From Cryptography to Hardware: Analyzing Embedded Xilinx BRAM for Cryptographic Applications.

Shivam BHASIN
<shivam.bhasin@TELECOM-ParisTech.fr>

CRYPTARCHI 2013

Monday, June 24nd, 2013

Joint Work with

Sylvain Guilley, Annelie Heuser, Wei He and Jean-Luc Danger

Presentation Outline

Presentation Outline

Motivation

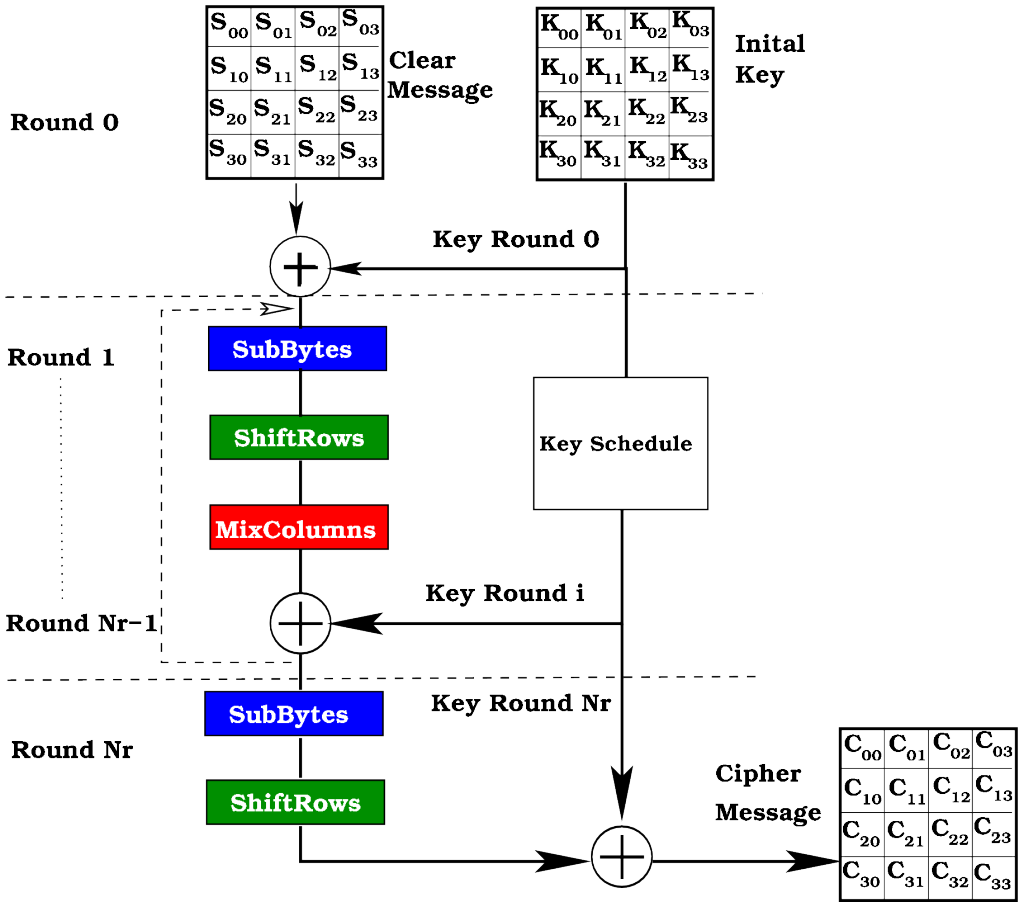
- Side-Channel Attacks (SCA) are known to break hardware implementations.
- Certain generic countermeasures are proposed to resist SCA.
- It also makes sense **to optimize** these countermeasures by using **target features.**
- Certain features of a device can be exploited to make countermeasures **compact and more robust.**

This talk covers ...

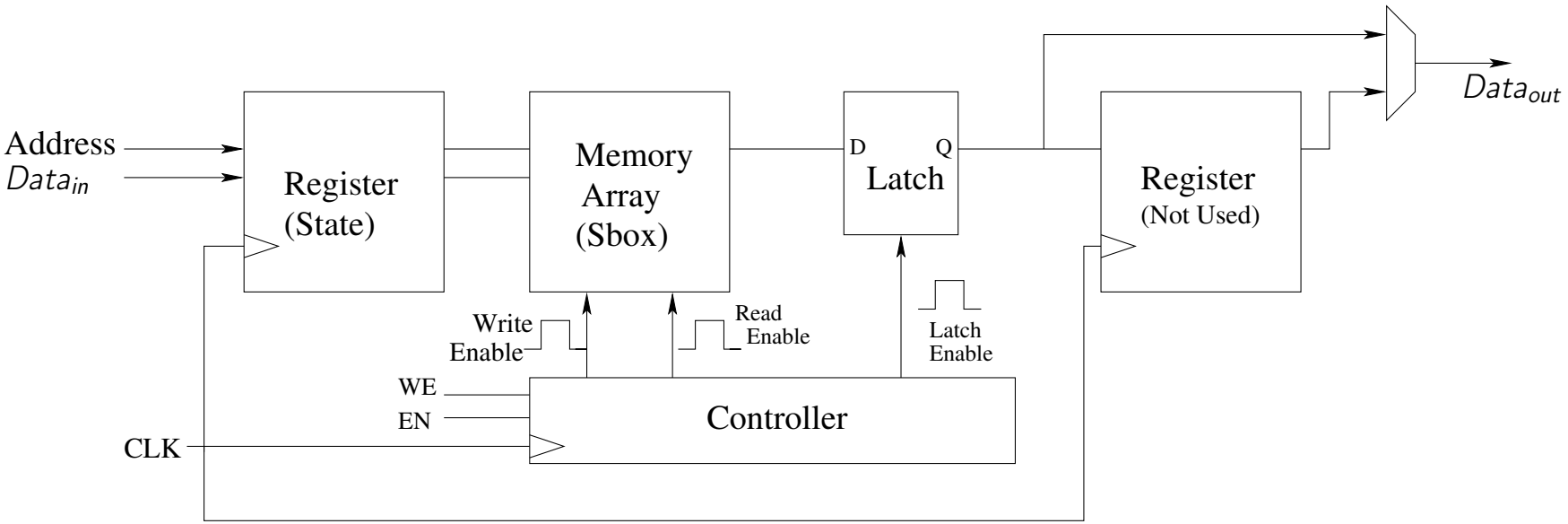
- Charaterizing different leakages from an FPGA.
- With special focus on FPGA Block RAM (BRAM) w.r.t. Side-Channel Analysis.
- Help designers to achieve elevated security by design.
- Use leakage characterstic to optimize countermeasures.

Advanced Encryption Standard (NIST FIPS 197)

- Designed by Joan Daemen and Vincent Rijmen.
- Substitution Permutation Network.
- Fixed Message size: 128 bits (4×4 bytes).
- Variable Key size : 128, 192, 256 bit.
- Variable round number: 10, 12, 14.



BRAM Architecture for Xilinx Devices



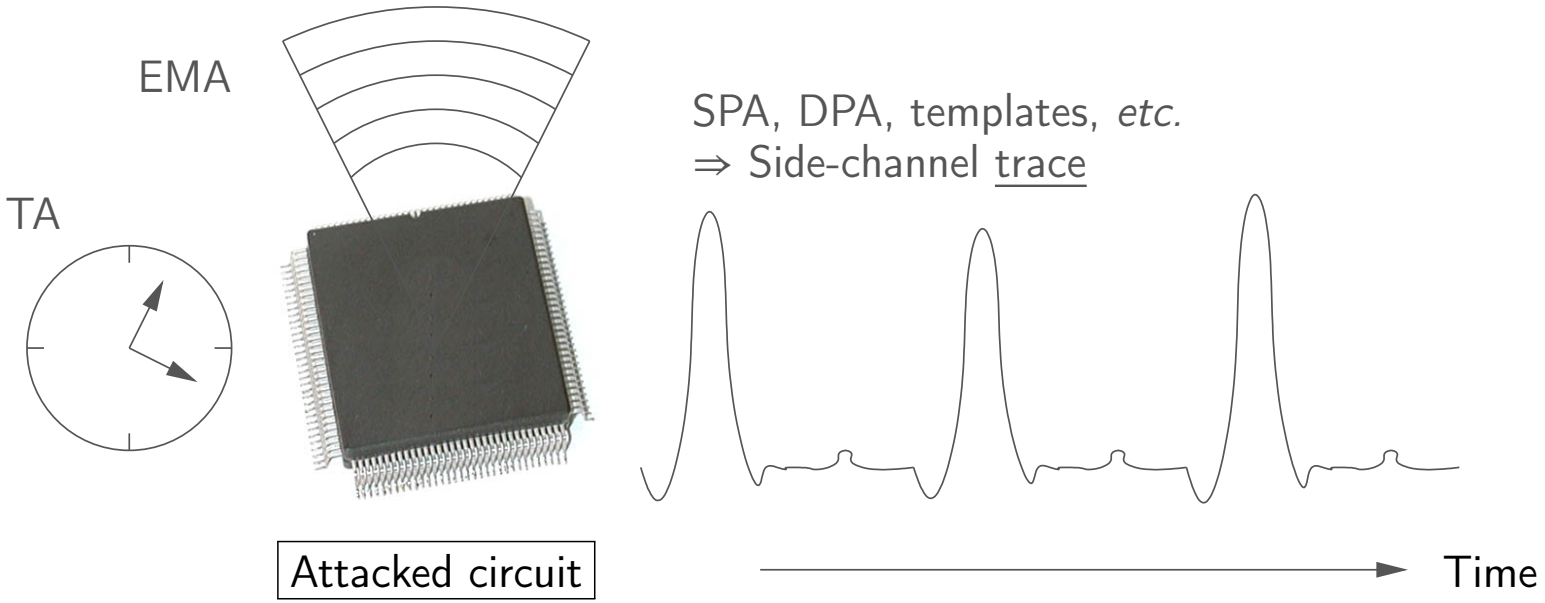
Prominent BRAM Features

BRAM Feature	Application to Cryptography
High Density RAM	To implement huge data[1]
Internal Register at input	To implement state register[1] Not connected to FPGA routing[2] No glitches[2]
Dual-Port Nature	Single block for multiple Sboxes[1]
Output Register	Available resource[1] To achieve better timing[2]
Reset	Better control on Output[1,2]
Hard Macro in $\leq 65nm$ CMOS	Low leakage power[2] Regular Structure[1,2]

In the table, 1 signifies an improvement in area or performance and 2 signifies an improvement in SCA resistance.

Presentation Outline

Side-Channel Attacks (SCA)



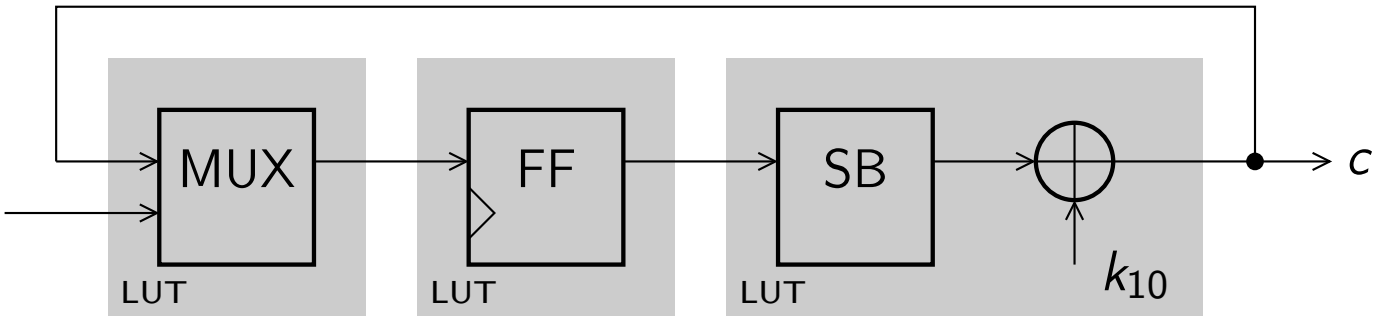
Background

- X is the measured leakage (from device).
- $\mathbb{L}$ is the modelled leakage (predicted).
- N is the measurement noise.
- CPA $\rho(X, \mathbb{L})$: It measures the correlation between X and $\mathbb{L}$ in presence of N.
- CPA finds maximum correlation between $\mathbb{L}$ and X for the correct predictions.

Evaluation Setup

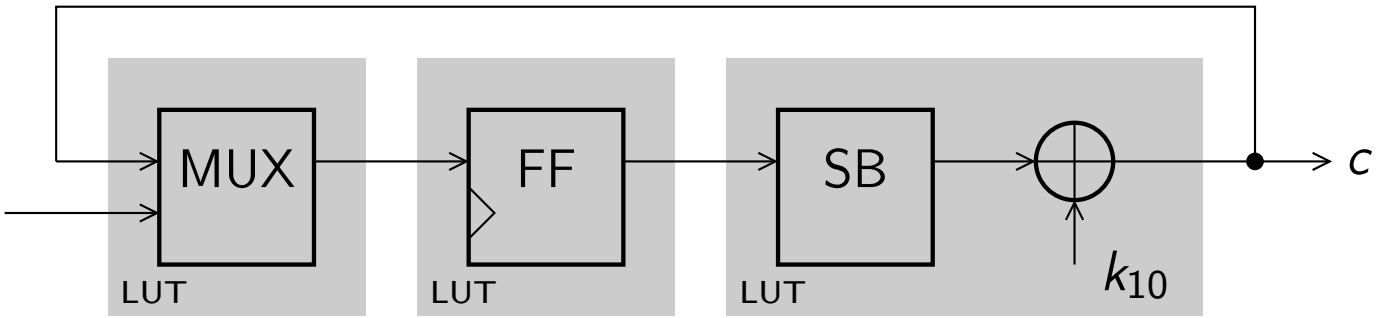
- Experiments performed on last round of various AES-128 implementations.
- AES running on an Xilinx Virtex-V FPGA of SASEBO-GII cards.
- SCA traces are acquired using an EM probe on a choosen decoupling capacitor.
- Attack on synchronous logic.
- In the following:
  - C is the Ciphertext, K_{10} is Last Round Key
  - SB and SB^{-1} are Sbox and InvSbox of AES.
  - ShiftRows is omitted for simplicity of representation.

Leakage Model: Architecture 1



Specs: State Register ⇒ FPGA Logic
Sbox ⇒ FPGA Logic

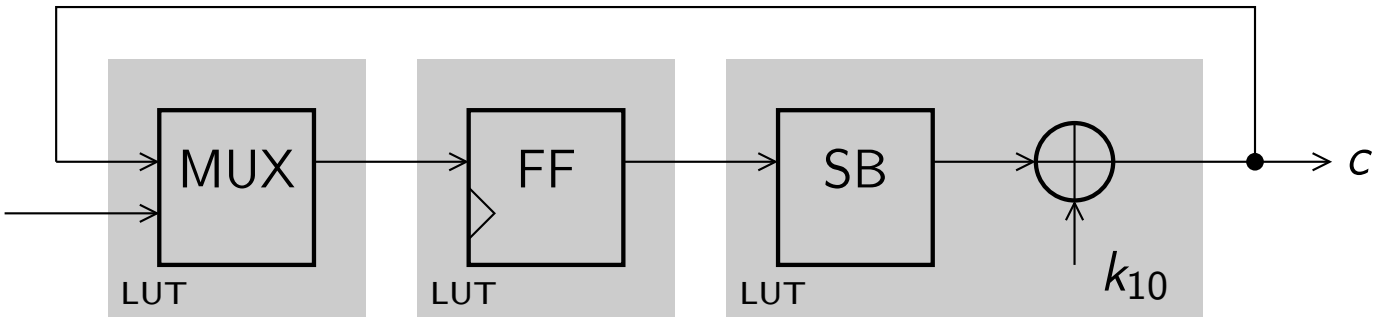
Leakage Model: Architecture 1



Specs: State Register ⇒ FPGA Logic
Sbox ⇒ FPGA Logic

Attack Target: Register $FF(M_{REG})$

Leakage Model: Architecture 1

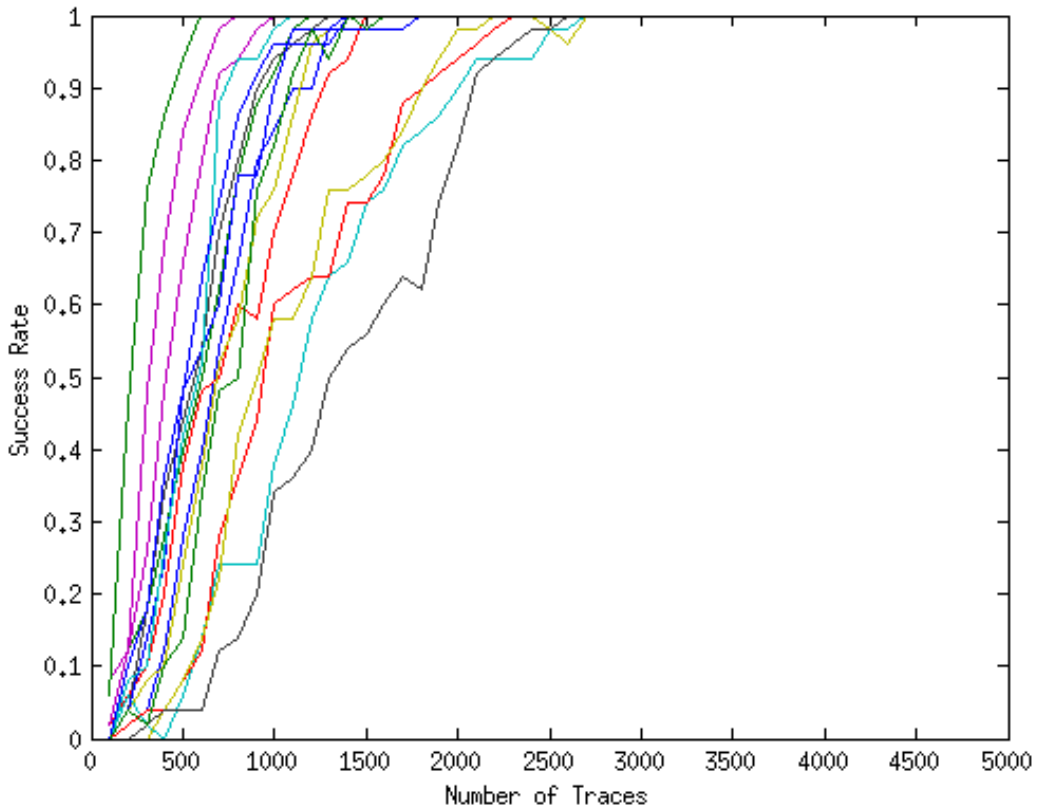


Specs: State Register ⇒ FPGA Logic
Sbox ⇒ FPGA Logic

Attack Target: Register $FF(M_{REG})$

$$M_{REG}: \quad \mathbb{L} = HW(C \oplus SB^{-1}(C \oplus K_{10})) + \mathbb{N}$$

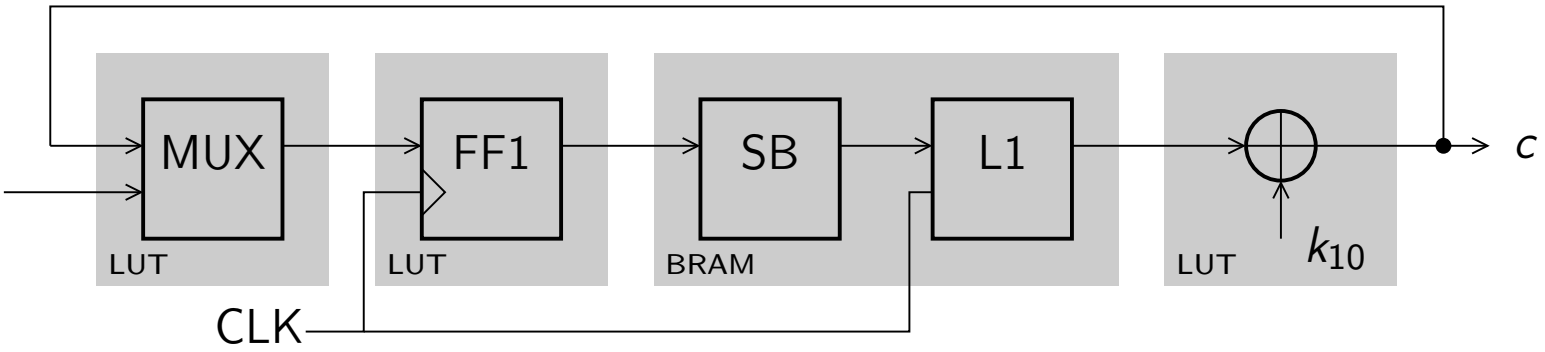
Leakage Model: Architecture 1



Success Rate.

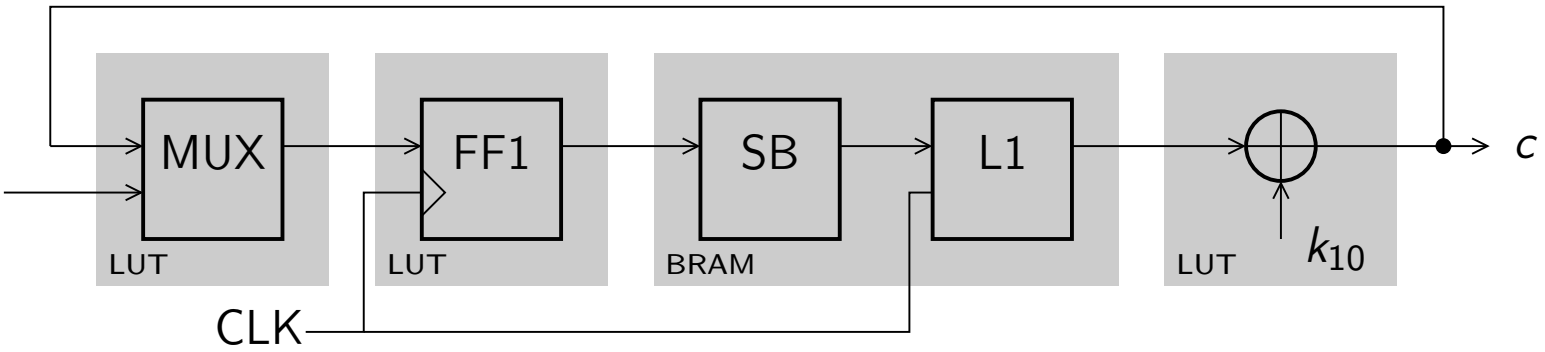
CPA @$FF(M_{REG})$: **1800 Traces**

Leakage Model: Architecture 2



Specs: State Register ⇒ FPGA Logic
Sbox ⇒ BRAM

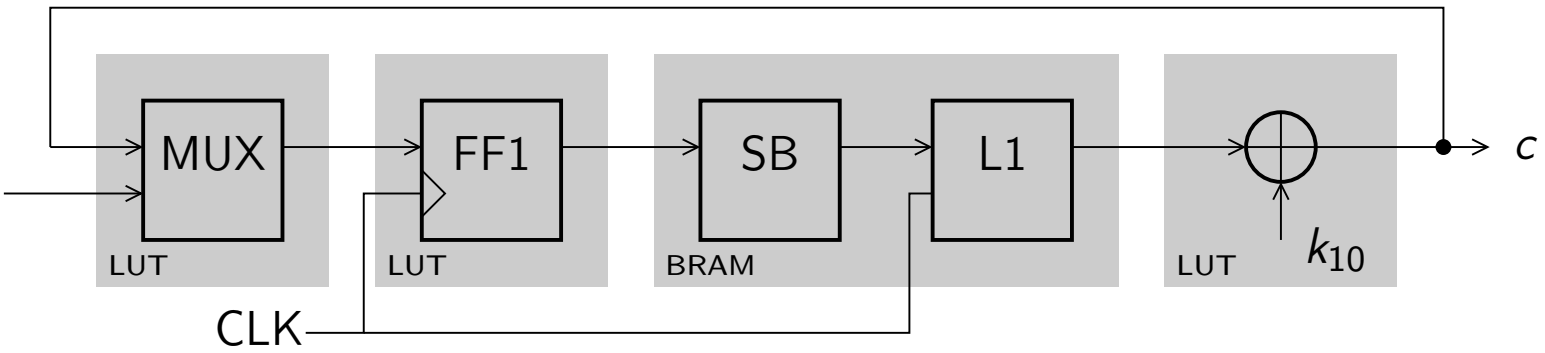
Leakage Model: Architecture 2



Specs: State Register ⇒ FPGA Logic
Sbox ⇒ BRAM

Attack Target: Register $FF1(M_{REG})$ and Latch $L1(M_{LATCH})$

Leakage Model: Architecture 2



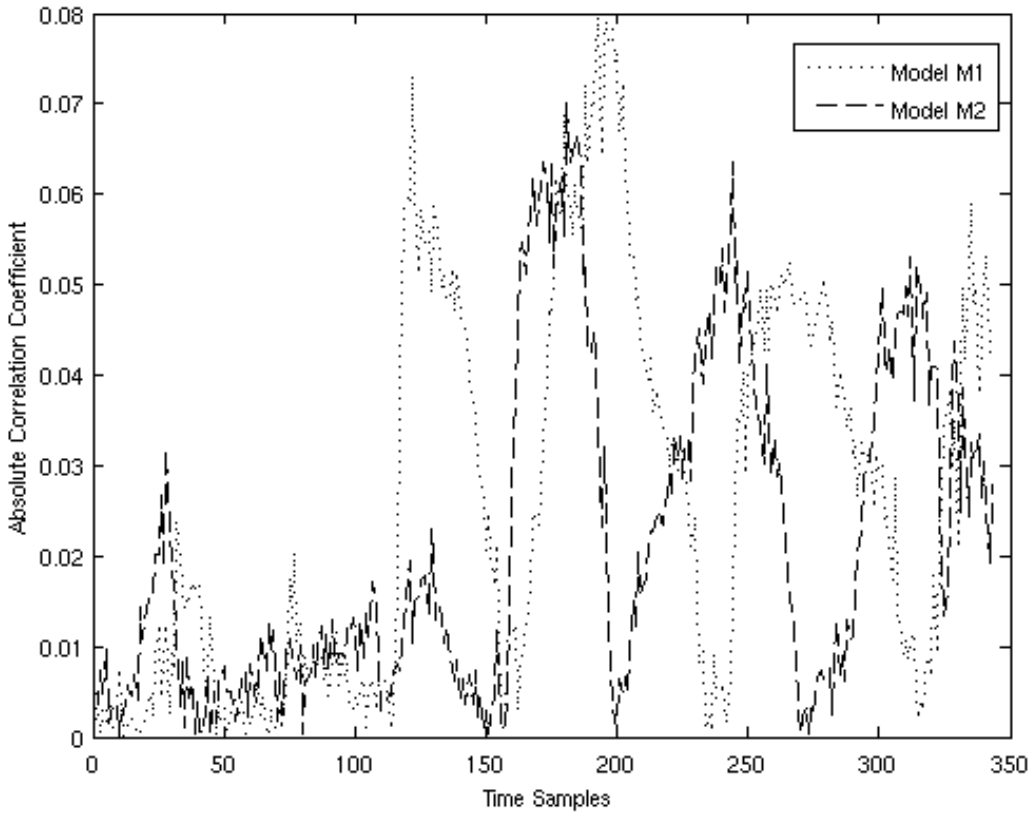
Specs: State Register ⇒ FPGA Logic
Sbox ⇒ BRAM

Attack Target: Register $FF1(M_{REG})$ and Latch $L1(M_{LATCH})$

$$M_{REG}: \quad \mathbb{L} = HW(C \oplus SB^{-1}(C \oplus K_{10})) + \mathbb{N}$$

$$M_{LATCH}: \quad \mathbb{L} = HW(SB(C) \oplus (C \oplus K_{10})) + \mathbb{N}$$

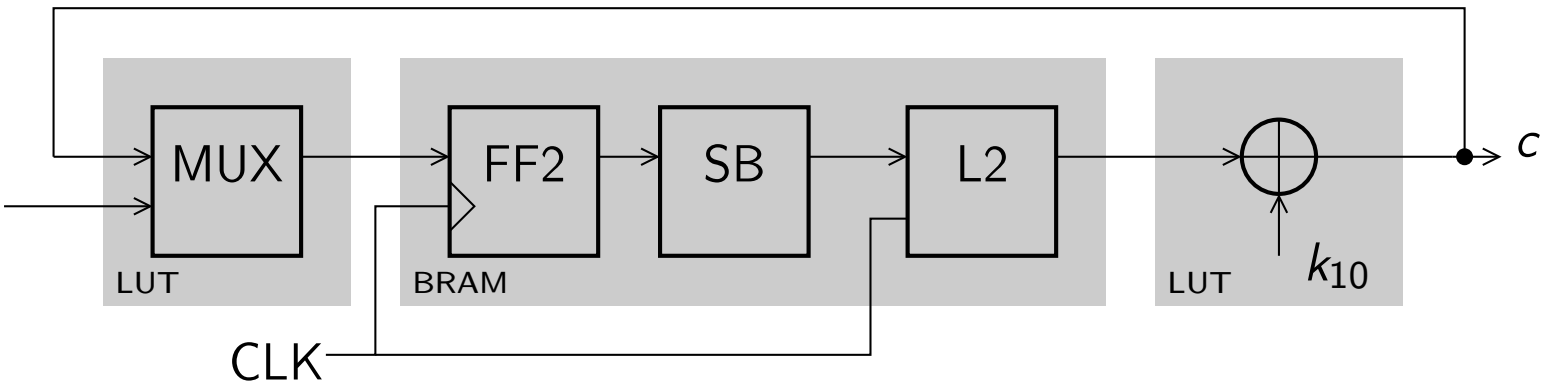
Leakage Model: Architecture 2



CPA @$FF1(M_{REG})$: **5000 Traces**

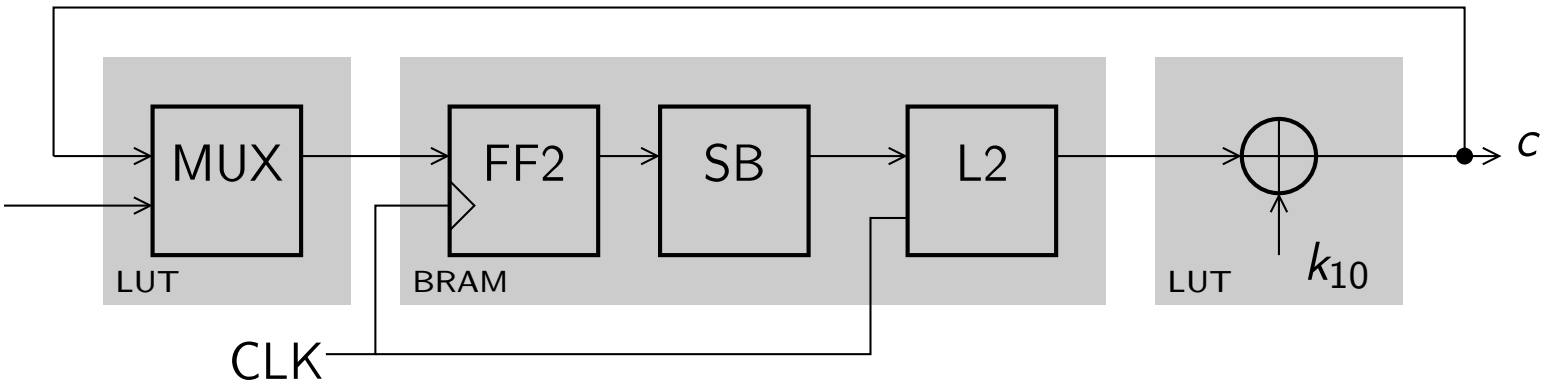
CPA @$L1(M_{LATCH})$: **4800 Traces** + 2^{16} **Brute force**

Leakage Model: Architecture 3



Specs: State Register ⇒ BRAM
Sbox ⇒ BRAM

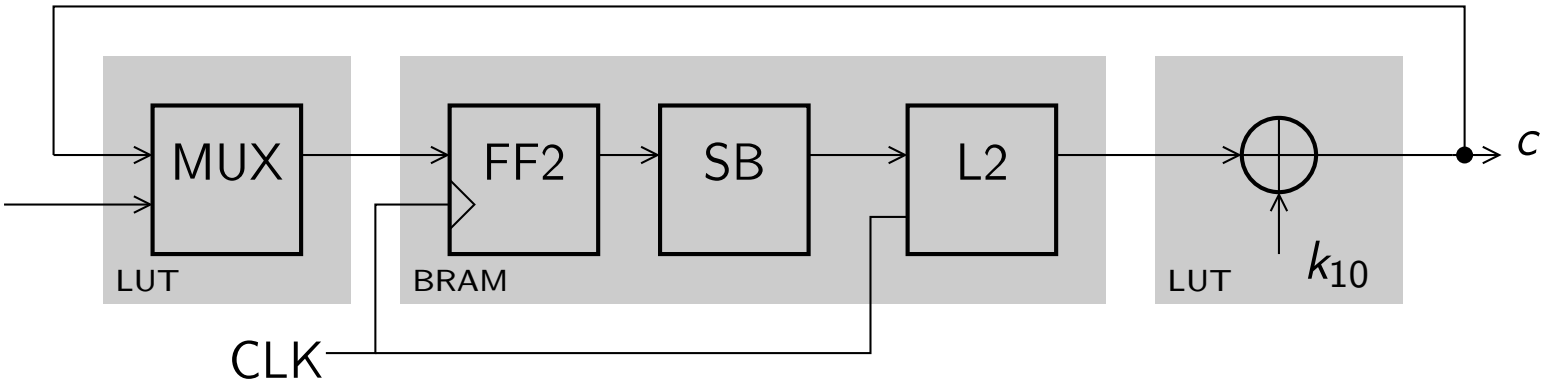
Leakage Model: Architecture 3



Specs: State Register ⇒ BRAM
Sbox ⇒ BRAM

Target: Register $FF2(M_{REG})$ and Latch $L2(M_{LATCH})$

Leakage Model: Architecture 3



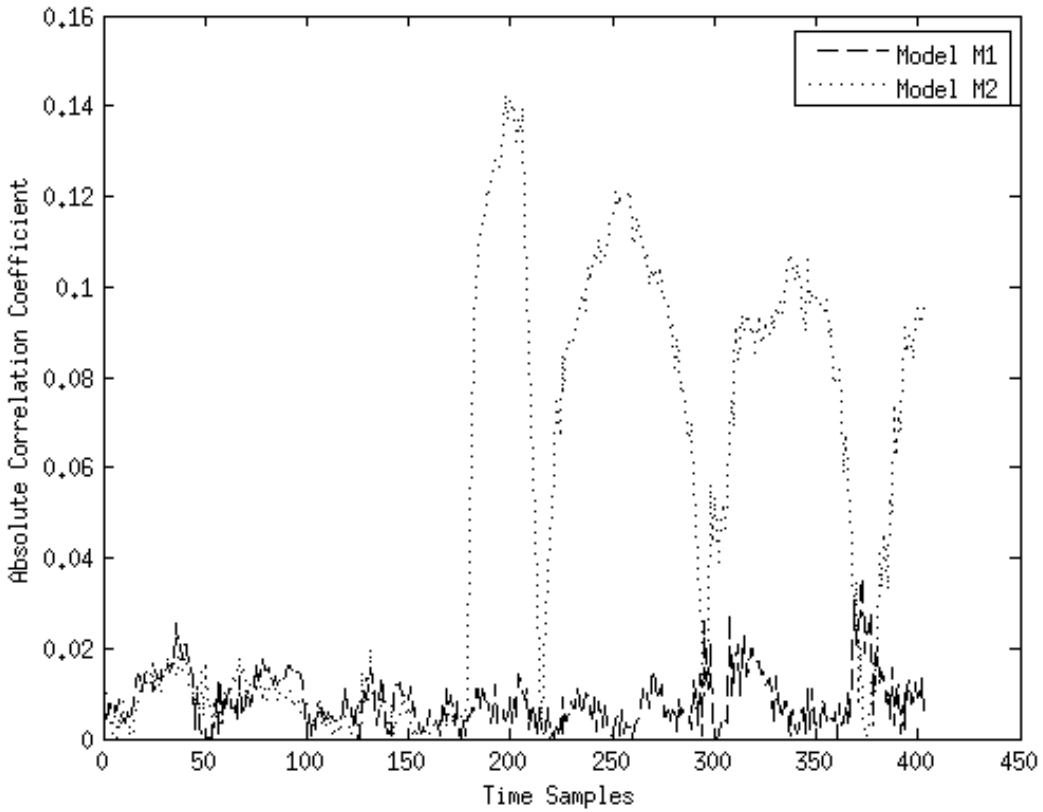
Specs: State Register ⇒ BRAM
Sbox ⇒ BRAM

Target: Register $FF2(M_{REG})$ and Latch $L2(M_{LATCH})$

$$M_{REG}: \quad \mathbb{L} = HW(C \oplus SB^{-1}(C \oplus K_{10})) + \mathbb{N}$$

$$M_{LATCH}: \quad \mathbb{L} = HW(SB(C) \oplus (C \oplus K_{10})) + \mathbb{N}$$

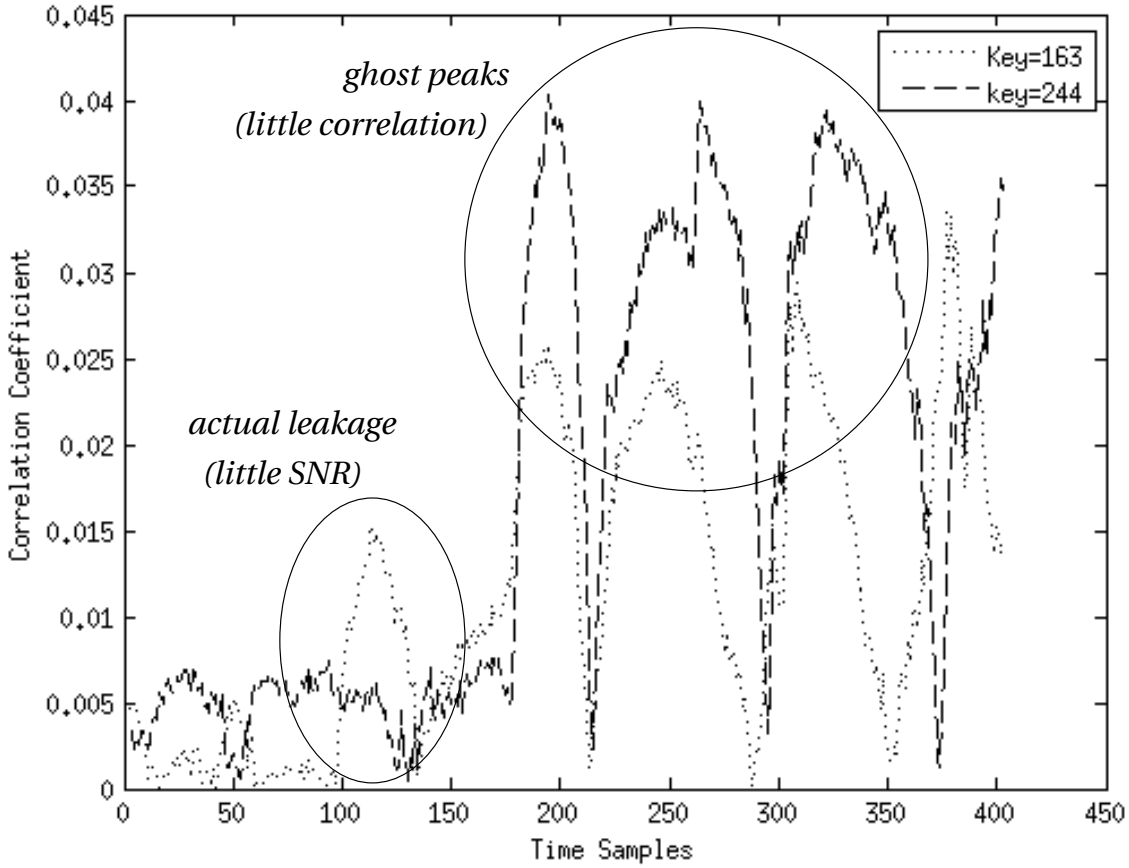
Leakage Model: Architecture 3



CPA @$FF2(M_{REG})$: ∞ **Traces**

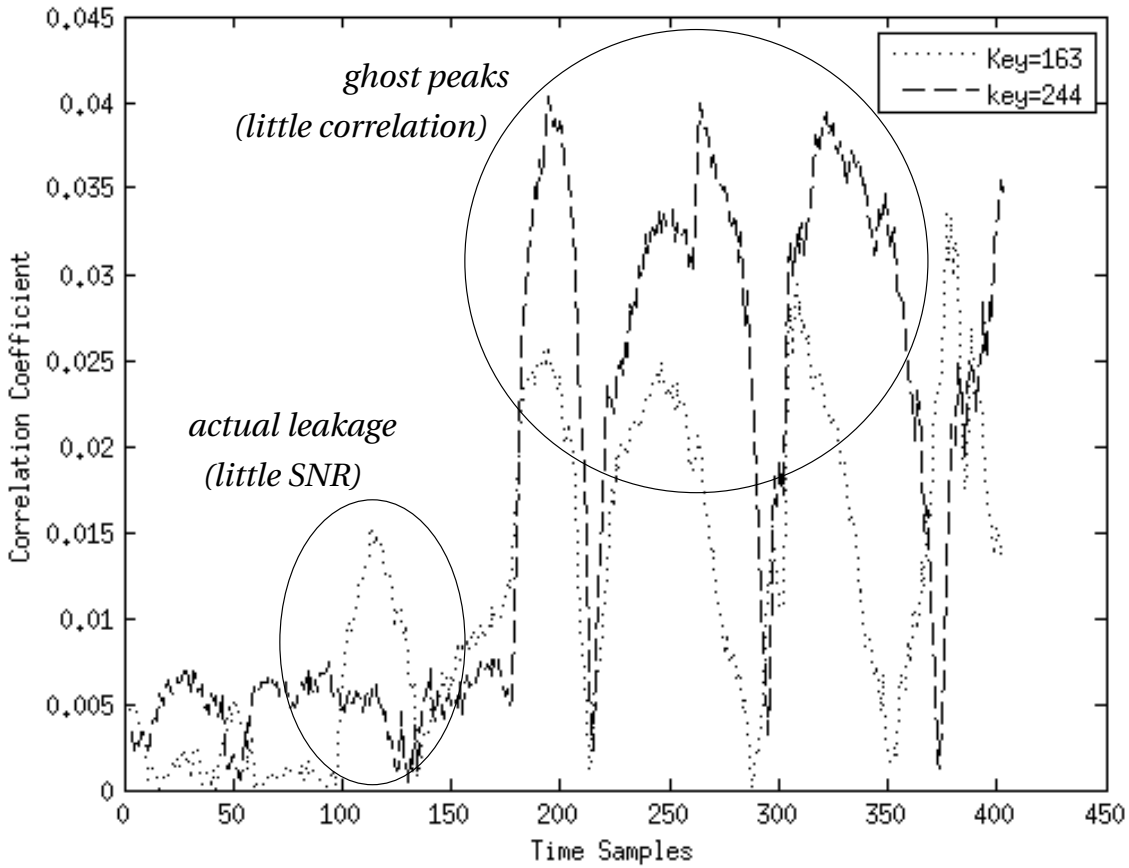
CPA @$L2(M_{LATCH})$: **2200 Traces** + 2^{16} **Brute force**

Does *FF*2 has no Leakage?



Correct Key (163) vs Guessed Key (244)

Does *FF*2 has no Leakage?



Correct Key (163) vs Guessed Key (244)

Localized CPA @$FF2(M_{REG})$: 88,000 Traces

Observations

SCA Evaluations		
Target	Register	Latch
Arch. 1	1800 (FF)	N.A.
Arch. 2	5000 ($FF1$)	4800 ($L1$)
Arch. 3	88,000* ($FF2$)	2200 ($L2$)

BRAM easily attackable at output register.

Why $L2$ leaks more than $L1$

- $L2$ timing path $\uparrow \Rightarrow$ Routing Buffers $\uparrow$.
- Routing Buffers $\uparrow \Rightarrow$ SNR $\uparrow$.
- $FF2$ timing path $\downarrow \Rightarrow$ Routing Buffers $\downarrow$.
- Same reason why $FF1$ leaks less than FF.

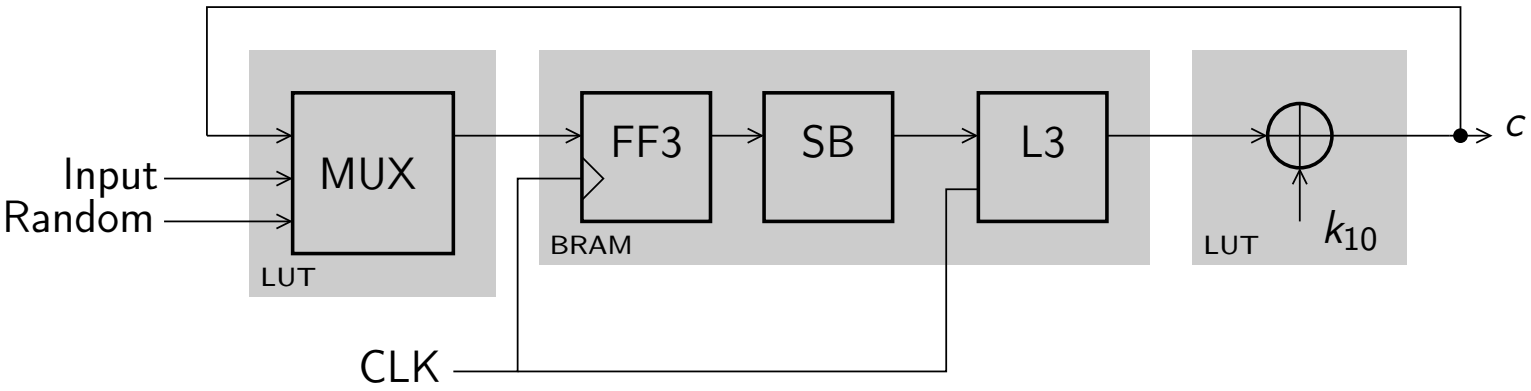
Observations

So We Know ...

- Arch. 3 seems best in cost and security.
- BRAM is easily attackable by changing the model.
- Attack on input register of BRAM is hard.
- Minimizing routing buffers can improve SCA resistance.

Presentation Outline

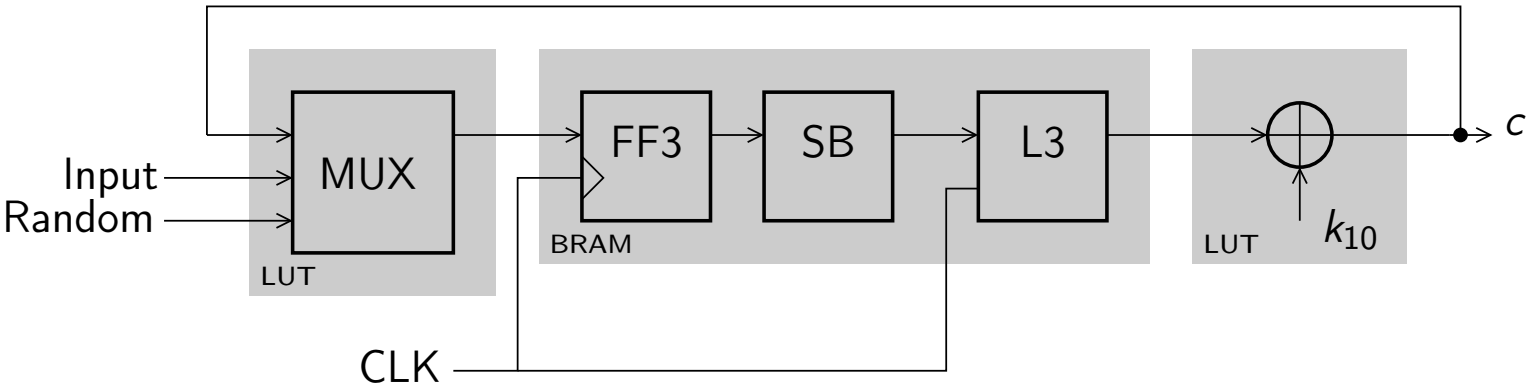
Protecting BRAM Register: Low-Cost Protection



Specs: Same as **Architecture 3 but ...**

*FF*3 stores a **Random** instead of C after last round.

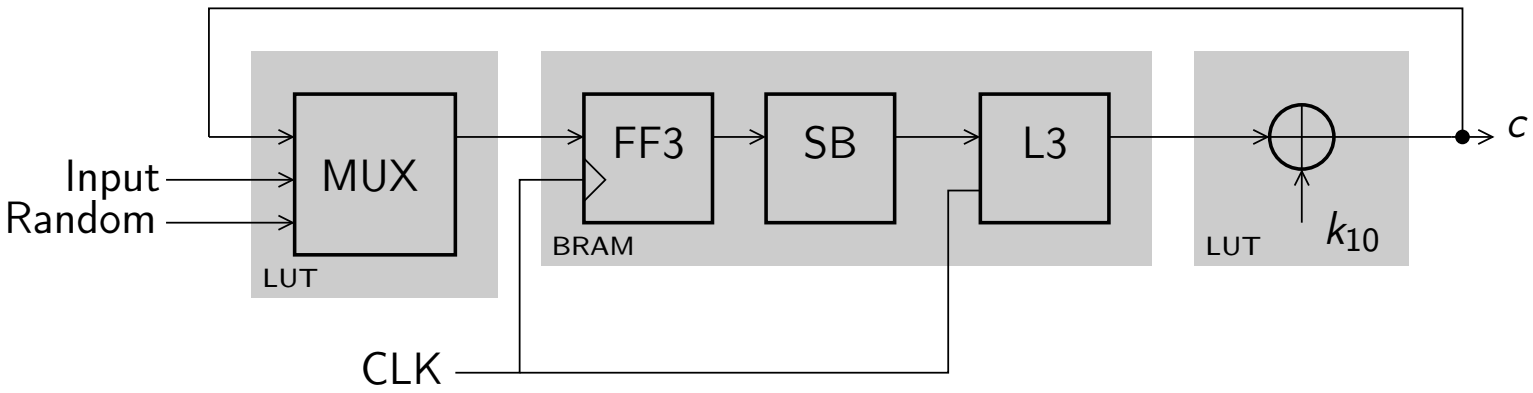
Protecting BRAM Register: Low-Cost Protection



Specs: Same as **Architecture 3 but ...**
$FF3$ stores a **Random** instead of C after last round.

Target: **Register** $FF3(M_{REG})$ **and Latch** $L3(M_{LATCH})$

Protecting BRAM Register: Low-Cost Protection



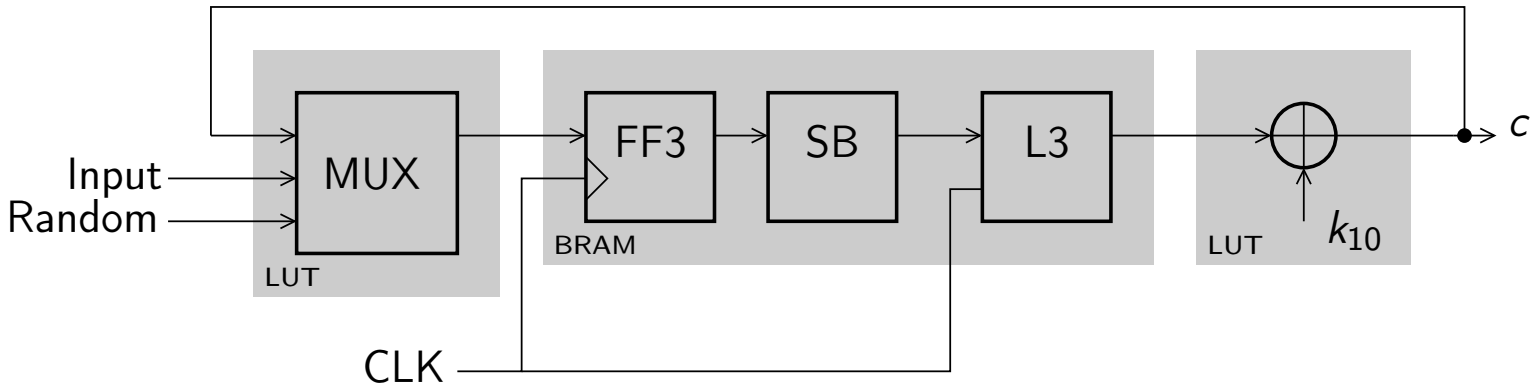
Specs: Same as **Architecture 3 but ...**
$FF3$ stores a **Random** instead of C after last round.

Target: **Register** $FF3(M_{REG})$ **and Latch** $L3(M_{LATCH})$

$$M_{REG}: \quad \mathbb{L} = HW(C \oplus SB^{-1}(C \oplus K_{10})) + \mathbb{N}$$

$$M_{LATCH}: \quad \mathbb{L} = HW(SB(C) \oplus (C \oplus K_{10})) + \mathbb{N}$$

Protecting BRAM Register: Low-Cost Protection



Specs: Same as **Architecture 3 but ...**
$FF3$ stores a **Random** instead of C after last round.

Target: **Register** $FF3(M_{REG})$ **and Latch** $L3(M_{LATCH})$

$$M_{REG}: \quad \mathbb{L} = HW(C \oplus SB^{-1}(C \oplus K_{10})) + \mathbb{N}$$

$$M_{LATCH}: \quad \mathbb{L} = HW(SB(C) \oplus (C \oplus K_{10})) + \mathbb{N}$$

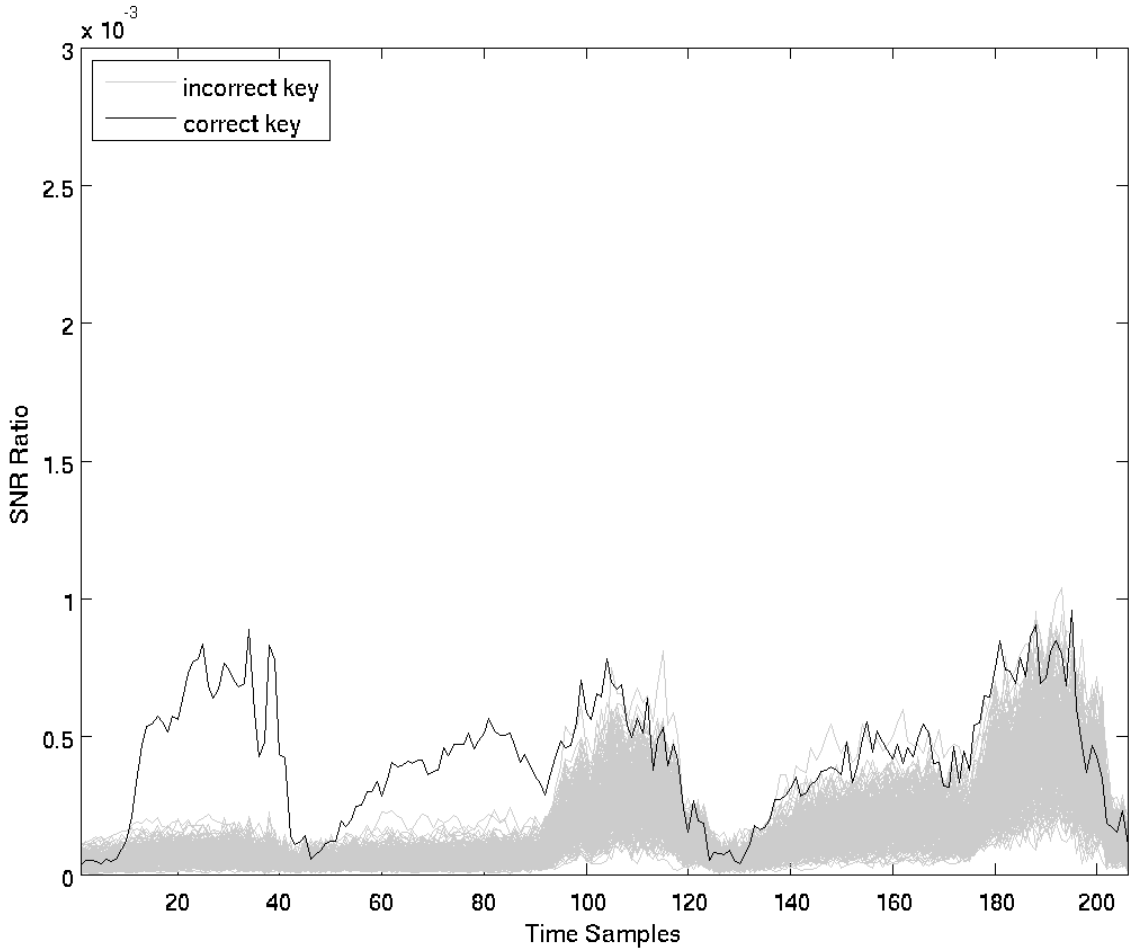
CPA: @$FF3$ and @$L3$ FAILS

Cost Comparison of Four Architectures

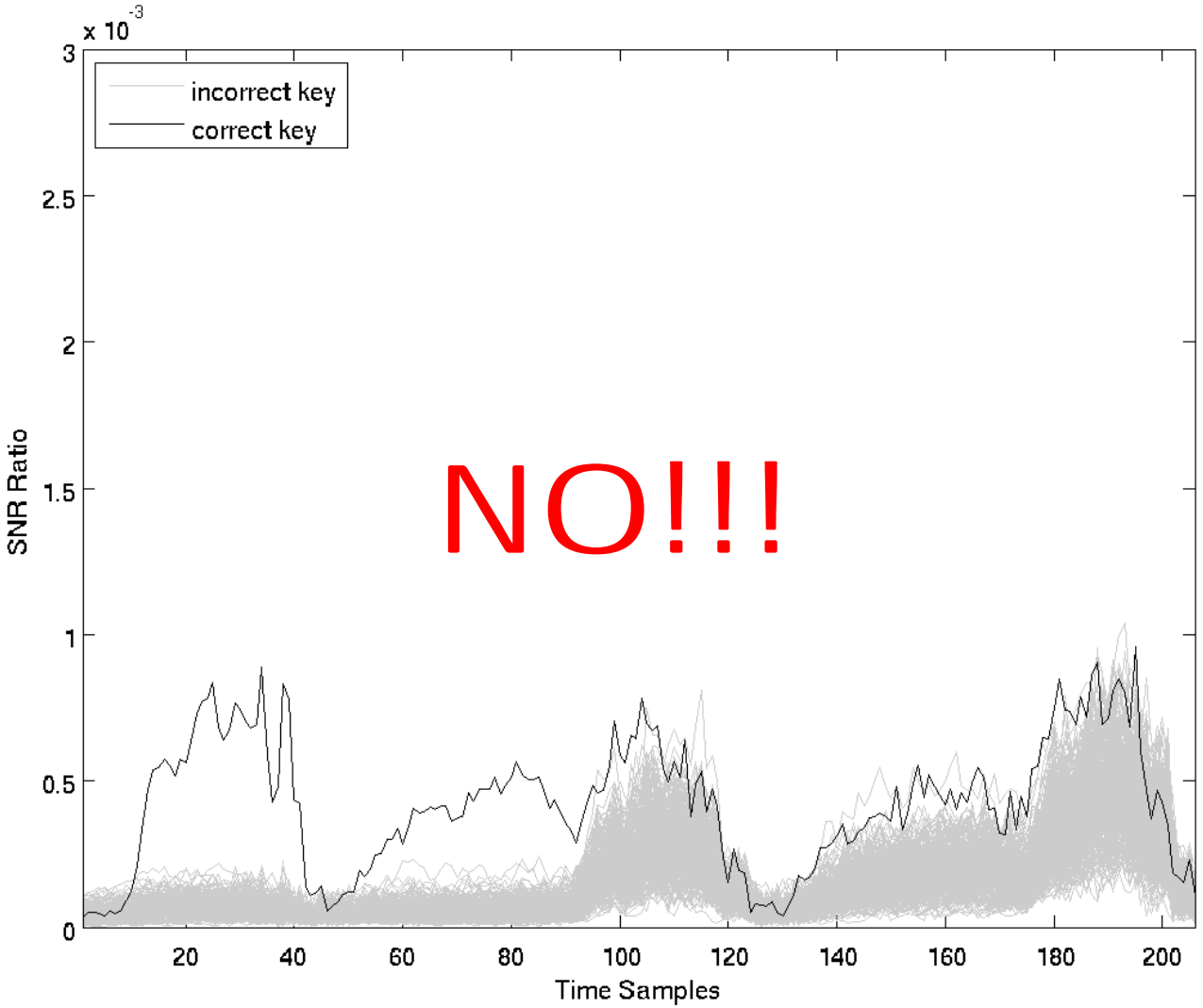
Architecture	Arch. 1	Arch. 2	Arch. 3	Arch. 4
LUT	1,360	848	848	977
Registers	268	268	140	272
BRAM	0	8	8	8
Max. Frequency [MHz]	214.3	218.5	218.5	198.53

Is the Applied Protection Enough?

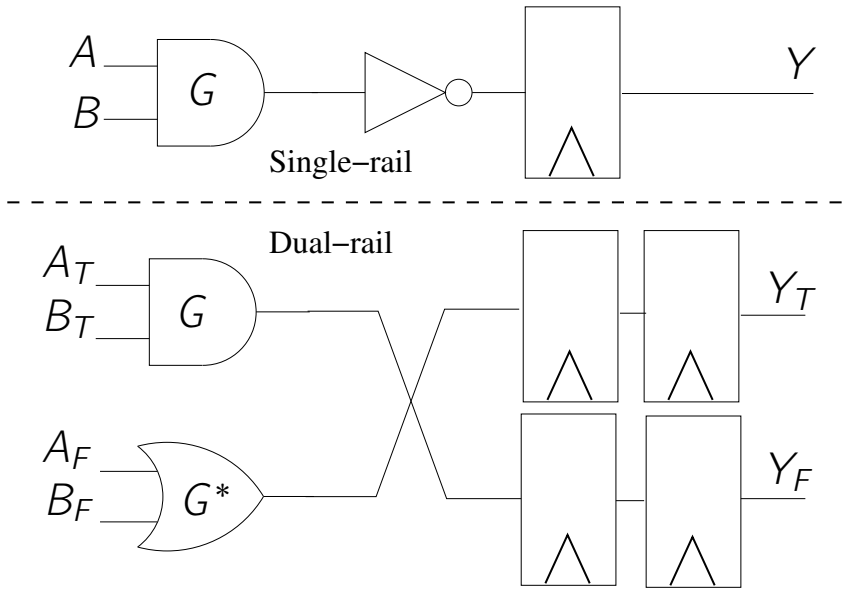
$$M'_{REG}: \quad L = \beta_0 + \sum_{b=1}^{8} \beta_b (C \oplus SB^{-1}(C \oplus K_{10}))[b] + \mathbb{N}$$



Is the Applied Protection Enough?



Improving DPL Implementations



Basics:

- Duplication ⇒ Balanced Activity.
- Two Phases ⇒ Constant Transitions.

Common Flaws in DPL

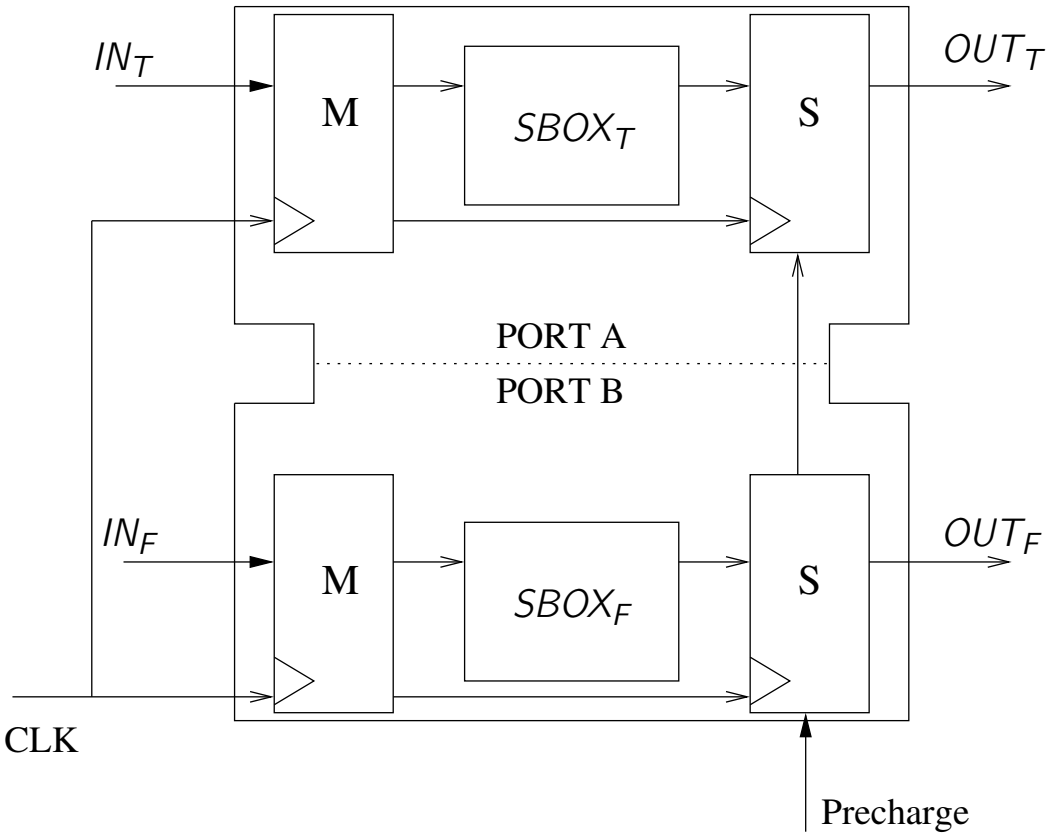
- **Early Propagation Effect**
  - Intra-delay of DPL inputs is observable at the output.
  - Solved by synchronizing inputs.
- **Technological Bias**
  - Consumption T $\neq$ Consumption F
  - Placement T $\neq$ Placement F
  - Routing T $\neq$ Routing F
  - Hard to Fix.

A Proposed DPL Architecture exploiting BRAM

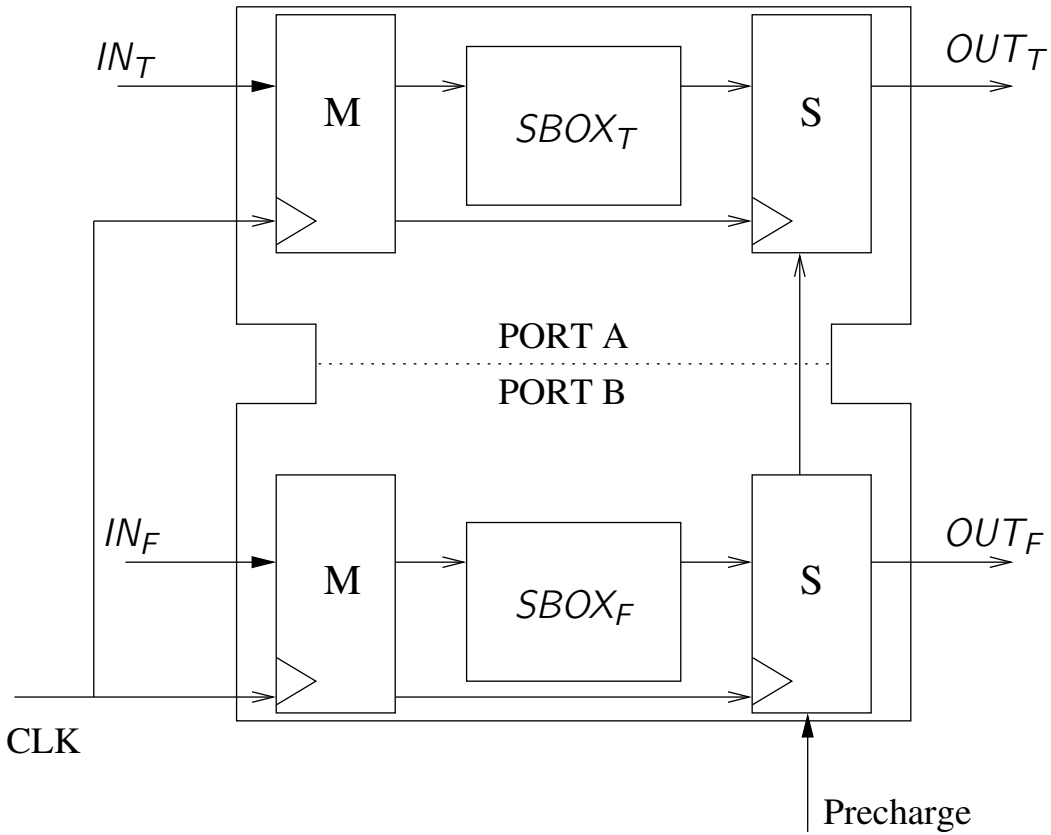
Features Applicable to DPL:

- Dual-Port ⇒ Duplication.
- Memory Array ⇒ SBOX.
- Input Registers ⇒ Master Registers.
- Output Registers ⇒ Slave Registers.
- Reset @Output ⇒ Precharge Propagation.
- Hard Macro ⇒ Balanced Placement.

A Proposed DPL Architecture exploiting BRAM



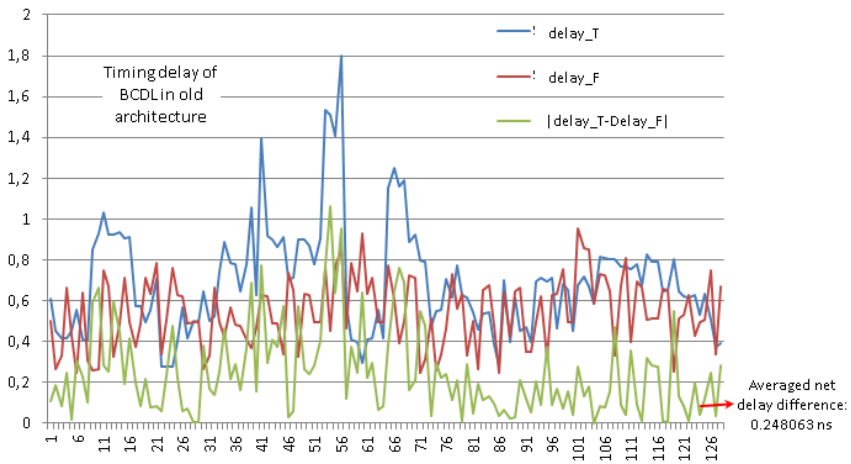
A Proposed DPL Architecture exploiting BRAM



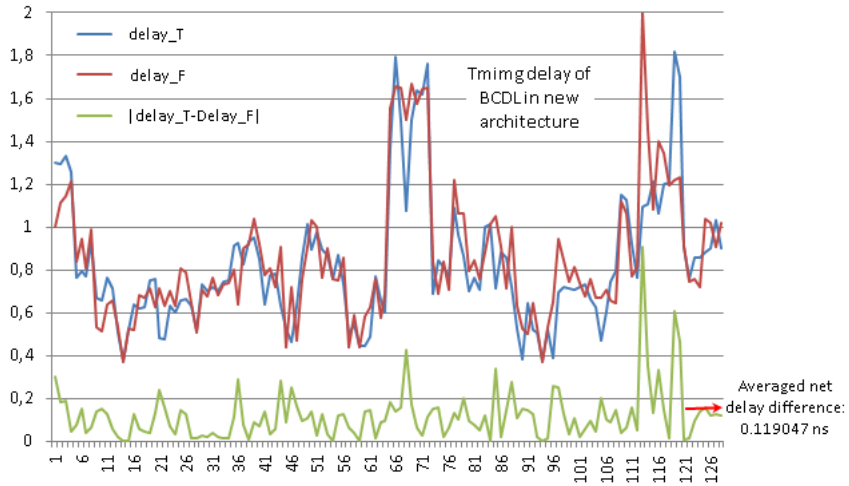
Whole Sequential part of DPL is placed in a balance manner.

$\Rightarrow$ Lower Imbalance.

Analyzing Imbalance



Unoptimised Architecture



Optimised Architecture

Presentation Outline

Conclusions

- FPGA logic and BRAM are equally vulnerable to SCA.
- BRAM leaks significantly at the output latch instead of input register.
- Leakage charaterization help designers to design compact and effective countermeasures.
- An aware designer can move from Arch. 1 to Arch. 4 with minimal overhead and 100× SCA resistance.
- Potentially applicable to Altera Architectures as well.

Perspectives

Open Questions:

How to Formally Charaterize All Leakages From a Given Hardware?