

# Fault Tolerant Implementations of Delay-based Physically Unclonable Functions on FPGA

*Presented by:*

**Sarani Bhattacharya**  
**SEAL, IIT Kharagpur**

**Authors:** Durga Prasad Sahoo, Sikhar Patranabis, Debdeep Mukhopadhyay, and Rajat Subhra Chakraborty  
Secured Embedded Architecture Laboratory (SEAL)  
Indian Institute of Technology Kharagpur, India

Fault Diagnosis and Tolerance in Cryptography, 2016

# Objective of Talk

- ▶ Overview of **laser fault attacks** on FPGA-based Physically Unclonable Functions (PUFs) implementations ([Tajik et al. in FDTC-2015](#)) and its consequences:
  - ▶ Accelerate the modeling attack
  - ▶ Entropy reduction
  
- ▶ **Our Contributions:**
  - ▶ Inclusion of additional logic to make the faults detectable — ***Fault Detection***
  - ▶ Recovery of PUF instance from the faulty state — ***Fault Recovery***
  - ▶ As case studies, APUF, XOR APUF and ROPUF will be discussed

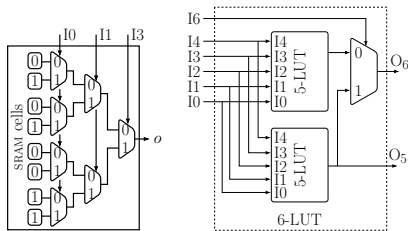
# Outline

- 1 Overview: Laser Fault Attack on SRAM FPGA and PUF
- 2 Laser Fault on XOR APUF and Its Detection
- 3 Laser Fault on ROPUF and Its Detection
- 4 Fault Recovery Schemes

- 1 Overview: Laser Fault Attack on SRAM FPGA and PUF
- 2 Laser Fault on XOR APUF and Its Detection
- 3 Laser Fault on ROPUF and Its Detection
- 4 Fault Recovery Schemes

## Logic Realization in SRAM FPGA

- ▶ Look-Up Tables (LUTs) in FPGA are used to implement any Boolean function
- ▶ A  $k$ -LUT ( $k$  inputs LUT) is composed of  $2^k$  SRAM cells and 2:1 MUX tree. See Fig. (a)

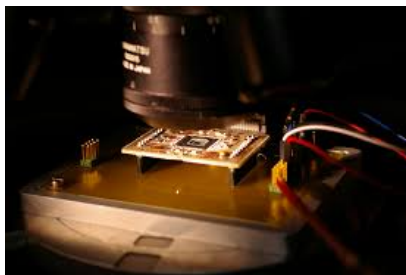


(a) Internal view of 3-LUT      (b) 6-LUT in Xilinx

- ▶ Dual-output 6-LUT in Xilinx 7-series FPGA is shown in Fig. (b)

## Laser Fault-injection on SRAM FPGA

- ▶ Objective is to modify the content of SRAM cells associated with an LUT
  - ▶ It results the LUT with modified functionality — called as **fault**



**Figure:** Laser fault-injection setup [Tajik et al., FDTC-2015]

- ▶ Laser Pulse can be used to read and modify the content of SRAM cells in FPGA
- ▶ Photonic emission analysis through IC back-side is used to identify the target components

## Traditional Fault Tolerance Approaches— not applicable in PUF

- ▶ A silicon Physically Unclonable Function (PUF) is a mapping

$$\gamma : \{0, 1\}^n \longrightarrow \{0, 1\}^k$$

where the  $k$ -bit output, known as **response**, are unambiguously identified by both the  $n$ -bit input, known as **challenge**, and the **unclonable**, **unpredictable** but **repeatable** instance specific manufacturing variations.

## Traditional Fault Tolerance Approaches— not applicable in PUF

- ▶ A silicon Physically Unclonable Function (PUF) is a mapping

$$\gamma : \{0, 1\}^n \longrightarrow \{0, 1\}^k$$

where the  $k$ -bit output, known as **response**, are unambiguously identified by both the  $n$ -bit input, known as **challenge**, and the **unclonable**, **unpredictable** but **repeatable** instance specific manufacturing variations.

- ▶ Two important properties of PUF:
  - ▶ **Randomness**: PUF outputs are random, and thus, there are no such reference outputs to detect faults
  - ▶ **Uniqueness**: Instances of a PUF design are expected to be unique



## Traditional Fault Tolerance Approaches— not applicable in PUF

- ▶ A silicon Physically Unclonable Function (PUF) is a mapping

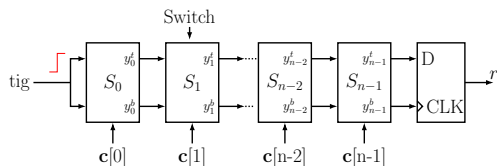
$$\gamma : \{0, 1\}^n \longrightarrow \{0, 1\}^k$$

where the  $k$ -bit output, known as **response**, are unambiguously identified by both the  $n$ -bit input, known as **challenge**, and the **unclonable**, **unpredictable** but **repeatable** instance specific manufacturing variations.

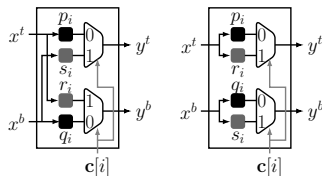
- ▶ Two important properties of PUF:
  - ▶ **Randomness**: PUF outputs are random, and thus, there are no such reference outputs to detect faults
  - ▶ **Uniqueness**: Instances of a PUF design are expected to be unique
- ▶ Traditional fault tolerance approach not applicable:
  - ▶ Spatial redundancy (**infeasible due uniqueness property**)
  - ▶ **In context of PUF, design-specific fault tolerance scheme is required. Next we discuss a few such schemes.**

- 1 Overview: Laser Fault Attack on SRAM FPGA and PUF
- 2 Laser Fault on XOR APUF and Its Detection**
- 3 Laser Fault on ROPUF and Its Detection
- 4 Fault Recovery Schemes

# Arbiter PUF (APUF)<sup>1 2</sup>



(a) APU architecture



(b) Classic SW (c) PDL SW

Figure: Arbiter PUF architecture with two types of switches

- ▶ A given challenge  $\mathbf{c} \in \{0, 1\}^n$  forms a pair of (ideally) symmetrically laid-out paths (**zero nominal delay difference**)
- ▶ Applied 'tig' signal propagates along these paths
- ▶ The response to  $\mathbf{c}$  is determined by the **unique** and **random** delay difference  $\Delta_{\mathbf{c}}$  of path-pair

<sup>1</sup> D. Lim, "Extracting Secret Keys from Integrated Circuits," Master's thesis, MIT, USA, 2004

<sup>2</sup> M. Majzoobi, F. Koushanfar, and S. Devadas, "FPGA PUF using Programmable Delay Lines," in *IEEE International Workshop on Information Forensics and Security (WIFS)*, Dec 2010, pp. 1–6

# XOR PUF and Its Modeling

- ▶ Output  $o$  of  $x$ -XOR APUF is:

$$o = r_0 \oplus r_1 \oplus \dots \oplus r_{x-1}$$

- ▶ **Security assumption:** outputs  $r_0, \dots, r_{x-1}$  of APUFs are not accessible to the attacker

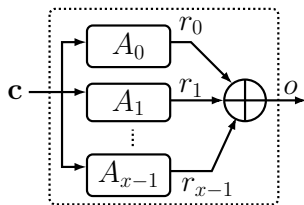


Figure:  $x$ -XOR APUF

- ▶ In LR-based modeling, no. of parameters to be learned is  $x(n + 1)$  for  $x$ -XOR APUF with  $n$ -bit challenge
- ▶ If  $x \geq 6$ , LR-based modeling is computationally infeasible. This bound is based on the serial implementation<sup>1</sup> of LR

<sup>1</sup>J. Sölter, "Cryptanalysis of Electrical PUFs via Machine Learning Algorithms," Master's thesis, Technische Universität München, 2009

# Fault-assisted Modeling of XOR APUF<sup>1</sup>

Summary of fault-assisted modeling attack published in FDTC-2015:

▶ **Adversary's Objective:**

- ▶ Modeling of  $x$ -XOR APUF is performed using the models of individual APUF
- ▶ Achieving modeling of  $x$ -XOR APUF with **linear time and data complexities**

▶ **Adversary's Task:**

- ▶ Get access to APUFs' outputs through XOR gate output
- ▶ **Laser based fault-injection can modify a XOR APUF circuit such that it behaves like  $i$ th APUF ( $i = 0, \dots, k - 1$ ) for a time interval**
- ▶ This makes the modeling of individual APUF feasible

---

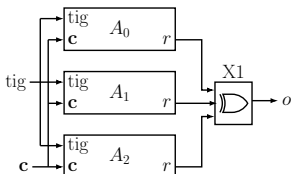
<sup>1</sup>S. Tajik, H. Lohrke, F. Ganji, J. P. Seifert, and C. Boit, "Laser Fault Attack on Physically Unclonable Functions," in *12th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, 2015

# Fault-assisted Modeling of XOR APUF (contd.)

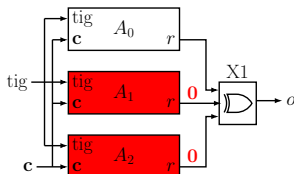
## ▶ Target components and fault-injection approach:

### ▶ Attack-I: APUF Switching Stage

- ▶ Modify the LUTs of last switch  $S_{n-1}$  APUF such that  $D$  inputs of D-FF (arbiter logic) is always '0'
- ▶ Perform this modification for all  $x - 1$  APUFs except the target APUF
- ▶ Thus, XOR APUF output is the same as the output of fault-free APUF
- ▶ Restart the circuit and repeat this for other APUFs



(a) 3-XOR APUF



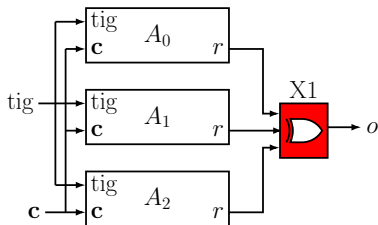
(b) 3-XOR APUF with faulty APUFs

# Fault-assisted Modeling of XOR APUF (contd.)

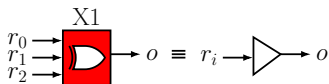
## ▶ Target components and fault-injection approach:

### ▶ Attack-II: XOR Logic in XOR APUF

- ▶ Modify the XOR logic such that it behaves like a buffer circuit for only one of its inputs
- ▶ Output of XOR APUF is now the same as one of its APUFs
- ▶ Repeat above steps for all  $x$  APUF instances to collect their CRPs

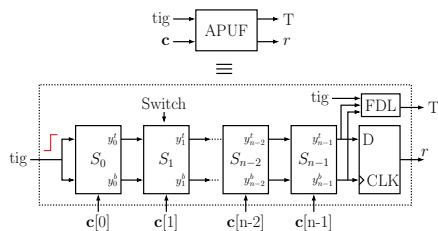


(c) 3-XOR APUF with faulty XOR



(d) Faulty XOR works as buffer for  $i$ -th input

# Fault Detection in APUF



**Table:** 3-input FDL

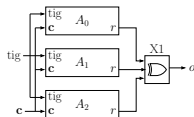
Inputs		Output T
tig	$y_{n-1}^t$ $y_{n-1}^b$	
0	0 0	1
1	1 1	1
x	0 1	0
x	1 0	0

**Figure:** APUF with fault detection logic (FDL)

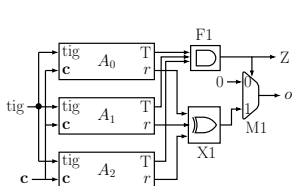
- ▶ If 'tig' is either 0 or 1, that value should be propagated to 'D' and 'CLK' inputs of D-FF in fault-free APUF
- ▶ Modification in switch  $S_{n-1}$  due to laser fault-injection results in 'D=0' regardless of 'tig' signal value
- ▶ Fault detection logic (FDL) can detect this fault
- ▶ The output of each individual APUF circuit is correct iff  $T=1$
- ▶ Thus, one should sample APUF response when  $T=1$



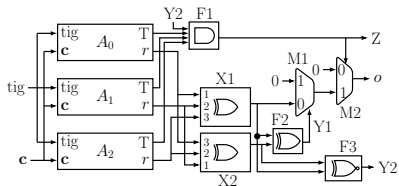
# Fault Detection in XOR APUF



(a) 3-XOR APUF



(b) Countermeasure- Attack-I



(c) Countermeasure- Attack-I & II

Figure: 3-XOR APUF with fault detection option.

- ▶ XOR APUF output is correct if  $Z=1$
- ▶ Sampling of XOR PUF response should be done when  $Z=1$

## Note

- ▶ Unlike PUF instances, fault detection logic (FDL) circuits are **deterministic**
- ▶ FDL circuits can be replicated to make them more robust against laser-faults
- ▶ PUF instances **cannot be replicated** due to its unique and instance-specific behavior

- 1 Overview: Laser Fault Attack on SRAM FPGA and PUF
- 2 Laser Fault on XOR APUF and Its Detection
- 3 Laser Fault on ROPUF and Its Detection**
- 4 Fault Recovery Schemes

# ROPUF and Fault Attack

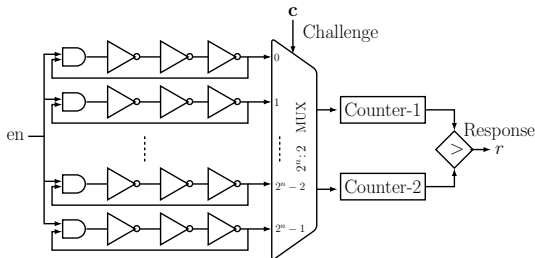
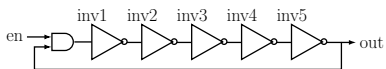


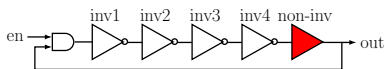
Figure: Ring Oscillator PUF

- ▶ ROPUF exploits a pair of ROs to generate a response bit
- ▶ Attacker might attempt to modify a RO of a pair of ROs to a **non-oscillating loop**
- ▶ Output corresponding to the RO pair would be biased
- ▶ **It results a ROPUF with reduced entropy**

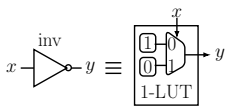
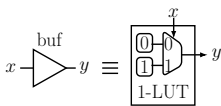
# Laser Fault-injection on RO



(a) RO



(b) Faulty RO

(c) Inverter using  
1-LUT

(d) Buffer using 1-LUT

- ▶ Each stage of RO can be realized using a 1-LUT
- ▶ LUT content of each stage is identical
- ▶ Attacker can modify the LUT content such that one inverting stage of RO becomes non-inverting
- ▶ Modified RO does not oscillate as there are even number of inverting stages

## Fault Detection in RO

- ▶ RO with fault detection logic:

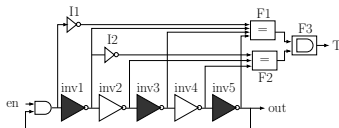
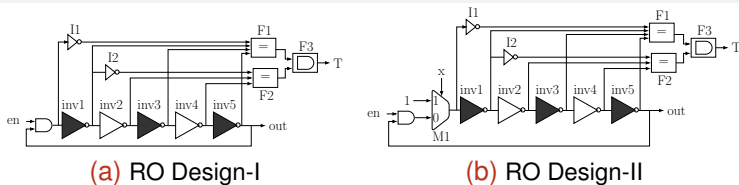


Figure: RO Design-I

- ▶ In each period of oscillation, odd stages produce the same output if there are no modifications in the odd stages. Likewise, it also happens for even stages.
- ▶ We incorporate two equality checking logic F1 and F2
- ▶ Inverters I1 is used to decide what are the expected outputs of odd stages of RO. I2 is used for even stages
- ▶ 'T=1' implies both F1 and F2 output 1s, and RO is fault-free for that oscillation period

## Fault Detection in RO (contd.)



**Figure:** RO with two different fault detection circuits.

- ▶ In **RO Design-I**, signal T becomes '1' at the end of each oscillation period for fault-free RO
- ▶ Thus, we can monitor this signal to detect the occurrence of fault, but this scheme is expensive as RO oscillates in MHz frequency
- ▶ Instead, we can check the RO before and after its frequency evaluation only to detect any faulty behavior
- ▶ Assuming that the adversary cannot revert the fault before the evaluation is finished
- ▶ This can be achieved by **RO Design-II** with an additional MUX (M1) with control signal 'x'

## Fault Detection in RO (contd.)

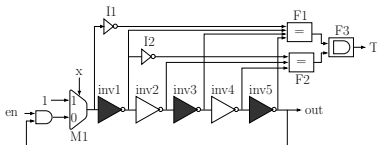


Figure: RO Design-II

- ▶ If 'x=0' and 'en=0', then input of inverter 'inv1' is set to '0', whereas for 'x=1' and 'en=0/1', input of 'inv1' is fixed to '1'
- ▶ These two assignments for signals 'en' and 'x' are required for fault detection
- ▶ For normal operation of RO required assignment is: 'x=0' and 'en=1'
- ▶ A ROPUF design would be robust if it employs the RO with the proposed fault detection logic
- ▶ A faulty RO can be recovered to its fault-free state using fault recovery schemes that will be discussed next



- 1 Overview: Laser Fault Attack on SRAM FPGA and PUF
- 2 Laser Fault on XOR APUF and Its Detection
- 3 Laser Fault on ROPUF and Its Detection
- 4 Fault Recovery Schemes**

# Fault Recovery Schemes for FPGA

- ▶ Broadly, two fault-recovery options are:

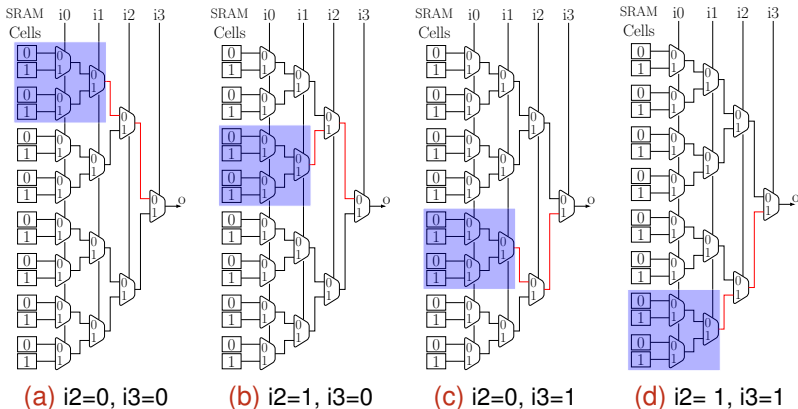
## 1 Rollback

- ▶ Objective here is to revert back to the original PUF instance with exactly the same timing behavior
- ▶ This can be achieved using either **configurable LUT** (CFGLUT5) or **dynamic partial reconfiguration** (DPR)
- ▶ Configurable LUT based recovery solution (**only 32 clocks**) is much faster than DPR

## 2 Random-sliding

- ▶ Objective in this case is to replace the faulty PUF instance with a different PUF instance with different timing behavior, **without extra LUTs**
- ▶ In case of authentication, verifier needs to maintain CRPs of all possible instances of PUF used to achieve fault tolerant feature

# Example of Random-sliding



**Figure:** A 2-variable Boolean function  $f(i_0, i_1)$  is implemented using 4-LUT. The circuit corresponding to  $f(i_0, i_1)$  shows four different timing behaviors for four different assignments for  $i_2 i_3 \in \{00, 10, 01, 11\}$ .

## Random-sliding for PUF Components

- ▶ Random-sliding feature can be used for APUF-based authentication:
  - ▶ An APUF switch utilizes a small portion of a LUT; thus, rest of the LUT part can be used to incorporate random-sliding feature in APUF switch
  - ▶ If the present configuration of APUF switch is found faulty, we can try with another random-slid configuration
- ▶ This features can also be used for other PUF designs
- ▶ Random-sliding is the fastest recovery scheme when it is applicable

# Conclusion

- ▶ Laser fault based modeling attack and entropy reduction attack can be a serious threat, although there are many physical constraints like IC depackaging
- ▶ Bare implementation of PUF is not enough to prevent physical attacks like laser fault attack
- ▶ Fault detection and recovery features should be included, and it should be a part of any future PUF design

**Thank You**