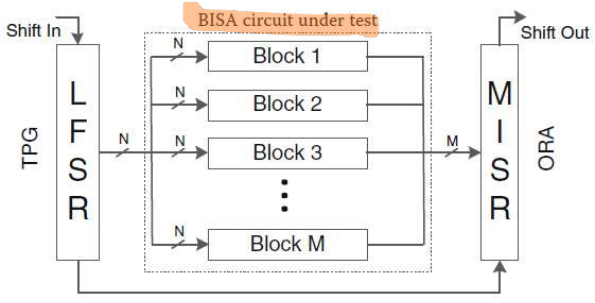
- BISA cells are similar to the ones we use for actual functional cells.
- Are those cells active during normal functionality? I can keep them deactivated, so no problem.

Trojan Prevention: Built-In Self-Authentication

- These **BISA cells** are connected together to form a **combinational circuit** that is independent from the original circuits
- By **testing** functions of the combinational circuit composed of **BISA** cells, designers would be able to **find out if these cells are modified** or not after fabrication
- **BISA cells** are the **same** as the **standard cells** the circuit uses, so **identifying these cells** will be **extremely difficult**, if not impossible

22

BISA Structure and Function



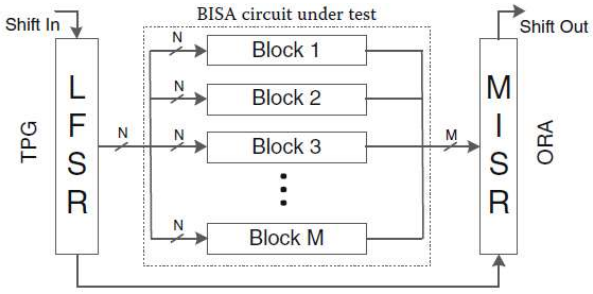
- To test the inserted filler cells with low overhead, a similar methodology to the **Logic Built-In Self-Test** (LBIST) system in VLSI testing is adapted
- The BISA, as in LBIST, is composed of three parts:
  - the BISA circuit under test,
  - the **Test Pattern Generator** (TPG), and
  - the **Output Response Analyzer** (ORA)
- The BISA circuit under test is composed of all BISA cells which are inserted into unused spaces during **physical design**

23

So what do I do then? I partition them in combinational blocks (easier to test small blocks). I can test them on chip with a LFSR. It is good because it generates random test patterns. So how do I check behaviour? I use a test compression circuit: a MISR: → the BIST circuit

after a bit of time of doing this, I read state of MISR to get information. Of course this introduces aliasing problems: I can have same output with different inputs. We will assume that MIST works well. So I load various outputs produced by BIST blocks, make MISR evolve for some time and I expect a certain behaviour. MISR: Multiple input signature register.

BISA Structure and Function

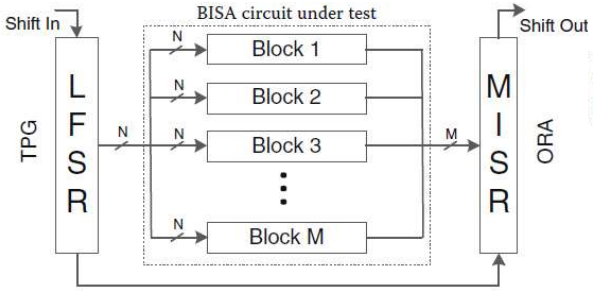


- In order to increase its test coverage, the BISA circuit is divided into a number of small combinational logic blocks, called **BISA blocks**
- The TPG generates test vectors that are shared by all BISA blocks
- The ORA will process the outputs of all BISA blocks and generate a signature

- TPG → linear-feedback-shift-register (LFSR)
- ORA → multiple-input-signature-register (MISR) (sometimes referred to as multiple-input-shift-register, or MISA: multiple-input-signature-analyser)

24

Operation modes in a design with BISA



	Normal mode	Authentication mode: Shift mode	Authentication mode: Test mode
Original circuit	Working	Idle	Idle
BISA circuit	Idle	Shift seed/signature	Testing BISA

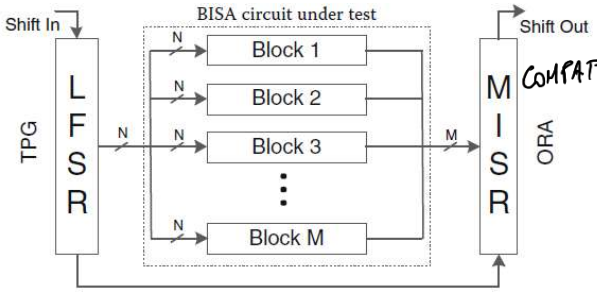
- Two operation modes:
  - Normal mode: the original circuit is working, but the BISA is completely shut down by **blocking clocks to LFSR and MISR** → BISA stays quiet and does not affect the original circuits → No dynamic power consumption, [Yes static power, but depends on area only]
  - Authentication mode: slow clock is provided to BISA

clock gating

10

25

Operation modes in a design with BISA



- Test mode (during Authentication):
  - LFSR generates N-bit test patterns and applies them to all BISA blocks at every clock cycle [Pseudorandom test patterns]
  - Each BISA block outputs one bit → the MISR will receive a total of M bits from M blocks
  - When a sufficient number of test patterns are applied, the BISA circuits is fully tested

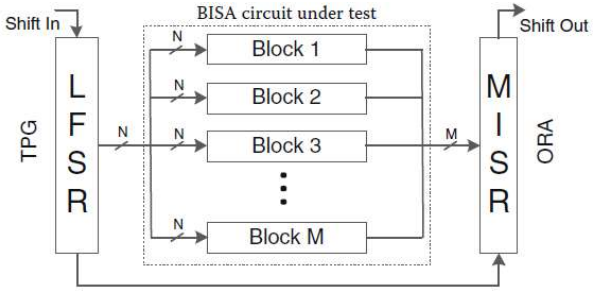
COMPATTATORE

① I apply a predetermined number of tests (sufficiently high to have a wise MISR analysis)

- The value stored in the MISR is the signature generated from the M BISA blocks' responses

26

Operation modes in a design with BISA



- Shift mode (during Authentication):
  - LFSR and MISR are connected in a series in the shift mode

    → the signature in MISR can be shifted out

    → the seed for LFSR can be shifted.

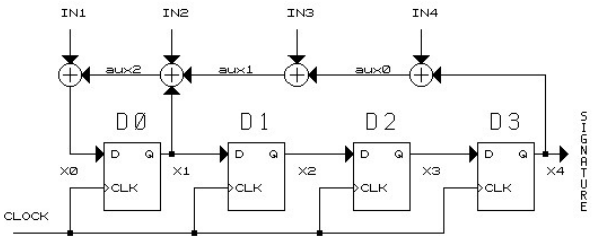
    new seed for another test

- Comparing the obtained signature with the correct signature from simulation shows whether the BISA structure has been tampered

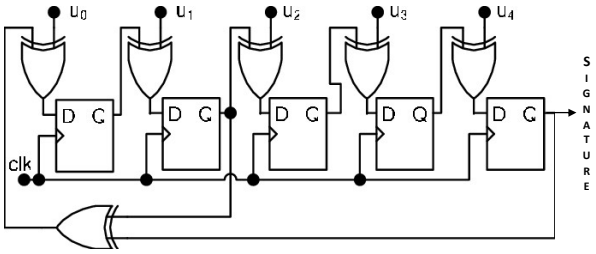
note: connection because you want to put unlikely values in MISR

27

Example of MISR Structure



Should misr output be unique ideally given inputs?



28

Additional References and Readings

1. M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," in the IEEE Design & Test of Computers, vol.27, no.1, pp. 10–25, 2010
2. X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting Malicious Inclusions in Secure Hardware: Challenges and Solutions," in Proc. of the IEEE International Workshop on Hardware-Oriented Security and Trust (HOST2008), pp. 15–19, 2008
3. Y. Jin and Y. Makris, "Hardware Trojan Detection using Path Delay Fingerprint," in Proc. Of the IEEE International Workshop on Hardware-Oriented Security and Trust (HOST 2008), pp. 51–57, 2008
4. Y. Alkabani and F. Koushanfar, "Consistency-Based Characterization for IC Trojan Detection," in Proc. of the International Conference on Computer-Aided Design (ICCAD09), pp. 123–127, 2009.
5. F. Wolff, C. Papachristou, S. Bhunia, R.S. Chakraborty, "Towards Trojan Free Trusted ICs: Problem Analysis and Detection Scheme," in Proc. of the Design, Automation and Test in Europe (DATE08), pp. 1362–1365, 2008.
6. S. Jha and S.K. Jha, "Randomization Based Probabilistic Approach to Detect Trojan Circuits," in Proc. of the High Assurance Systems Engineering Symposium (HASE08), pp. 117–124, 2008.
7. M. Banga and M. S. Hsiao, "A Region based Approach for the Identification of Hardware Trojans," in Proc. of the IEEE International Workshop on Hardware-Oriented Security and Trust (HOST2008), pp. 40–47, 2008
8. X. Zhang and M. Tehranipoor, "Case Study: Detecting Hardware Trojans in Third-Party Digital IP Cores," in Int. IEEE Hardware-Oriented Security and Trust (HOST), 2011.

12

31

Additional References and Readings

9. R. S. Chakraborty, S. Bhunia, "HARPOON: An Obfuscation-Based SoC Design Methodology for Hardware Protection", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (Volume: 28, Issue: 10, Oct. 2009)
10. R. S. Chakraborty, S. Bhunia, "Security Against Hardware Trojan Attacks Using Key-Based Design Obfuscation", Journal of Electronic Testing 27(6):767-785, Dec. 2011.
11. K. Xiao; M. Tehranipoor, "BISA: Built-in self-authentication for preventing hardware Trojan insertion," in 2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 2013
12. K. Xiao, D. Forte, M. Tehranipoor, "A Novel Built-In Self-Authentication Technique to Prevent Inserting Hardware Trojans," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (Volume: 33, Issue: 12, Dec. 2014)
13. S. Bhunia, M. S. Hsiao, M. Banga, S. Narasimhan, "Hardware Trojan Attacks: Threat Analysis and Countermeasures", Proceedings of the IEEE (Volume: 102, Issue: 8, Aug. 2014)