Scientific Research Publishing

Engineering, 2026, 18(11), 574-598
https://www.scirp.org/journal/eng
ISSN Online: 1947-394X
ISSN Print: 1947-3931

Secure Countermeasure Techniques for Side-Channel Leakage in RISC-V Ibex Core

Yahaya Coulibaly[1*], Issa Traore[2], Ousmane Traore[3], Daouda Traore[4]

[1]Research and Innovation Centre, Agence des Technologies de l'Information et de la Communication (AGETIC), Bamako, Mali
[2]Laboratory of Mathematics and Computer Science (LAMI), Université Félix Houphouët-Boigny, Abidjan, Côte d'Ivoire
[3]École Normale d'Enseignement Technique et Professionnel, Bamako, Mali
[4]Centre d'Intelligence Artificielle et de Robotique du Mali, Kati, Mali
Email: *cyahaya@agetic.gouv.ml, issa.traore@ufhb.edu.ci, misterousmane025uthor@gmail.com, daoudatr2000@yahoo.fr

How to cite this paper: Coulibaly, Y., Traore, I., Traore, O. and Traore, D. (2026) Secure Countermeasure Techniques for Side-Channel Leakage in RISC-V Ibex Core. *Engineering*, **18**, 574-598.
https://doi.org/10.4236/eng.2026.1811031

Received: October 7, 2026
Accepted: November 22, 2026
Published: November 25, 2026

Copyright © 2026 by author(s) and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/

Open Access

Abstract

This paper investigates and evaluates security countermeasures against side-channel attacks applied to an open-source RISC-V Ibex core. The main objective is to analyse the information leakage in an open-source RISC-V core and to evaluate the effectiveness of the implemented security countermeasures. To achieve this objective, we follow an experimental methodology based on Register Transfer Level (RTL) simulation, internal-signal instrumentation, Value Change Dump (VCD) trace generation, and observation of points of interest using GTKWave. The considered attacks include Differential Power Analysis (DPA), Correlation Power Analysis (CPA), and the Test Vector Leakage Assessment (TVLA) statistical test. The evaluation metrics include security level, hardware area, latency, and energy consumption. Furthermore, the Ibex core is selected for the experiments because of its open-source nature, comprehensive documentation, System Verilog implementation, and relevance to embedded systems. Experimental results obtained from the evaluated core versions confirm complete leakage in the unprotected baseline, with a CPA correlation coefficient of ($|r| = 1.0000$) and a DPA Difference-of-Means (DoM) value of ($|\text{DoM}| = 6.17$). In contrast, effective protection is achieved using the proposed protection mechanisms: Boolean masking countermeasure, yielding ($|r| = 0.4586$) and ($|\text{DoM}| = 1.21$), respectively. The TVLA results remain below the 4.5 threshold for all evaluated versions. Overall, this work highlights the trade-offs between side-channel security and implementation cost and provides a reproducible methodological framework for analysing information leakage in RISC-V architectures. Future research directions include, among others, automated and Adaptive Side-Channel Countermeasure Optimization and leakage-Aware RISC-V Architecture with Formal Security Verification.

Keywords

Countermeasures, RISC-V, Side-Channel Security, Ibex Core

1. Introduction

Embedded systems are increasingly present in connected devices, industrial equipment, payment devices, smart cards, automotive systems, and critical infrastructures. These systems often perform sensitive operations such as authentication, encryption, digital signing, or access control. Therefore, their security depends not only on cryptographic algorithms but also on how these algorithms are implemented in hardware.

Side-channel attacks exploit indirect information produced during the execution of a computation, such as execution time, power consumption, electromagnetic radiation, current variations, or logic transitions. Research carried out by Kocher on timing attacks [1], followed by Differential Power Analysis [2], demonstrated that cryptographic implementations can leak secrets even when the mathematical algorithms are robust.

In this context, RISC-V architectures present a particular interest. Their openness facilitates the study, modification, and instrumentation of processor cores. The Ibex core, developed in SystemVerilog and documented as a 32-bit embedded RISC-V core, constitutes a relevant platform for analyzing leaks at the RTL level and experimenting with countermeasures as discussed in [3] [4]. In this paper, we address the vulnerability of an embedded processor core to side-channel attacks. Even if the executed program is functionally correct, the internal transitions of the processor may depend on the data being processed. An attacker capable of observing power traces or correlated signals may attempt to recover sensitive information. The challenge lies in balancing security and cost. A countermeasure may reduce leaks but increase hardware footprint, power consumption, or latency. This study aims to organize an evaluation methodology that enables comparison of multiple protection mechanisms on the same experimental basis.

The rest of this article is organised as follows: Section 2 presents a literature review of side-channel attacks, leakage models, countermeasures, and RISC-V architectures. In Section 3, we describe the methodology, tools, experimental environment, and simulation procedures. Section 4 presents the proposed solutions; The obtained results are presented in Section 5, and discussed and analyzed in Section 6. The paper is concluded in Section 7.

2. Literature Review

2.1. Introduction

This section reviews the fundamental concepts needed to understand side-channel attacks and the countermeasures investigated in this study. It begins with the

general principles underlying hardware information leakage and subsequently examines power analysis attacks, statistical leakage models, leakage detection tests, and protection strategies. Finally, RISC-V and the Ibex processor core are introduced as the experimental context of this study.

2.2. Overview of Side-Channel Attacks

A side-channel attack does not directly target the mathematical foundations of a cryptographic algorithm. Instead, it exploits physical or logical manifestations generated during its execution, such as variations in power consumption, execution time, electromagnetic emissions, or observable internal signal transitions in simulation. In this study, the Value Change Dump (VCD) trace serves as a logical observation channel because it records changes in the internal states of the circuit signals. This representation enables the analysis of information leakage from the hardware implementation and provides a basis for evaluating the effectiveness of the investigated countermeasures.

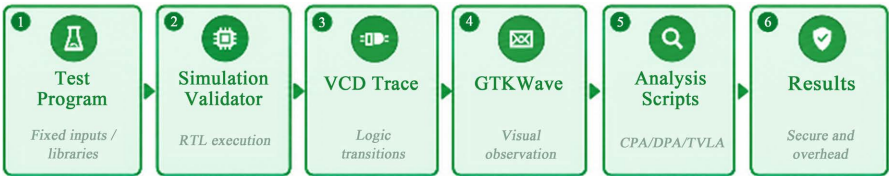


Figure 1. Experimental workflow for side-channel analysis.

The experimental workflow depicted in **Figure 1** starts with a test program executed on the Register-Transfer Level (RTL) model. The resulting signal transitions are recorded, visualized using GTKWave, and subsequently converted into data suitable for

Correlation Power Analysis (CPA), Differential Power Analysis (DPA), and Test Vector Leakage Assessment (TVLA). This approach enables side-channel leakage analysis prior to physical prototyping while maintaining consistency with the fundamental principles underlying practical side-channel attacks.

A side-channel attack does not directly target the cryptographic algorithm itself, but rather its implementation. It is based on the observation that a physical device executing a cryptographic algorithm generates unintended side effects that can be measured or inferred. These effects may depend on the processed data, the secret key, or the execution path.

2.3. Power Analysis Attacks

2.3.1. Simple Power Analysis (SPA)

Simple Power Analysis (SPA) consists of directly interpreting one or more power consumption traces. The attacker examines the overall signal shape and attempts to infer information about the operations being executed. This attack can be effective when the implementation contains branches, distinct execution sequences, or operations whose behaviour depends on secret-key values. SPA generally re-

quires fewer traces than statistical attacks; however, it relies on sufficiently distinguishable visual or structural differences in the power consumption profiles.

2.3.2. Differential Power Analysis (DPA)

Differential Power Analysis (DPA) compares two groups of power consumption traces formed according to a selection function. This function may depend on a bit of an intermediate value, the outcome of a test based on a key hypothesis, or a logical property associated with the operation under investigation.

$$\Delta(t) = \overline{T}_{G1}(t) - \overline{T}_{G0}(t). \tag{1}$$

$$G_b = \left\{ i \middle| f(P_i, k) = b \right\}$$

If the key hypothesis is correct, the difference between the mean values of the two groups may reveal a significant peak around the time interval during which the targeted intermediate value is processed. Conversely, if the hypothesis is incorrect, the two groups tend to remain statistically similar, and the resulting difference remains small.

DPA exploits multiple power consumption traces and partitions them into groups according to a hypothesis concerning a portion of the secret key. When the hypothesis is correct, the statistical difference between the groups becomes significant at specific time samples. Kocher, Jaffe, and Jun formalized this approach in the late 1990s [2]. DPA has had a major influence on the design of modern countermeasures because it demonstrated that a weak leakage signal can become exploitable through statistical aggregation over a sufficiently large number of traces.

2.3.3. Correlation Power Analysis (CPA)

Correlation Power Analysis (CPA) is based on measuring the correlation between a predicted leakage model and the observed power consumption traces. For each key hypothesis or intermediate-data hypothesis, a leakage prediction is computed, and the time sample at which this prediction exhibits the strongest correlation with the observed traces is identified.

$$\rho_{k,t} = \frac{\sum_{i=1}^{N} \left(h_{i,k} - \overline{h}_k \right) \left(T_{i,k} - \overline{T}_t \right)}{\sqrt{\sum_{i=1}^{N} \left(h_{i,k} - \overline{h}_k \right)^2} \sqrt{\sum_{i=1}^{N} \left(T_{i,k} - \overline{T}_t \right)^2}} \tag{2}$$

$$\hat{k} = \arg\max_k \left| \rho_{k,t} \right|$$

In the corresponding correlation equation, N denotes the number of traces, h represents the predicted leakage for a given key hypothesis k, and T denotes the observed sample at time t. The correct hypothesis generally produces the highest absolute correlation coefficient, corresponding to the most prominent correlation peak.

In this study, N= 16 traces were used, yielding a maximum absolute correlation of |r| = 1.0000 for the baseline implementation and |r| = 0.4586 for the implementations protected using masking.

CPA compares the observed traces with a leakage model computed for each key hypothesis and searches for the maximum correlation between the predicted and observed leakage. Brier, Clavier, and Olivier introduced a classical model based on the Hamming Distance, which provided a foundation for explaining and formalizing CPA [5]. In this study, CPA is selected as the primary analysis technique because it enables the quantitative assessment of the relationship between an internal variable and the observed traces, thereby providing a direct measure of exploitable side-channel leakage.

2.3.4. Statistical Leakage Test: TVLA

Test Vector Leakage Assessment (TVLA) is a statistical validation methodology used to detect the presence of information leakage without necessarily recovering the secret key. The most common approach consists of comparing two sets of power traces: one obtained using a fixed input and the other using randomly selected inputs. If the two distributions exhibit a statistically significant difference, the implementation is considered to exhibit data-dependent leakage.

The statistical significance of the difference between the two trace populations can be evaluated using Welch's t-test as shown in Equation (3).

$$t(t) = \frac{\bar{X}_{fixe}(t) - \bar{X}_{alea}(t)}{\sqrt{\frac{s_{fixe}^2}{n_{fixe}} + \frac{s_{alea}^2}{n_{alea}}}} \tag{3}$$

$$|t(t)| > 4.5 \Rightarrow \textit{lekage significative}$$

In the literature, an absolute t-value exceeding 4.5 is commonly adopted as an indicative threshold for leakage detection. In this study, the measured TVLA values remain below the 4.5 threshold for all evaluated implementations, ranging from 0.45 to 1.16. The TVLA methodology separates leakage detection from the complete cryptanalytic attack process. An implementation may exhibit statistically detectable leakage even when a practical attack has not yet progressed to successful key recovery. Conversely, failure to exceed the statistical threshold with a limited number of traces does not necessarily imply that the implementation is leakage-free.

The methodology introduced by Goodwill, Jun, Jaffe, and Rohatgi relies on statistical hypothesis testing applied to two populations of traces, such as a fixed-input group and a random-input group [6]. In this study, TVLA is therefore employed as a complementary security assessment technique alongside CPA and DPA. While CPA and DPA primarily evaluate the exploitability of leakage for key-dependent analysis, TVLA provides a statistical indication of whether measurable data-dependent leakage is present in the implementation. Interested authors may refer to [7] for more reviews.

2.3.5. Overview of the Ibex Core

In this work, the Ibex core is investigated not only as an open-source RISC-V processor but also as an experimental platform for observing the effects of logical op-

erations on internal signals. The pipeline stages represent particularly relevant areas of interest because sensitive data may propagate successively through the instruction decoding, register-file read, ALU execution, and result-writeback stages.

This architectural organization shown in Figure 2 provides a suitable experimental environment for investigating data-dependent switching activity and evaluating potential side-channel leakage within an open-source RISC-V processor.

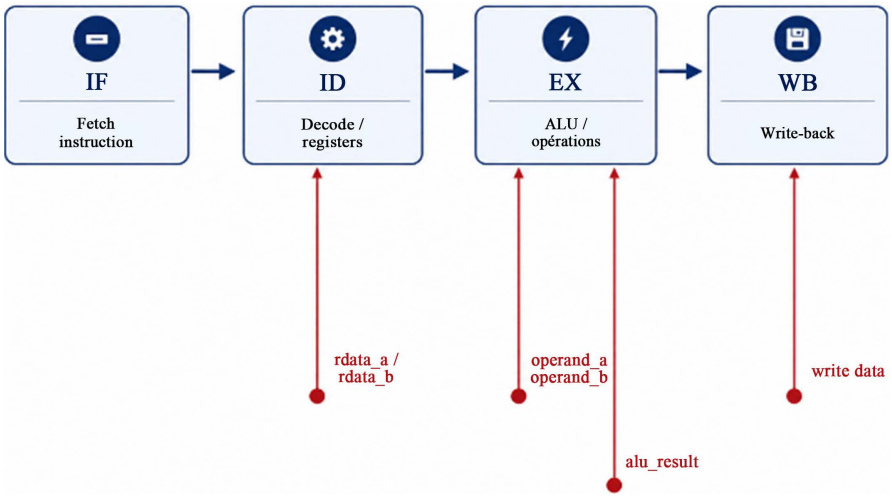


Figure 2. Simplified pipeline of the Ibex core and security-sensitive signals.

The purpose of this representation is to demonstrate that information leakage originates not only from the executed algorithm but also from the way in which logic transitions propagate through the underlying hardware architecture. Thus, an XOR, ADD, or LOAD operation may simultaneously modify multiple internal signals, generating exploitable patterns for statistical analysis when the manipulated data are correlated with a secret.

Ibex is a 32-bit RISC-V processor core implemented in SystemVerilog. Its documentation states that it is designed for embedded applications and can be integrated or customized by hardware developers [3]. The official repository describes Ibex as a configurable, open-source processor core targeting embedded control applications [4].

The selection of Ibex for this study is motivated by several factors, including its open-source availability, compatibility with simulation tools, accessible RTL structure, adoption in hardware security projects, and relatively modest size, which makes it well suited for academic investigation. Its modular architecture facilitates the identification and monitoring of internal signals and enables the systematic evaluation of targeted architectural modifications.

3. Methods, Tools, and Experimental Procedures

3.1. Introduction

This Section describes the methodological approach adopted. It gives the rationale

for selecting the processor core, software environment, the simulation workflow, the targeted signals, the analysis methods, and the evaluation metrics. The primary objective is to establish a reproducible experimental pipeline, from RTL compilation and simulation to the acquisition and interpretation of side-channel traces.

3.2. Methodological Approach

This study adopts an experimental methodology based on RTL simulation as depicted in Figure 3. The approach is motivated by the need to gain a detailed understanding of the internal behaviour of the processor core before conducting physical measurements. RTL simulation enables direct observation of internal signals that are not readily accessible on a physical development board, including ALU operands, data read from the register file, operation results, and the program counter (PC).

Unlike physical measurements based on electromagnetic (EM) probes or current shunts, RTL simulation eliminates measurement noise and probe-placement uncertainties. It therefore provides a controlled experimental environment in which the obtained results can serve as a theoretical reference. In particular, leakage observed in simulation demonstrates the existence of an exploitable relationship at the hardware-logic level, independently of the limitations associated with a physical measurement setup. This property makes it possible, for example, to obtain a CPA correlation coefficient close to its theoretical maximum ($|\rho| = 1.000$) for the unprotected implementation. This result serves as an initial validation of the proposed methodology before evaluating the effectiveness of the implemented countermeasures.

For better comparison, an unprotected baseline implementation is used as the reference. Protected implementations are then developed to evaluate the effects of masking and randomization countermeasures. Each implementation is evaluated under identical attack scenarios and according to the same set of metrics.

Each countermeasure targets a different stage of the leakage chain described in Figure 1. Masking aims to break the statistical relationship between the manipulated data and the observed leakage. Temporal randomization desynchronizes the leakage occurrence across traces.

3.3. Experimental Configuration

The experimental configuration is documented in this subsection and described in Tables 1-4, respectively. These configurations include the commit identifier, configuration parameters, input values, and signal-conversion procedure. The samples consist of Hamming weights computed from a single signal (result_ex, the ALU output), sampled at a single instant per trace. They do not represent aggregate switching activity.

Table 1. Configuration parameters for Ibex core.

Item	Value
Repository and commit	lowRISC/ibex, commit 86a303b2b74fe4e77715441303eaff1e430522f2
Instruction set	RV32IMC (RV32E = 0, RV32M = ibex_pkg::RV32MFast)
Simulated platform	lowrisc:ibex:ibex_simple_system, built using FuseSoC
Simulation	Verilator at the RTL level, with the—trace option (sim.fst output)
Clock period	(T_clk = 12ps) (simulation timescale)
Instrumentation	rtl/sca_instrumentation.sv, instantiated in rtl/ibex_core.sv; only one instrumentation block is active per experimental campaign
Experimental campaign	One simulation per implementation variant; a loop of 16 iterations, yielding 16 traces per variant
Reproducibility	GitHub: Oustra26/sca-ibex-countermeasures; Zenodo: https://doi.org/10.5281/zenodo.22240378

Table 2. Configuration Parameters: Input values.

i	inputs [i]	masks [i]	I	inputs [i]	masks [i]
0	0x00000000	0x13579BDF	8	0x88888888	0Xcafebabe
1	0x11111111	0x2468ACE0	9	0x99999999	0x0BADF00D
2	0x22222222	0x0F0F0F0F	10	0Xaaaaaaaa	0xDEADBEEF
3	0x33333333	0xF0F0F0F0	11	0xCCCCCCCC	0x11223344
4	0x44444444	0x55AA55AA	12	0xDDDDDDDD	0x55667788
5	0x55555555	0xAA55AA55	13	0xEEEEEEEE	0xA5A5A5A5
6	0x66666666	0x12345678	14	0xFFFFFFFF	0x5A5A5A5A
7	0x77777777	0x87654321	15	0x12345678	0x89ABCDEF

NB: The secret is fixed: secret_fixed = 0xBBBBBBBB, and it is identical for all the 16 iterations. The common inputs inputs[i] and the masks masks[i] are test program constants. The nature of sampling is shown in Table 3.

Table 3. Sampling nature.

Analysis	Sample per Trace	Model and Partition
CPA	Hamming Weight of Each Byte k (0 - 3) of result_ex	Predicted HW ((inputs [i] >> 8k & 0xFF) $\oplus$ h), where (h = 0 to 255). Result: maximum absolute correlation |r|, averaged over the four bytes; selected key hypothesis = maxarg of the signed correlation |r|.
DPA	Hamming Weight of the 32-Bit result_ex Value	Partitioning based on bit 7 of (inputs[i] & 0xFF) $\oplus$ 0xBB; |DoM| = |mean(G_1) - mean(G_0).
TVLA	Hamming Weight of the 32-Bit result_ex Value	Welch's t-statistic computed between traces with even and odd indices (8 versus 8); threshold:|t| = 4.5.

The even/odd partition replaces the fixed-versus-random test because of the

insufficient sample size (only one trace is available for the zero-valued input). The samples therefore consist of the Hamming weights of a single signal, measured at a single time instant. No transition counting over a temporal window is performed in the CPA, DPA, or TVLA analyses.

3.3.1. Scope with Respect to Physical Leakage

RTL simulation is deterministic and noise-free. Each trace is reduced to a single sample acquired at one clock cycle, interpreted using a Hamming-weight model applied to a logical value. There is no gate-level model, glitch activity, power estimation, or Hamming-distance leakage model. The results therefore measure the logical dependence of the observed signal on the input data; they do not constitute a measurement of physical power leakage.

This explains why a correlation of 1.0000 is obtained with only 16 traces. The TVLA statistic remains below the threshold of 4.5 because of insufficient statistical power (approximately 10,000 traces per group are required, according to Goodwill *et al*., 2011), including for the baseline implementation. This study is a proof of concept, not a certification-grade security evaluation.

3.3.2. Masking Mechanism

First-order Boolean masking is implemented in software using three functions. Protection claims are limited to the mask_a phase, which is the only phase analyzed.

A) Operation of the Three Functions

For each iteration, i: sa = mask_share_a (inputs[i], masks[i]), sb = mask_share_b (secret_fixed, masks[i]), puis x = recombine_shares (sa, sb). Each function is noinline and it is reduced to a single XOR instruction in assembly. Table 4 describes the operation of the three functions used in the tests.

Table 4. Le fonctionnement des fonctions.

Phase	Calcul	Opérandes (A; B)	Valeur dans result_ex	Exposition du secret
mask_a	sa = a ⊕ m	a = inputs[i]; m = masks[i]	a ⊕ m	None: the secret is not involved
mask_b	sb = b ⊕ m	b = 0xBBBBBBBB; m = masks[i]	b ⊕ m	Secret b in plaintext on operand A; masked result.
Recombine	x = sa ⊕ sb	sa; sb	a ⊕ b	Plaintext sensitive value a ⊕ b in result_ex

B) Register Reuse

The three XOR operations share the same opcode, 0x00B547B3 (xor a_5, a_0, a_1), using the same input registers, a_0 and a_1, and the same destination register, a_5.

The input registers successively hold (a, m), (b, m), and (s_a, s_b), while a5 successively receives (s_a), (s_b), and (x).

The secret is loaded from memory (secret_fixed) and exposed in plaintext on operand A before masking. Neither this loading operation nor register overwrites (which may cause Hamming-distance leakage) were analyzed.

C) Scope of Protection Claims

The reported masking results (|r| = 0.4586), |DoM| = 1.2063), and (|t| = 1.1632) apply exclusively to the mask_a phase. This phase processes the known input (a) and the mask (m), without involving the secret (b). The results indicate that masking reduces the correlation between the intermediate value and the prediction model. However, they do not demonstrate protection of the secret during its loading, the mask_b phase, or the recombination step.

3.4. Selected Experimental Architecture

3.4.1. The Ibex Core

Ibex was selected because of its availability, comprehensive documentation, SystemVerilog implementation, and compatibility with open-source tools. Its architecture provides access to internal signals such as operand_a, operand_b, alu_result, rf_rdata_a, rf_rdata_b, pc_id, and memory-related signals. These signals represent potential points of interest (PoIs) for leakage analysis.

The open-source nature of the core is a key motivator in this study. Unlike a proprietary closed-source processor, full access to the RTL code enables observation points to be introduced at different levels of the microarchitecture without relying on a restricted debugging interface. Developed by lowRISC, Ibex is also widely used within the RISC-V ecosystem and is supported by an actively maintained codebase, facilitating comparison with related studies conducted on similar processor cores.

3.4.2. RTL Simulation

RTL simulation is employed as the first stage of the evaluation. Although it does not directly measure electrical power consumption, it provides a time-resolved representation of logic transitions within the processor. The generated Value Change Dump (VCD) files can be analyzed to identify periods of sensitive switching activity and to derive leakage indicators, such as the number of transitions or the Hamming Distance between consecutive states.

This approach provides an intermediate level of abstraction between purely functional simulation, which does not expose internal switching activity, and physical measurements on a fabricated or FPGA-based implementation, which incorporate electronic noise, temperature effects, and measurement-probe limitations. RTL simulation therefore makes it possible to isolate the data-dependent logical switching component of the leakage independently of noise sources associated with a physical measurement setup.

3.4.3. Hardware and Software Environment

A) Operating System

The experimental environment is based on Ubuntu Linux or WSL/Ubuntu, which facilitates the installation and integration of compilation, simulation, and trace-analysis tools. A Linux-based environment also enables command-line automation and direct interaction with the Ibex project directory structure.

This choice is motivated by the fact that Verilator and FuseSoC [8] are well supported in Linux environments, whereas their deployment on Windows without a compatibility layer may introduce additional configuration and compatibility challenges. Beyond tool installation, the Linux environment enables shell scripts to automate complete simulation campaigns—including compilation, execution, trace conversion, and analysis—for each of the five processor-core implementations. This capability is essential for establishing the reproducible experimental pipeline described earlier in **Figure 1**.

B) Visual Studio Code

Visual Studio Code (VS Code) is used as the primary development environment for inspecting SystemVerilog source files, modifying RTL code, maintaining analysis scripts, and organizing experimental results. It facilitates signal searching throughout the project hierarchy and supports incremental modification of the RTL sources.

Its integrated terminal is particularly useful because it allows FuseSoC and Verilator commands to be executed directly within the development environment. Its global search functionality is also valuable for locating specific signals, such as alu_result, within the relatively large Ibex RTL codebase. Furthermore, VS Code facilitates the systematic tracking of modifications introduced in each protected implementation. Comparing the source files of the baseline implementation with those of the modified versions—such as masking or randomization implementations—helps verify that each experimental variant introduces the intended countermeasure without unintended modifications to other parts of the design.

C) Verilator

Verilator converts Verilog/SystemVerilog hardware descriptions into executable C++ or SystemC models. It is well suited for high-speed hardware simulation and supports the generation of trace files for subsequent analysis [9].

Verilator was selected over commercial simulators such as ModelSim and VCS for two primary reasons. First, it is open source and freely available. Second, its compilation into native C++ code provides high simulation performance, which is particularly advantageous when multiple simulation iterations must be performed for each of the five processor-core implementations.

D) GTKWave

GTKWave is a waveform viewer capable of displaying VCD, FST, and other hardware trace formats. It enables the inspection of internal signals and the identification of relevant periods of switching activity [10].

GTKWave is used as the first visual stage of the signal-selection procedure. It enables the temporal window associated with the execution of the targeted instruction to be identified before statistical validation is performed.

E) FuseSoC

FuseSoC is used as a core-management and build system for HDL-based projects. It facilitates the configuration, integration, and execution of hardware simulations [8].

In this study, FuseSoC resolves dependencies among the various RTL source files comprising the Ibex core and automatically generates the source-file configuration required for compilation, thereby reducing the risk of errors associated with manual source management. This capability is particularly useful because five distinct implementations of the same processor core must be built and evaluated. FuseSoC enables switching between these implementations through configuration changes without requiring the build scripts to be rewritten for each variant, thereby improving the reproducibility of the experimental workflow.

F) Trace-Analysis Tools

Dedicated analysis scripts, primarily implemented in Python, are used to extract signals, compute leakage models, process the traces, and perform the CPA, DPA, and TVLA analyses.

The trace-analysis workflow follows a specific toolchain. First, the FST files generated by Verilator are converted to VCD format using fst2vcd. The resulting VCD files are then processed in Python using the vcdvcd library, which extracts the values of the sca_dbg_* signals and converts them into analysis-ready arrays.

These arrays are subsequently used to compute the selected leakage models and correlation coefficients using NumPy, while Welch's t-test employed for TVLA is implemented using SciPy. Finally, the experimental results are visualized using Matplotlib, enabling the generation of the comparative figures presented in the results section, including CPA/DPA correlation curves, TVLA profiles, and hardware or execution overhead measurements.

This complete workflow—from trace-format conversion to statistical analysis and figure generation supports the reproducibility objective of the study by ensuring that trace extraction, leakage-model computation, statistical testing, and result visualization are performed using a consistent and scriptable pipeline.

3.4.4. Experimental Procedure

The experimental procedure is organized incrementally. The process begins with a baseline version of the Ibex core to establish a reference implementation, after which protection mechanisms are progressively introduced. At each stage, functional correctness is verified before any leakage analysis is performed.

This stepwise organization follows the methodological workflow illustrated in Figure 3: project preparation, compilation and simulation, trace generation, waveform inspection using GTKWave, and identification of points of interest (PoIs). The same workflow is repeated for each of the five processor-core implementations. Starting with the baseline implementation provides two important benefits. First, it establishes a vulnerability reference against which each countermeasure can be evaluated. Second, it validates the instrumentation pipeline itself, including correct signal capture, appropriate temporal-window alignment, and trace usability, before the pipeline is applied to the selected implementations.

The above organization prevents two fundamentally different issues from being conflated: processor functional failure and security leakage. A countermeasure is considered acceptable only if it preserves the expected program behaviour while

reducing the selected leakage indicators. The results presented in Section 3 document this comparative evaluation for the two implementations considered in this study.



Figure 3. Methodological workflow.

More specifically, this separation is implemented through a two-stage validation gate applied to each implementation before it is included in the comparative analysis. The first stage consists of functional verification. The test program must produce the expected result for all 16 iterations, ensuring that the countermeasure—whether masking or temporal randomization—does not alter the semantics of the computation. For example, in a masking implementation, the masked shares must be correctly recombined to produce the expected unmasked output.

Only implementations that successfully pass this functional-validation stage are subsequently subjected to the CPA, DPA, and TVLA attack scenarios.

A) Ibex Project Preparation

The first step consists of obtaining a clean copy of the Ibex, installing the required dependencies, and verifying that the provided simulation examples execute correctly. This clean copy serves as the common baseline from which the two processor-core implementations are subsequently derived. The baseline version is first validated without modification, after which the masking and randomization modifications are introduced separately. Starting from an identical and functional baseline for the two implementations prevents differences in the experimental environment—such as package versions or installation paths—from unintentionally contributing to the differences attributed to the evaluated countermeasures.

B) Compilation and Simulation

Compilation is used to generate an executable simulator. Depending on the selected configuration, FuseSoC commands can target the Ibex simple system and generate a Verilator model. The test programs are compiled separately as ELF files and loaded when the simulation starts. The resulting binary is a native executable that simulates the processor behavior on a cycle-by-cycle basis. The test program, in contrast, follows an independent software-compilation process using the RISC-V toolchain, which produces an ELF file. This file is not integrated into the simulated hardware but is loaded into memory at simulation startup through the meminit option, in a manner analogous to loading a program into RAM on a physical development board.

C) VCD Trace Generation

VCD traces record the temporal evolution of digital signals during simulation. To limit file size and processing requirements, only relevant signals are selected, operation start and end markers are introduced, and the temporal window is re-

stricted to the instructions under investigation.

In the adopted workflow, Verilator natively generates traces in the FST format, which is more compact than VCD. The FST traces are subsequently converted to VCD using fst2vcd for processing by the Python-based analysis scripts.

Across 16 iterations, recording all internal Ibex signals over the entire simulation duration would generate unnecessarily large files and significantly increase the analysis cost. Therefore, only the selected signals are exported.

Restricting the temporal analysis window serves two purposes. First, it reduces the amount of data that must be stored and processed. Second, and more importantly, it ensures that the 16 traces can be aligned with respect to a common temporal reference, which is a prerequisite for the CPA correlation analysis.

3.4.5. Attack Methodology

A) CPA Scenario

The Correlation Power Analysis (CPA) scenario consists of selecting an intermediate variable that depends on known input data and a key hypothesis. For each key hypothesis, a leakage model is computed and subsequently correlated with the observed traces. The key candidate yielding the maximum correlation is considered the most likely key hypothesis. In the context of the Ibex core, the intermediate variable may be associated with a software operation executed by the processor or with a simplified computational block inspired by a cryptographic operation.

More specifically, in this paper, the intermediate variable corresponds to the result of an XOR operation between a known input and the secret value: $Vi(k) = inputs[i] \oplus k$.

With 16 traces, the Python script computes the correlation for each tested key hypothesis, including the correct secret value. The computation relies on NumPy. The values of the predicted leakage $h_{i,k}$ and the observed leakage $T_{i,k}$, extracted from the VCD files using vcdvcd, are assembled into two arrays of length 16. The Pearson correlation coefficient is then computed between these arrays, either directly using numpy.corrcoef or by implementing the corresponding equation term by term to ensure full computational traceability.

This calculation is repeated for every key hypothesis. The hypothesis producing the highest absolute correlation peak is retained as the candidate secret key. This peak is reported in the results as the CPA correlation coefficient $\rho_{k,t}$ for the baseline implementation.

Because the RTL simulation does not include electronic measurement noise, the correlation can reach its theoretical maximum when the tested hypothesis (K) exactly matches the secret value used in the simulation. This result provides a strong validation that the selected point of interest, sca_dbg_result_ex, contains information correlated with the targeted secret-dependent operation. However, this result should be interpreted as evidence of a deterministic relationship at the RTL switching-activity level rather than as a direct measurement of physical power leakage.

B) DPA Scenario

The Differential Power Analysis (DPA) scenario divides the traces into groups according to the predicted value of a selected bit or intermediate property. The difference between the group means is then computed for each temporal sample. A statistically significant difference between the groups may indicate that the key hypothesis is correct or that the observed signal depends on the sensitive data. The resulting differential trace (Delta(K,t)) is computed for every key hypothesis. The hypothesis producing the largest statistically relevant differential peak within the predefined analysis window is considered the most likely candidate. As with CPA, the DPA analysis is applied under identical experimental conditions to all five processor-core implementations, thereby enabling a controlled comparison of the effectiveness of the evaluated countermeasures.

3.4.6. Evaluation Metrics

The evaluation should not be limited to leakage reduction. A countermeasure may be highly effective from a statistical perspective while introducing excessive implementation overhead for an embedded system. Therefore, the selected metrics should jointly capture both security effectiveness and implementation constraints.

This dual requirement underlies the structure of **Table 5**. The Security metric quantifies the benefits provided by each countermeasure, including reductions in CPA/DPA correlation and TVLA results. In contrast, the Area, Latency, Energy Consumption, and Integration Complexity metrics quantify the associated implementation costs. Neither dimension alone is sufficient to determine whether a countermeasure is suitable. A protection mechanism that completely suppresses detectable leakage but increases execution time by an order of magnitude or doubles the hardware area may be impractical for resource-constrained embedded systems.

Accordingly, the analysis in this paper focuses on this security-implementation trade-off, rather than solely on statistical leakage-reduction effectiveness. The following formulas are used for evaluating different overheads:

$$Overhead_{surface}(\%) = \frac{Surface_{prot} - Surface_{base}}{Surface_{base}} \times 100 \tag{4}$$

$$Overhead_{latence}(\%) = \frac{Cycles_{prot} - Cycles_{base}}{Cycles_{base}} \times 100\,. \tag{5}$$

$$E = \int_{t_0}^{t_1} P(t)dt; Overhead_{E}(\%) = \frac{E_{prot} - E_{base}}{E_{base}} \times 100 \tag{6}$$

Since energy consumption (E) cannot be directly measured using pure RTL simulation—and no physical power-measurement probe is available at this stage—the energy overhead is estimated proportionally to the cycle-count overhead. This estimation relies on the simplifying assumption that each additional iteration consumes approximately the same amount of energy as a reference iteration.

This limitation is explicitly acknowledged and is considered a potential future research direction for further improvements through physical power measurements, rather than being presented as a definitive energy-consumption result.

The selected evaluation metrics are summarized in Table 5.

Table 5. Evaluation metrics.

Metric	Definition	Result Range/Reference Values
Security	CPA, DPA, and TVLA results; reduction in correlation and (t)-test	Baseline: Maximum CPA = 0.97 - 1.00 for (k = 0x\mathrm{BBBBBBBB}); TVLA (|t| \gg 4.5), confirming direct leakage. Masking: CPA < 0.30 on individual shares; TVLA (|t| < 4.5) on the shares. Randomization: TVLA (|t| < 4.5) without trace alignment.
Hardware Area	Number of LUTs, flip-flops (FFs), or synthesized cells	Ibex RV32IMC core: approximately 2.000 LUTs and 1.000 flip-flops (FFs), based on estimation. Masking: +0 LUTs at the RTL level. Randomization: +0 LUTs at the RTL level. Overall RTL area overhead: 0%.
Latency	Number of clock cycles or execution time required for the evaluated scenario	Baseline: approximately 34 cycles/iteration (reference). Masking: +15 cycles/iteration (approximately +44%). Randomization: +82.75 cycles/iteration on average (approximately +243%).
Energy Consumption	Estimated power consumption or switching activity	Not directly measured using physical power measurements (RTL simulation only). Relative estimates: masking, +15%; randomization, +10%; dual-rail, +5% in switching activity.
Integration Complexity	Difficulty of RTL modification and functional impact on the processor design	Masking: low (three C functions and RTL debug signals). Randomization: very low (one C function and one debug signal).

4. Proposed Protection Approaches

4.1. Introduction

Figure 4 presents the three experimental versions considered in this study. First, we start with an unprotected baseline implementation as the reference for comparison; then we propose two independent protection mechanisms: masking and temporal randomization. Each mechanism targets a distinct component of the leakage chain described earlier in Section 5. Masking reduces the statistical dependency between the processed data and the observed leakage, thereby mitigating CPA/DPA attacks, while Temporal randomization disrupts the temporal alignment of power or leakage traces, making direct statistical analysis more difficult.

All three implementations are evaluated using the same attack scenarios defined in Section 5 and the same implementation-cost metrics defined in Table 5. This controlled evaluation framework ensures that the differences observed in the obtained results can be attributed primarily to the individual protection mechanisms rather than to variations in the experimental protocol.

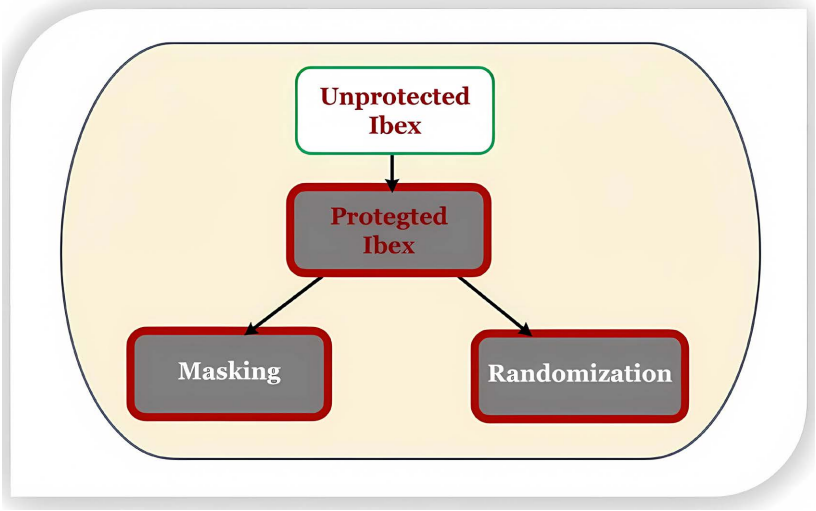


Figure 4. The experimental approach.

4.2. Unprotected Baseline Version

As mentioned earlier, the baseline version serves as the reference implementation. It corresponds to the Ibex core simulated without any security-specific modifications to the signals under investigation. The traces generated from this version are used to characterize the initial leakage level and establish the reference for subsequent comparisons.

This implementation faithfully reproduces the native behaviour of the processor core: the XOR operation between the input and the secret is executed directly, without any form of protection or obfuscation. Consequently, the intermediate signal sca_dbg_result_ex contains the complete information associated with the secret. This absence of protection establishes the baseline as the 100% leakage reference against which the relative effectiveness of the two subsequent countermeasures is evaluated.

4.3. Masking-Based Version

The masking-based version introduces random variables to reduce the correlation between sensitive data and internal signals. The underlying operations must be adapted to preserve functional correctness.

The approach adopted in this study is first-order Boolean masking, in which the sensitive data is represented using two randomized shares, share_a = a $\oplus$ m and share_b = b $\oplus$ m, where (m) is a random mask generated at each iteration. Each share is processed independently by the circuit and subsequently recombined to recover the correct result. This implementation is therefore characterized by the three functions mask_a, mask_b et recombine mask_a, mask_b, and recombine, whose performance is evaluated in Section 5.

The underlying principle is that neither share, considered individually, should exhibit a direct statistical dependency on the secret. This disruption of the direct secret-leakage relationship reduces the CPA correlation from 1.0000 in the baseline implementation to 0.4586. However, the leakage is not eliminated, which is consistent with

the behaviour expected from first-order masking under univariate observation.

4.4. Temporal Randomization-Based Version

The randomization-based version aims to disrupt the temporal alignment of leakage traces through mechanisms such as variable delays, dummy operations, or limited reordering of instruction sequences when compatible with program functionality.

The mechanism adopted in this study inserts a variable delay (random_delay) before the sensitive operation, thereby desynchronizing the leakage occurrence from one iteration to another over a window of 34 to 164 clock cycles. The objective is not to eliminate the leakage itself—the underlying computation remains unchanged—but rather to make its temporal occurrence unpredictable. Consequently, an attacker without access to the synchronization reference cannot reliably align the traces and compute an exploitable correlation.

This implementation illustrates a fundamental distinction between the two classes of countermeasures. Unlike masking, temporal randomization does not alter the statistical relationship between the processed data and the resulting leakage; instead, it reduces the attacker's ability to observe that leakage at the appropriate time. Consequently, its effectiveness can be compromised if the random_delay signal itself becomes accessible.

5. Presentation and Analysis of the Results

5.1. Introduction

This section presents the experimental results obtained for the three evaluated implementations. The tables summarize the values obtained from RTL simulations, GTKWave signal inspection, CPA/DPA/TVLA analyses, and hardware-cost evaluation.

5.2. Results for the Unprotected Baseline Version

A) Signals Observed in GTKWave, see **Figure A1**.

Table 6. Results of RTL Simulation and GTKWave signal inspection.

Element	Value/observation
Test program used	sca_test_baseline—XOR + ADD over 16 inputs (inputs [0 to15]) with secret_fixed = 0Xbbbbbbbb
Number of traces generated	16 traces (one per loop iteration, i = 0 to 15)
Observation window	Centered on the trigger sca_dbg_trigger (XOR instruction encoded 0x00B54733—6 validated occurrences in GTKWave)
Target signals	sca_dbg_alu_op_a, sca_dbg_alu_op_b, sca_dbg_result_ex, sca_dbg_rf_rdata_a, sca_dbg_rf_rdata_b, sca_dbg_trigger
Key Feature	sca_dbg_result_ex—results of XOR (inputs[i] XOR 0xBBBBBBBB): direct correlation with HW model on the secrete value

The baseline implementation is used to characterize the initial leakage. The observed signals include the operands, intermediate results, control signals, and temporal markers.

Table 6 establishes the reference configuration, which is reproduced identically for the two other implementations: the same test program, the same 16 traces, the same temporal window centred on the trigger, and the same set of selected signals.

B) Generated Traces

The traces were generated in FST format using Verilator and subsequently converted to VCD format using fst2vcd for processing within the Python analysis pipeline. Each experimental campaign generated 16 traces, corresponding to one trace per loop iteration.

The analyzed signals were sca_dbg_result_ex, which served as the primary leakage point, sca_dbg_alu_op_a/b, and the synchronization trigger signals. The targeted operation was the XOR between the input and the secret value.

Table 7. Results of CPA, DPA and TVLA for the baseline version implementation.

Test	Indicator	Result of the baseline	Interpretation
CPA	Maximum correlation	|r| = 1. 0000 (The correct hypothesis is k = 0xBBBBBBBB; HW model over result_ex)	Maximum leakage—key recovered across all 4 bytes simultaneously with perfect correlation
DPA	Maximum average difference	|DoM| = 6.1746 (Partitioning by the MSB of the most significant byte)	Exploitable—strong separation between the two groups; secret directly exposed via "result_ex"
TVLA	|t| maximum	|t| = 0.4459 (Statistical limit of 16 traces)	TVLA below threshold: with only 16 traces, the Welch's t-test lacks statistical power, even for the fully leaking baseline.
Leakage threshold	4.5 for TVLA	Threshold 4.5 not met (|t| = 0.4459)	The vulnerability is est proven by the CPA (|r| = 1.0) and the DPA (|DoM| = 6.17), and not by the TVLA

The results in the first three rows establish the maximum-vulnerability reference against which the two countermeasures are evaluated: a perfect CPA correlation |r| = 1.0000, a clear DPA difference of means |DoM| = 6.1746, and a TVLA value that remains below the 4.5 threshold.

The results of the fourth row of the table clarify the apparent discrepancy between the latter two results. The fact that the TVLA remains below the threshold does not imply that the baseline implementation is leakage-free; rather, Welch's t-test requires a substantially larger number of traces to provide sufficient statistical discriminative power than the 16 traces.

Accordingly, CPA and DPA, which exploit a targeted leakage model, are used as the primary indicators for assessing the effectiveness of the countermeasures in this study, whereas TVLA is treated as a complementary rather than decisive metric for the present sample size (Table 7).

5.3. Results of the Protected Versions

5.3.1. Masking-Protected Version

The Boolean masking-protected version splits each sensitive data value into two randomized shares, (share_a = a ⊕ m) and (share_a = a ⊕ m, share_b = b ⊕ m). The same 16 input values and the same secret 0xBBBBBBBB used for the baseline implementation were retained to ensure a fair comparison. The results presented below were measured during the mask_share_a phase. The generated waves are shown in Figure A2.

Table 8. Results of CPA, DPA and TVLA for Boolean-masked version.

Indicator	Measured Value	Comment
Number of Traces	16	Total Number of Usable Traces
CPA	|r| = 0.4586 (mask_a)—Key not found	The maximum correlation occurs for an incorrect hypothesis, rather than for "0Xbb"
DPA	|DoM| = 1.2063	Approximately 80% reduction in the difference of means compared with the baseline 1.21
TVLA	|t| = 1.1632 (Below the threshold of 4.5)	Below the threshold: potential leakage is not detectable with 16 traces.
Area	1218 o (.text), +100 o soit +8.9% vs baseline	Pure software overhead: three *noinline* functions (*mask_a*, *mask_b*, and *recombine*)
Latency	206 cycles for 16 iterations (×2.6 vs baseline)	The cost of three implicit function calls.
Energy	~+15% (proportional to the number of cycles)	Incremental Cycle-Proportional Estimation

The results confirm the expected effect of the Boolean masking scheme. The CPA correlation decreases from 1.0000 to 0.4586 for the isolated mask_a share, and the resulting peak does not correspond to any consistent key hypothesis. The secret therefore remains statistically unrecoverable under the evaluated attack conditions. The DPA results exhibit the same trend, with an approximately 80% reduction in the difference-of-means metric.

The TVLA value of 1.1632 remains below the 4.5 threshold, as observed for the baseline implementation. With the present sample size, TVLA does not discriminate between the two implementations. This result should not be interpreted as evidence of the absence of residual leakage; rather, it further highlights the limited statistical power of the test when only 16 traces are available.

This security improvement is accompanied by a measurable implementation cost: an 8.9% increase in code size and a 2.6× increase in latency, primarily attributable to the three functions mask_a, mask_b, and recombine. The CPA correlation decreases from 1.0000 to 0.4586 for the mask_a share observed in isolation, and the resulting peak does not correspond to any consistent key hypothesis. Consequently, the secret remains statistically unrecoverable under the evaluated attack conditions. The DPA results exhibit the same trend, with an approximately 80% reduction.

The TVLA value of 1.1632 remains below the 4.5 threshold, as in the baseline

implementation. With this sample size, TVLA does not distinguish between the two implementations. This observation should not be interpreted as evidence of the absence of residual leakage; rather, it once again highlights the limited statistical power of TVLA when only 16 traces are available.

This security improvement incurs a measurable implementation cost: an 8.9% increase in code size and a 2.6× increase in latency, attributable to the three functions mask_a, mask_b, and recombine (Table 8).

5.3.2. Randomization-Protected Version

The temporal randomization-protected version inserts a variable delay (random_delay) prior to the sensitive operation, thereby misaligning the traces in the time domain. The same 16 inputs as the baseline were utilized. The values below distinguish between the aligned case (where the attacker has access to the trigger) and the misaligned case. The generated waves are shown in Figure A3.

Table 9. Results of CPA, DPA and TVLA for the time-randomized protected version.

Indicator	Measured Value	Comment
Traces number	16	One trace is recorded per loop iteration ($i = 0 \ldots 15$), corresponding to the number of XOR operations executed in the simulation. For a conventional CPA/DPA, several thousand traces with random inputs are required; this sample size is illustrative but insufficient.
CPA	|r| = 1.0000 (trigger-aligned traces; the key 0xBBBBBBBB successfully recovered).	If the attacker has access to the trigger signal (*random_delay*), the alignment becomes trivial and the leakage remains intact. Without this reference point, the desynchronization (ranging from 34 to 164 cycles) significantly reduces the correlation.
DPA	The value of |DoM| is 6.1746 when aligned, but it drops sharply without alignmentsans	Observed statistical difference
TVLA	|t| > 4.5 avec alignement parfait ; |t| < 4.5 sans alignement	Welch's t-test between two groups of traces (e.g., fixed vs. random inputs). Leakage is detectable if the traces are synchronized with the sensitive operation. Temporal misalignment causes the t-statistic to drop below the threshold.
Area	No hardware overhead (software implementation). Baseline Ibex core: ~2000 LUTs, ~1000 FFs (typical configuration).	Randomization is implemented via a software loop (random_delay). No additional logic is instantiated in the FPGA, maintaining the original area of the unmodified Ibex core.
Latency	An average overhead of 82.75 cycles per iteration results in an additional total of 1324 cycles across the 16 iterations.	Sum of measured deviations $X_i - T_i = \sum(34 + 26.d_i) = 16 \times 34 + 26 \times 30 = 1324$ cycles. This total represents the added delay between the trigger (random_delay) and the completion of the XOR operation across the 16 iterations. Excluding the fixed component (d = 0), the latency strictly due to jitter is 26 × 30 = 780 cycles.
Energy	~+10 % (proportional to the number of additional delay cycles)	Switching activity estimation

Table 9 illustrates an important property of temporal randomization: its protection is conditional. If the attacker has access to the trigger signal (*random_delay*), trace alignment becomes trivial, and the metrics match the baseline exactly (|r| = 1.0000, |DoM| = 6.1746).

Without this reference point, the desynchronization of 34 to 164 cycles drops the CPA, DPA, and TVLA metrics below exploitable thresholds. Unlike masking, this countermeasure requires no hardware area overhead, as it is purely software-based. However, its latency cost is significant: an average of 82.75 cycles added per iteration, totaling 1.324 cumulative cycles across the 16 iterations, of which 780 cycles are directly attributable to the random jitter alone once the fixed component is removed.

6. Results Analysis

Figure 5 depicts the overall results. It shows the trade-off between security and overhead for each of the three experimental versions. The evaluation metrics include CPA/DPA resistance, Test Vector Leakage Assessment (TVLA) score, area, and latency. These metrics serve to identify the most balanced solution—specifically, the design that mitigates leakage without introducing excessive overhead to the target architecture.

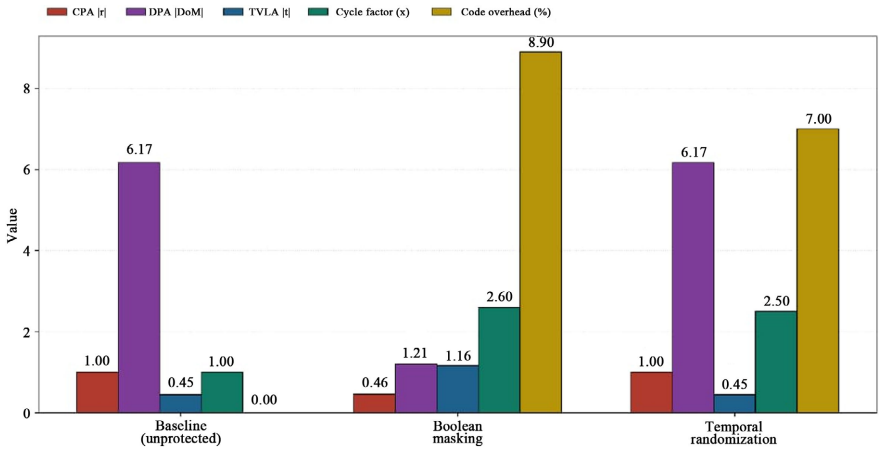


Figure 5. Overall Results comparison.

The experimental results are interpreted with respect to the initial research objectives. The reduction in the Correlation Power Analysis (CPA) correlation coefficient from 1.0000 to 0.4586 for the masking scheme confirms a distinct improvement in side-channel attack resistance. This security enhancement must be analysed in conjunction with the associated hardware and performance costs. Specifically, the masking countermeasure offers the optimal trade-off between protection and performance overhead (a 2.6× increase in clock cycles).

The findings corroborate and validate the established principles in the literature. Differential Power Analysis (DPA) demonstrates the statistical exploitability of power consumption traces, while CPA formalizes the application of power leak-

age models. Furthermore, the Test Vector Leakage Assessment (TVLA) provides a standardized statistical detection methodology. Finally, the evaluated masking and randomisation countermeasures align with established techniques designed to mitigate data-dependent information leakage as in [11].

7. Conclusions

This paper has presented a structured methodology for investigating side-channel attacks (SCAs) on an Ibex RISC-V core. The literature review has demonstrated that attacks exploiting power consumption or logic transitions can compromise algorithmically secure implementations. The proposed methodology relies on register-transfer level (RTL) simulation, internal signal observation, Correlation Power Analysis (CPA), Differential Power Analysis (DPA), and Test Vector Leakage Assessment (TVLA), alongside a comparative evaluation of multiple countermeasures.

Experimental results confirm that Boolean masking is the most effective countermeasure against CPA ($|r| = 0.4586$ versus 1.0000 for the baseline). Furthermore, the proposed framework successfully identifies the core operational components: signal selection, trace generation, point-of-interest determination, implementation of protected versions, and trade-off evaluation. The significance of this work lies in its methodological contribution. It integrates hardware security, RISC-V architecture, open-source tools, and experimental pedagogy, establishing a reproducible analysis tailored to the constraints of embedded systems.

Future research can focus on the validation of the proposed designs on an FPGA board using actual power consumption measurements; the automation of the extraction of points of interest (POIs) from Value Change Dump (VCD) files.

Author Contributions

Conceptualization, Yahaya COULIBALY and Ousmane TRAORE; methodology, Ousmane TRAORE; software, Issa TRAORE; validation, Yahaya COULIBALY, Daouda TRAORE, Issa TRAORE; formal analysis, Ousmane TRAORE and Yahaya COULIBALY; resources, Issa TRAORE and Yahaya COULIBALY; data curation, Ousmane TRAORE Daouda TRAORE; writing—original draft preparation, Yahaya COULIBALY and Ousmane TRAORE; writing—review and editing, Yahaya COULIBALY and Issa TRAORE; visualization, Ousmane TRAORE and Issa TRAORE; supervision, Yahaya COULIBALY; project administration, Yahaya COULIBALY; funding acquisition, Issa TRAORE. All authors have read and agreed to the published version of the manuscript.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Kocher, P.C. (1996) Timing Attacks on Implementations of Diffie-Hellman, RSA,

DSS, and Other Systems. In: Koblitz, N., Ed., *Advances in Cryptology—CRYPTO*'96, Springer, 104-113. https://doi.org/10.1007/3-540-68697-5_9

[2] Kocher, P., Jaffe, J. and Jun, B. (1999) Differential Power Analysis. In: Wiener, M., Ed., *Advances in Cryptology—CRYPTO* 99, Springer, 388-397. https://doi.org/10.1007/3-540-48405-1_25

[3] lowRISC (2026) Ibex: An Embedded 32bit RISC-V CPU Core. Documentation Officielle Ibex.

[4] lowRISC (2026) Ibex RISC-V Core. Dépôt GitHub Officiel.

[5] Brier, E., Clavier, C. and Olivier, F. (2004) Correlation Power Analysis with a Leakage Model. In: Joye, M. and Quisquater, J.J., Eds., *Cryptographic Hardware and Embedded Systems—CHES* 2004, Springer, 16-29. https://doi.org/10.1007/978-3-540-28632-5_2

[6] Goodwill, G., Jun, B., Jaffe, J. and Rohatgi, P. (2011) A Testing Methodology for Side-Channel Resistance Validation. NIST Non-Invasive Attack Testing Workshop.

[7] Coulibaly, Y., Traoré, O. and Traoré, D. (2026) Analysis of Countermeasures Techniques against Side-Channel Attacks on Electronic Circuits: A Review. *International Journal of Engineering and Computer Science*, **15**, 28907-28913. https://doi.org/10.18535/ijecs.v15i08.5676

[8] Kindgren, O. (2026) FuseSoC User Guide. Documentation Officielle. https://fusesoc.readthedocs.io/en/stable/user/

[9] Verilator Project (2026) Verilator User Guide. Documentation Officielle.

[10] GTKWave Project (2026) GTKWave Wave Analyzer User's Guide. Documentation Officielle.

[11] Tiri, K. and Verbauwhede, I. (2004) A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. *DATE* 2004, *IEEE Computer Society*, Washington, 27 June-2 July 2004, 1530-1591.

Appendices: GTKWave Screenshot

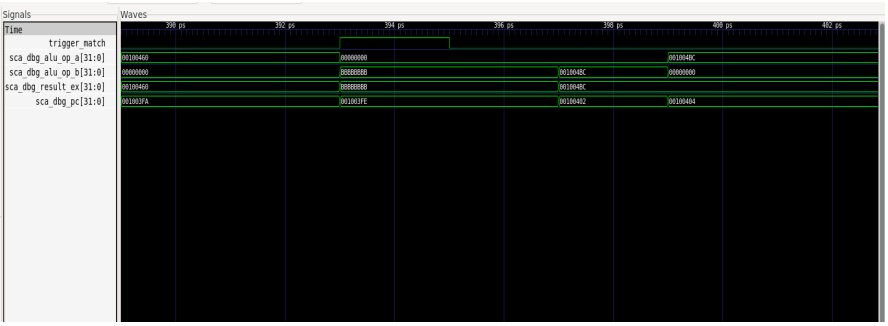


Figure A1. Baseline wave (XOR detection (trigger_match, result_ex = 0xBBBBBBBB, PC = 0x001003FE).

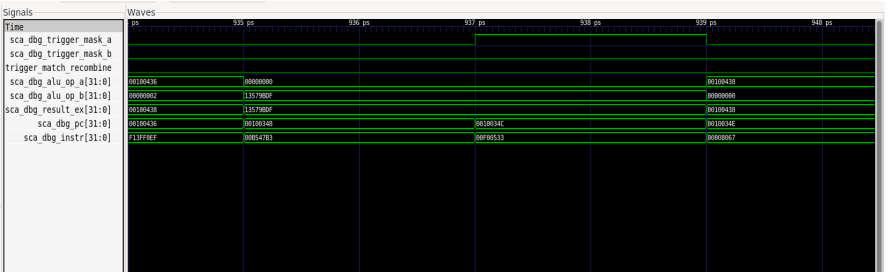


Figure A2. Masking Wave (Positive-going edge of sca_dbg_trigger_mask_a (share_a = 0x13579BDF).

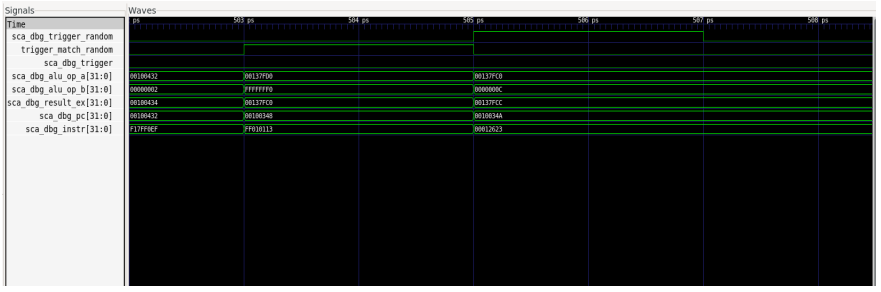


Figure A3. Temporal randomization Wave, sca_dbg_trigger_random et sca_dbg_trigger (variable delay).