NG RAD: Next-Generation RISC-V ASIC Design

Anna Craig, Alexander Defalco, Joshua Joseph, Daniel Odi, Alyssa Pinnock
University of Central Florida, Department of Computer and Electrical Engineering, Orlando, Florida, 32816, U.S.A.

***Abstract*—Side-channel attacks exploit physical characteristics of hardware, such as power consumption and timing behavior, to extract sensitive information. The need for secure processor designs has become increasingly critical and serves as the primary motivation for this project. This work presents a secure processor architecture that integrates side-channel leakage mitigation techniques into a baseline RISC-V Ibex core. The design extends the processor with SCALER-based modifications, including Boolean masking, noise injection, and clock randomization to reduce susceptibility to attacks. A comprehensive verification methodology is implemented using the Ibex DV environment and co-simulation with the Spike reference model, along with gate-level simulation and latency measurement analysis. Results demonstrate that the modified design maintains functional correctness while introducing measurable tradeoffs in performance and timing behavior. This work highlights the full ASIC design flow for secure hardware and provides a foundation for future development.**

***Index Terms*—Side-Channel Attacks, Boolean Masking, Noise Generating Circuits**

I. Introduction

Side-Channel leakage has emerged as a significant threat to modern digital hardware, enabling attackers to infer sensitive data by observing physical characteristics such as power consumption, electromagnetic emissions, and timing variations. Unlike traditional attacks, timing attacks exploit unintended information leakage at the hardware level. As a result, systems that are functionally correct and logically secure may still be vulnerable in practice.

This challenge underscores a critical limitation in conventional processor design methodologies, which primarily focus on functional correctness, performance, and area optimization. While these metrics ensure that a processor behaves under normal operation, they do not guarantee resistance to side-channel attacks.

To address this gap, secure processor design must incorporate countermeasures that actively reduce of obscure observable leakage while maintaining correct functionality. This requires a holistic approach that includes architectural modifications, post-synthesis validation, and well-defined engineering specifications that guide both design decisions and evaluation criteria. In this work, we explore these challenges through the design and evaluation of a modified RISC-V professor that integrates side-channel mitigation techniques, with an emphasis on maintaining both functional integrity and security across the ASIC design flow.

This work was supported by Northrop Grumman Corporation.

A. Project Motivation

The growing use of hardware security-critical systems has made resistance to side-channel attacks a fundamental requirement.

A key challenge is ensuring these protections remain effective after synthesis. Techniques introduces at the architectural level may be altered or removed during optimization, making post-synthesis evaluation necessary to verify their effectiveness.

This project is also motivated by the importance of understanding the full ASIC design flow. As industry demand for end-to-end hardware knowledge increases, this work provides hands-on experience across design, synthesis, and verification in a security focused context.

II. Background & Design Objectives

Side-channel attacks exploit physical effects such as power consumption, switching activity, and timing variation to infer sensitive information from otherwise functionally correct hardware. Because of this, secure processor design must be evaluated not only for correctness and performance, but also for how well it suppresses or obscures unintended leakage throughout the implementation flow.

The SCALER project is motivated by two parallel goals. The first is to explore practical techniques for reducing side-channel leakage in a RISC-V processor core. The second is to carry those modifications through a realistic ASIC-oriented workflow so the design can be evaluated beyond RTL alone. In this work, the design objectives are therefore centered on extending a baseline processor with leakage-reduction mechanisms while preserving ISA-correct behavior, compatibility with an established verification flow, and practical implementability through synthesis and post-synthesis validation.

A. Ibex as the Baseline Core

SCALER is built on the open-source Ibex RISC-V core, selected for its balance of practicality, flexibility, and verification support. Ibex is written in SystemVerilog, includes a mature verification environment, and emphasizes verifiability and maintainability, making it a strong foundation for architectural modification [1].

Ibex is also a useful security-oriented baseline because it supports Physical Memory Protection (PMP), which allows configurable memory access regions and privilege enforcement [1], [2]. In the original report, this was identified as a meaningful architectural feature for exploring protected memory behavior and reducing information leakage through memory access patterns. Combined with its existing verification infrastructure, these features make Ibex a realistic platform for studying side-channel-aware processor extensions.

B. Side-Channel Mitigation Goals

The primary security goal of SCALER is to reduce the correlation between sensitive internal activity and externally

observable behavior while maintaining correct processor functionality. Masking is a widely used side-channel countermeasure, and this work explores first-order Boolean masking, where a sensitive value is represented with two shares so that no single observation directly reveals the secret [3]. It is also important to note that nonlinear operations require special care, making Biryukov's optimal first-order Boolean masking method for secure AND and OR constructions especially relevant to this implementation [4].

In addition to masking, the project also considers hiding-style countermeasures. We explore random noise injection as a way to reduce the signal-to-noise ratio seen by an attacker and make statistical extraction more difficult, while dynamic frequency scaling is used to introduce temporal misalignment between traces [5], [6]. These techniques are especially relevant in combination: prior work has explored secure gate replacement and related masking and hiding methods on a RISC-V core, while post-synthesis masking insertion through synthesis-script-based gate replacement was also identified as a practical implementation path for this project [6], [7]. Together, these goals make the mitigation strategy both architectural and flow-aware: the countermeasures must reduce leakage in principle and remain meaningful after synthesis in practice.

C. Engineering Specifications & Evaluation Metrics

The engineering specifications for the NG RAD processor are defined to balance security, performance, and implementation feasibility across the ASIC design flow. This design is based on the RV32I instruction set to ensure simplicity and deterministic execution, while operating within a constrained voltage range of 1.0 V to 1.25 V and a temperature range of 0°C to 125°C to maintain stable electrical characteristics for side-channel resistance. A 45 nm process node is selected to provide a practical tradeoff between performance and leakage control. Power integrity is enforced through an IR drop limit of 5%, and the system targets an average operating frequency of at least 15 MHz to meet throughput requirements without introducing excessive dynamic power variation. From a security perspective, first-order masking is implemented as a baseline mitigation strategy, while architectural constraints such as limiting IPC to 1.2 ensure predictable timing behavior.

Evaluation metrics are aligned with these specifications and focus on verifying both functional correctness and security preservation throughout the design flow. This includes validating that side-channel mitigation structures remain intact after synthesis, ensuring stable timing behavior, voltage and temperature variations, and measuring performance characteristics such as latency and frequency. Together, these metrics provide a comprehensive framework for assessing whether the design meets its security objectives while remaining practical for real-world ASIC implementation.

III. SCALER Extension Design

The SCALER extension introduces a set of RTL and gate-level modifications to the baseline Ibex core to reduce the correlation between sensitive data and observable side-channel leakage. First, multiple noise generators are incorporated to introduce spurious switching activity, lowering the signal-to-noise ratio available to an attacker observing power or electromagnetic emissions. In addition to the added noise, SCALER integrates Boolean masking into the design by representing each value as two randomized shares, which are propagated throughout the entire processor. Lastly, the design integrates randomized clock behavior, causing the disruption of temporal alignment. When used in conjunction, these modifications create a layered defense against side-channel analysis while maintaining the integrity of the original Ibex processor.

TABLE I
Engineering Specifications

Parameter	Specification	Priority
Instruction Set	RV32I	High
Operating Temperature Range	0°C – 125°C	Low
Operating Voltage Range	1.0 V – 1.25 V	High
Process Node	∼45 nm	High
IR Drop	$\leq 5\%$	Low
Average Frequency	≥ 15 MHz	High
Masking Order	First-order minimum	High
Instructions per Cycle (IPC)	≤ 1.2	Medium
Cache Size	≥ 2 KB	High
Timing Failure Count	0 Faults	High
Interrupt Latency	≤ 12 cycles	Medium
Leakage Power (25°C)	≤ 5 mW	High

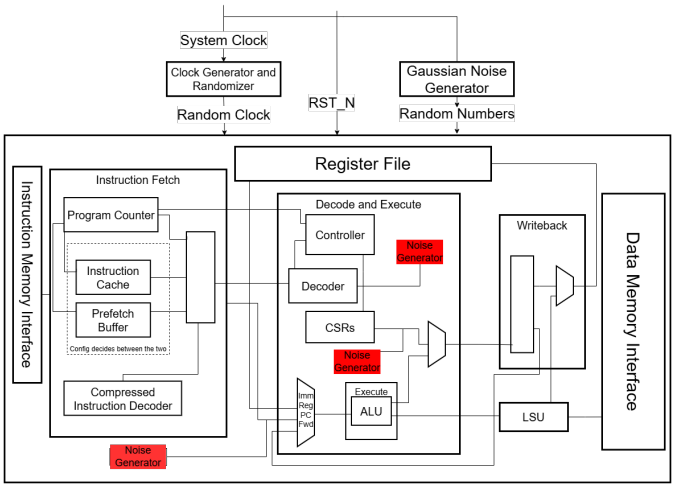

Fig. 1. Ibex pipeline with random clock and noise generators, adapted from [1].

A. Boolean Masking Modifications

To combat data-dependent leakage within the processor, our design integrates first-order Boolean masking with the baseline Ibex core. Our approach is based on that of [7]. This causes every logical value to be split into 2 uncorrelated values that can later be recombined to derive the original data. We do this by XORing the original(unmasked) data with a pseudo-randomly generated mask. The first share will be equal to the random mask, while the other is the output of the

previously mentioned XOR operation. Once the transformation is complete, the shares are statistically independent shares that individually reveal no information about the underlying value.

Of course, to maintain the integrity of the original processor, the underlying combinational logic also needs to be transformed. While linear operations(XOR) can work on the masked shares, non-linear operations(AND, OR, etc.) require additional care in order to remain effective. As shown in Fig. 2, simple AND and OR gates are translated into composite logic structures that operate on masked shares, producing two outputs that together reconstruct the original value. The transformations undergone by these non-linear operations ensure that intermediate computations do not introduce first-order leakage, while also having outputs that adhere to the two-share format mentioned previously.

Instead of being limited to specific functional units, the masking transformations will be applied to the entire processor datapath to further strengthen side-channel defense. This means that once original values are sent into the SCALER design, they will be split and kept as separate shares throughout the entire pipeline. Masking the processor in its entirety minimizes the chances of unprotected intermediate values being exposed, greatly hindering the effectiveness of first-order side-channel attacks.

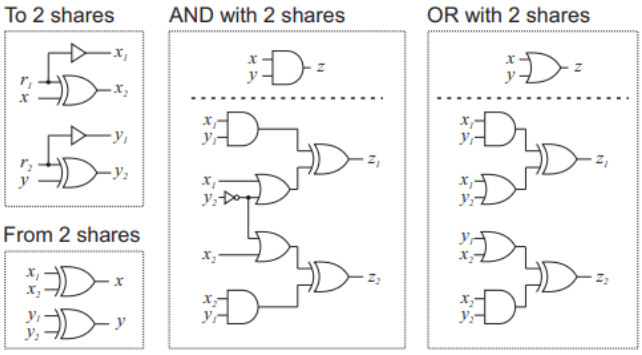

Fig. 2. Share splitting logic with non-linear operations on shared values [7].

B. Noise Injection and Clock Randomization

On top of Boolean masking, SCALER adds noise generation completely independent from program data by introducing additional switching activity. To generate these pseudo-random values, SCALER implements cellular automata shift registers (CASRs) and ring oscillators. The implemented noise generators inject spurious transitions into selected regions of the datapath. These additions are effective in defending against side-channel attacks since the added noise lowers the signal-to-noise ratio, making statistical extraction of sensitive information more difficult for an attacker.

To complement spatial noise injection, SCALER also incorporates temporal variability through its randomized clock behavior by inserting pseudo-random delays within the execution flow. This significantly disrupts the temporal alignment

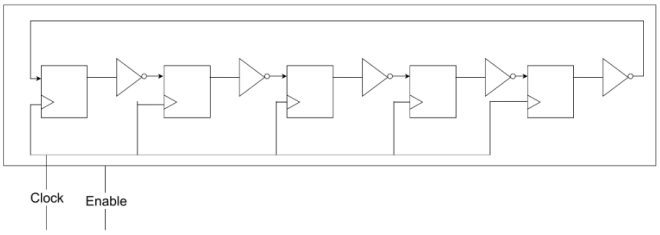

Fig. 3. Ring oscillator logic for noise injection.

of repeated operations, causing externally observed traces to become misaligned across executions. The inconsistencies brought on by the randomized clock cycles reduce the effectiveness of correlation-based side-channel attacks that rely on consistent timing.

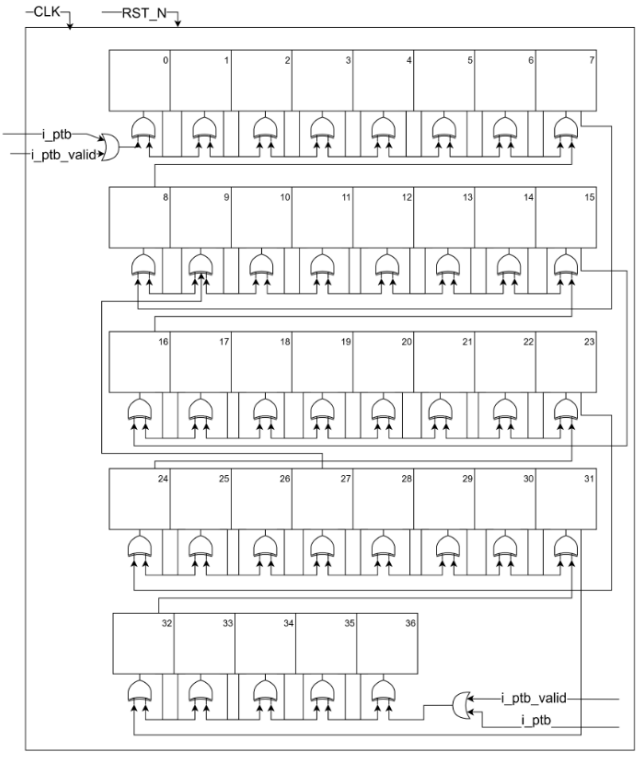

Fig. 4. Cellular Automata Shift Register for noise injection.

C. Gate-Level/Netlist Security Logic Preservation

Upon completion of Boolean Masking on the Ibex Core, measures had to be taken to ensure that the post-synthesis gate transformations were not undone in later steps of the RTL-to-GDS flow. Instances in previous research have demonstrated that EDA tool synthesis optimizations can compromise the security of a chip during the synthesis process by undoing masking measures and replacing them with speed and/or area-efficient equivalents of the original logic. This presents a significant challenge, as the very optimizations that improve

performance can inadvertently expose sensitive data paths that were deliberately obfuscated. Synthesis of our core in this project occurs in two distinct parts. The first is the initial masking discussed earlier, in which gates are replaced with their secure counterparts as described earlier in this section, and the second is the integration of this masked core into a larger SoC.

In the first synthesis stage, preservation of the security logic is ensured by omitting the syn_opt step entirely. As discussed in earlier sections, Boolean masking of the Ibex Core occurs after the syn_map step, which precedes syn_opt. Executing the syn_opt step would result in Cadence Genus applying its optimization algorithms to the Boolean netlist, which is precisely what we aim to avoid. Since these optimizations operate without any awareness of the security constraints imposed by the masking scheme, they risk collapsing or restructuring the masked logic in ways that would defeat its purpose. Therefore, removing the syn_opt step from the synthesis script was the most straightforward and reliable course of action to maintain the integrity of the masked netlist.

For the second synthesis stage, the goal is to connect the masked core to the broader SoC while leaving the core itself completely unmodified. The remaining components of the SoC should proceed through syn_generic, syn_map, and syn_opt as they normally would. To accomplish this, after having Genus read in the netlist of the masked core and prior to the synthesis step, a database variable was configured to instruct Genus to leave the masked core untouched. This approach ensures that all modules external to the masked core can still be optimized freely, as there is no reason to allow those modules to unnecessarily consume additional area and time within the datapath while also removing any risk of inadvertently altering the carefully constructed security logic of the core.

IV. Verification and Experimental Methodology

The verification methodology for this project is intended to confirm both functional correctness and implementation fidelity after the SCALER modifications are applied to the Ibex core. Since the design includes security-oriented structural changes, RTL simulation alone is not sufficient. A processor may appear correct before synthesis, yet still experience meaningful changes after logic mapping and optimization. For that reason, this work avoids treating the post-synthesis design as a black box and instead uses gate-level simulation to directly validate the synthesized implementation.

Verification is performed by comparing the modified design against the expected behavior of the original Ibex core. Representative RTL regressions, interrupt-focused tests, and compiled C workloads are used to check that the processor continues to execute valid programs and preserve ISA-visible behavior. The purpose of this flow is to ensure that the added masking and hiding logic improve security characteristics without breaking the original architectural functionality.

Gate-level simulation is particularly important because it confirms that the synthesized netlist, not just the RTL description, still behaves correctly. This provides confidence that the secure design intent survives beyond high-level modeling and remains present in the implemented core.

A. Ibex DV and Spike Co-Simulation Flow

The Ibex DV/co-simulation environment is a UVM-based verification framework that executes RISC-V programs on the DUT while monitoring architectural behavior at runtime. At a high level, the environment instantiates the processor testbench infrastructure, instruction/data memory interface agents, interrupt and control stimulus components, protocol monitors, and a scoreboard/checking path for architectural consistency. Through a co-simulation bridge, Spike executes the same instruction stream as a software golden model leveraging RVFI, allowing cycle-progressed DUT execution to be checked against ISA-correct reference behavior. This setup provides a practical link between microarchitectural simulation and architecture-level correctness checking.

Architectural validation is performed by running each test with a fixed seed to generate deterministic stimulus and an executable program image. During the baseline run, the flow captures both DUT architectural trace information and a Spike reference trace for the same program. The identical binary is then executed on the target implementation under evaluation (modified RTL), ensuring that any observed differences come from implementation behavior rather than workload mismatch.

For comparison, traces are normalized to remove non-architectural fields (for example simulator timing columns) and aligned at the instruction level. The checker then performs instruction-by-instruction equivalence against the baseline reference stream. A run is classified as passing when architectural execution is identical; when traces differ only in simulation termination conditions, prefix-equivalent behavior is accepted if the common execution window matches exactly. This methodology provides a robust ISA-visible correctness criterion while remaining practical for large seeded regression campaigns.

TABLE II
Representative RTL Regression Tests (Functional)

Test Name	**Component Tested**	**Purpose**
RISC-V Arithmetic Basic Test	ALU datapath, decoder, writeback	Verifies integer arithmetic and logic behavior, including basic dependency handling.
RISC-V RV32IM Instruction Test	RV32I/M decode and execute paths	Exercises a broad instruction mix to validate architectural state updates.
RISC-V Random Instruction Test	Pipeline interactions under random streams	Stresses corner-case instruction sequences using seeded randomized programs.
RISC-V Unaligned Load/Store Test	Load/store unit and memory access path	Validates handling of unaligned memory operations and related control behavior.
RISC-V Illegal Instruction Test	Trap/exception entry logic	Confirms illegal-instruction detection and correct trap entry response.

B. RTL Regression Strategy

RTL verification is performed using a seeded regression workflow that combines directed and randomized RISC-V programs. Directed tests target specific architectural behaviors (for example arithmetic correctness, exceptions, privilege transitions, and control-flow edge cases), while randomized programs improve coverage of instruction interactions and corner-state sequences that are difficult to enumerate manually.

Each test is executed over multiple seeds to evaluate stability and repeatability under varied execution traces. For every run, the flow records simulation status, architectural trace output, and summarized pass/fail metadata. Functional closure is determined from aggregate pass rate, consistency with baseline architectural behavior, and reproducibility across seeds. This strategy provides both breadth (cross-category workload coverage) and depth (seed diversity), while preserving deterministic debug paths for failure triage.

Representative functional tests are listed in Table II, while privilege and interrupt-focused tests are listed in Table III.

TABLE III
REPRESENTATIVE RTL REGRESSION TESTS (PRIVILEGE AND INTERRUPT)

Test Name	Component Tested	Purpose
RISC-V Single Interrupt Test	Interrupt detect/entry/return flow	Measures baseline interrupt handling correctness for isolated interrupt events.
RISC-V Multiple Interrupt Test	Repeated interrupt service path	Checks robust servicing under periodic and repeated interrupt arrivals.
RISC-V Nested Interrupt Test	Nested trap state handling	Evaluates nested interrupt behavior and correct save/restore ordering.
RISC-V Interrupt CSR Test	Interrupt and CSR interaction path	Validates interrupt handling while control/status registers are actively modified and observed.
RISC-V Interrupt WFI Test	WFI wake-up and interrupt entry path	Checks wake-from-idle behavior and correct handler entry timing around wait-for-interrupt states.

C. Gate-Level Simulation Setup

Gate-level verification is utilized to simulate the synthesized masked Ibex core netlist in place of the RTL core design with functional timing settings to validate post-synthesis architectural correctness. Since our secure implementation utilizes a dual-rail masking setup, a netlist wrapper is used to map the conventional Ibex testbench interfaces into their masked signal counterparts, vital for representing masked signals and reconstructing ISA-relevant outputs for verification.

In a goal to maintain comparability with RTL verification, gate-level runs would ideally execute the same test binaries used in baseline and modified RTL campaigns. However, an oversight with the RVFI signals being excluded from the synthesized design prevents this, and instead is a future objective. Rather, we simply concern ourselves with the snythesized, masked (modified) Ibex core's ability ro run RISC-V compiled binaries through the design with valid outputs.

Various C programs are written with the *ibex_simple_system.h* header file, compiled into binaries, and converted into VMEM utilizing a shell scripts (due to an inability to install srec_cat on the UCF VLSI servers). These binaries are then run on the modified Ibex core, while the original program is also written in a native C format without the ibex header, allowing us to compare functional equivalence. The gate-level validation subset and the intent of each test are summarized in Table IV.

TABLE IV
GATE-LEVEL C PROGRAM TESTS EXECUTED VIA VMEM

Test Name	Component Tested	Purpose
Hello Program Test	Program load path, fetch/decode, basic execution flow	Validates end-to-end C-to-VMEM bring-up and confirms successful execution of a minimal program.
Fibonacci Sequence Test	Integer ALU, loop control, memory accesses	Validates iterative arithmetic and control-flow correctness over repeated loop execution.
CRC32 Hashing Test	Bitwise/shift operations and data path integrity	Checks correctness of checksum-oriented computation with heavy bit-manipulation behavior.
The Sieve of Eratosthene	Nested loops, branches, and memory traversal	Stresses sustained control-flow and memory-access behavior in algorithmic workloads.

D. Latency Measurement Methodology

Interrupt latency is measured using controlled interrupt-injection runs on interrupt-capable workloads that are prepackaged in the Ibex verification environment.. In this project, the five latency-focused tests we have focused on are: *RISC-V Single Interrupt Test*, *RISC-V Multiple Interrupt Test*, *RISC-V Interrupt WFI Test*, *RISC-V Nested Interrupt Test*, and *RISC-V Interrupt CSR Test* (listed in Table III). Within our testbench, we include a set of modular stimulus parameters, including initial interrupt delay, inter-interrupt spacing, and maximum interrupt count, providing repeatable stress conditions across seeds.

For each interrupt event, the framework records the interrupt assertion cycle and the handler-entry cycle. Latency is computed as

$$L_{\mathrm{irq}} = C_{\mathrm{handler\_entry}} - C_{\mathrm{irq\_assert}}.$$

Event-level data are aggregated per test and seed to report minimum, maximum, average, and percentile behavior. This captures both nominal interrupt response and tail-latency effects, enabling consistent comparison of interrupt responsiveness across workload classes and verification runs.

V. RESULTS

This section presents the experimental results obtained from synthesizing and validating the modified Ibex Core after the SCALER security extensions were introduced. The results are organized to evaluate the design from four complementary perspectives: architectural correctness, post-synthesis functional

fidelity, interrupt latency behavior, and synthesis outcomes. The verification-oriented subsections demonstrate that the design maintains functional correctness consistently across RTL simulation and gate-level execution, while the synthesis results confirm that the design meets the necessary timing and area requirements to proceed to physical implementation.

A. Synthesis Results

As stated in prior sections, generating the netlist for the masked core design required two distinct stages of synthesis: the initial creation of the Boolean Masked core and its subsequent integration into the broader SoC. Since the second synthesis stage represents the point at which the netlist is fully realized, it is during this step that the Synopsys Design Constraints (SDC) file is read and used to generate both timing and area results.

Timing is among the most critical outcomes of synthesis, as it determines whether the circuit is capable of operating reliably at the frequency specified in the SDC file. Failure to meet timing implies that combinational logic paths cannot settle to a valid output before the next active clock edge, which prevents flip-flops from reliably capturing and synchronizing results. This is particularly consequential in a masked design, as the secure gate replacements introduced during Boolean Masking inherently increase the propagation delay of every combinational datapath in the circuit. As discussed in prior sections, each secure gate introduces additional logic depth relative to its unprotected equivalent, which extends the critical path and places greater pressure on timing closure. Fine-tuning the design to achieve a positive slack on the critical path is therefore essential to ensuring the core can execute instructions correctly and securely at the target frequency.

Area is an equally important metric, as it reflects the silicon resources consumed by the design and has direct implications for fabrication cost and physical implementability. The introduction of Boolean Masking is expected to significantly increase the area of the core relative to its unprotected counterpart, given that each original gate is replaced by multiple secure gate instances as described in prior sections. Ensuring that the resulting area remains within acceptable bounds is necessary for the design to be viable for physical implementation and eventual tape-out. Table V present the timing and area results achieved from the successful synthesis runs, demonstrating that the masked Ibex Core meets timing at the target frequency while providing a quantitative characterization of the area overhead introduced by the masking scheme.

TABLE V
SECURE IBEX CORE TIMING RESULTS

Core	**Secure Ibex**
Period (ns)	47
Frequency (MHz)	∼21
Critical Path (ns)	46.336
Slack (ns)	0.544
Area (μm^2)	190,605.150

B. RTL Functional Verification Results

The RTL functional verification results demonstrate that all implemented modules operate according to their intended specifications under controlled testbench conditions. Verification was performed primarily using EDA playground, configure with SystemVerilog/Verilog language and Icarus Verilog 12.0.

The `casr37` module demonstrated correct reset behavior, enable gating, and neighbor-based state evolution, with proper perturbation handling and the expected one-cycle sequential delay. The `clkgen` module correctly generated a randomized clock, with LFSR-driven edge suppression and valid perturbation timing. The `lfsr43` module exhibited correct seed initialization, shift-register operation, XOR-based output generation, and periodic valid pulses every 32 cycles. The `lfsr8` module validated correct feedback polynomial implementation, controlled perturbation mixing, and avoidance of illegal states. The `ring_oscillator` module was verified for structural correctness and integration, with full functional validation deferred to post-synthesis analysis.

Overall, all modules exhibited deterministic reset behavior, state stable transitions, and correct implementation of security-related logic, establishing readiness for for gate-level verification and further evaluation of side-channel mitigation effectiveness.

C. Ibex DV and Co-Simulation Execution Results (Baseline vs. Pre-Synthesis Modified Core)

The Ibex DV/Co-sim environment was utilized in order to effectively run the pre-packaged Ibex tests before and after applying changes to the core, while still being prior to synthesis. In this series of executions, 8 seeds were executed on each of the 5 tests showcased in Table II. The results can be found in Table VI.

We see that for 4 out of 5 tests on all 8 seeds, all tests pass with flying colors, implying that their memory traces were all executed identically. However, with `riscv_rand_instr_test`, 4 seeds pass and 4 seeds "fail" the original processed workflow. After some analysis, we realized that our testing methodology focused on the Ibex core compared to the SPIKE golden model, which leverages RVFI signals to control CSRs, which dictates the exact branching execution path. The 4 "failures" all stemmed from a different branch being taken, however all instructions executed by the memory trace were still compliant with the RISC-V ISA.

D. Gate-Level Simulation Results

For our Gate-Level Simulations, we focused on running a few basic C programs through the RISC-V GCC toolchain and ran them on the synthesized ibex core. As showcased in Table VII, All programs compiled and tested were able to execute with the desired output accordingly, allowing for us to confirm and be confident in the validity of the fact that the baseline Ibex functionality is still infact, preserving the bare minimum RV32I instruction set.

TABLE VI
REPRESENTATIVE RTL REGRESSION TESTS (FUNCTIONAL)

Test Name	Seeds Tested	Seeds Passing
RISC-V Arithmetic Basic Test	8	8
RISC-V RV32IM Instruction Test	8	8
RISC-V Random Instruction Test	8	4
RISC-V Unaligned Load/Store Test	8	8
RISC-V Illegal Instruction Test	8	8

Note: Each test was executed across 8 seeds; partial passing indicates some randomized runs did not directly match the baseline Ibex SPIKE trace.

TABLE VII
GATE-LEVEL SIMULATION RESULTS

C Program Name	RISC-V GCC Compiled	Executed on Ibex	Matched C Compiler
Hello_test.c	✓	✓	✓
Fibonacci.c	✓	✓	✓
CRC32.c	✓	✓	✓
Sieve.c	✓	✓	✓

Note: Each C test was compiled with the RISC-V GCC toolchain, executed on the Ibex-based design, and checked for agreement with the expected C-level behavior.

E. Interrupt Latency Measurement Results

We evaluated interrupt latency by running a subset of the interrupt-enabled tests pre-packaged with Ibex on the pre-synthesis version of the core, with representative tests shown in Table II. For these experiments, the interrupt configuration was set to `Interrupt Spacing` = 300 cycles, `First interrupt delay` = 500 cycles, and `Max interrupts per test` = 200. Each test was then executed across 9 seeds, producing the latency results summarized in Table VIII.

Across all measured tests, the minimum observed interrupt latency was 6 clock cycles, while the maximum was approximately 44 clock cycles. The average latency, aggregated by test and seed, ranged from 10.7 to 16.8 clock cycles. Taken together, these results produced an overall average interrupt latency of 15.1 clock cycles, which we report as the representative latency for the design.

F. Observed Tradeoffs and Limitations

During the implementation phase of this project, several necessary concessions were made in order to successfully apply Boolean Masking to the Ibex Core. These trade-offs are inherent to the nature of the masking methodology employed and have measurable consequences on the performance and area characteristics of the resulting design. The library of secure gates used in this work is based on the constructions presented by Biryukov [4], which is comprised of only five elementary gate types: AND2, XOR2, OR2, INV, and DFF. This represents a substantially constrained gate set, as it excludes gates with fan-in greater than two as well as any complex or compound logic gates. In a standard, unprotected synthesis flow, such higher-complexity gates offer notable improvements in both propagation delay and silicon area by consolidating multiple logic operations into a single cell. However, because no formally verified secure equivalents exist for these gate types within the adopted library, their use cannot be permitted in a masked design without potentially introducing exploitable information leakage. Consequently, during the generic synthesis step, the design must be constrained to only those gates for which secure equivalents are available, forgoing the area and timing benefits that a full standard cell library would otherwise provide.

This limitation has two significant and compounding implications for the synthesized design. First, because each secure gate introduces a minimum of two additional gates into the datapath to accommodate the masking scheme, the depth of every logic cone in the design is increased relative to its unprotected counterpart. This elongation of logic paths directly extends the critical path of the design, thereby reducing the maximum achievable operating frequency and necessitating a lower clock frequency in order to satisfy timing closure requirements. Second, the replacement of each original gate with multiple secure gate instances results in a substantial increase in the total cell count and, by extension, the overall silicon area of the chip. These two effects, degraded timing performance and increased area overhead, represent the primary costs of applying Boolean Masking within the constraints of the current secure gate library and should be carefully considered when evaluating the practicality of this approach for resource-constrained or performance-critical applications.

TABLE VIII
INTERRUPT LATENCY MEASUREMENT RESULTS

Test Name	Min	Max	Avg
RISC-V Single Interrupt Test	6	43	15.2
RISC-V Multiple Interrupt Test	6	43	16.0
RISC-V Interrupt WFI Test	6	44	10.7
RISC-V Nested Interrupt Test	6	43	16.8
RISC-V Interrupt CSR Test	6	44	15.1

Note: Values represent the observed minimum, maximum, and average interrupt latency across the measured runs for each test.

VI. IMPLEMENTATION STATUS AND REMAINING FLOW

The design for the augmented ibex core with Boolean masking, noise generation, and a random clock signal has been completed and verified at the RTL level. Although proven satisfactory, the verification is an iterative process, and edge-cases are constantly being evaluated. The design is currently progressing toward physical implementation, with physical design stages still underway.

A. Synthesis and Netlist Generation

Synthesis has been completed using the Cadence Genus synthesis tool targeting the 45 nm process node standard cell library. Gate-level netlists for both the masked and unmasked cores have been generated, as well as the netlist that integrates them into the whole SoC. Multiple synthesis runs have been and will continue to be executed to reduce the area and timing overhead from the security extension as much as possible. Reports generated from the synthesis tool identify the critical path of the design and verify that the current design meets the frequency targets. As stated previously, the only remaining tasks are iterative synthesis to further optimize timing constraints, as we plan to cover for worst-case corners to ensure robust operation.

B. Remaining Path Toward Full Physical Implementation

Following synthesis and netlist generation, the design progresses into physical implementation using Cadence Innovus. At the time of writing, floorplanning is actively in progress, which involves defining the core area, assigning I/O pin locations, and establishing the power distribution network to maintain IR drop within the 5% specification. Current efforts are focused on achieving an optimized floorplan such that ideal density and IR drop are maintained throughout place-and-route. Any DRC violations that arise during the routing stage will be resolved using the tool's GUI, after which static timing analysis (STA) will be performed to verify that the design meets its target operating frequency. As such, this work represents a pre-silicon implementation spanning the full ASIC design flow, from RTL design and verification through synthesis and place-and-route, demonstrating both architectural feasibility and security integration. While the physical implementation remains ongoing, the completed stages establish a solid foundation from which the remaining steps can be executed. The resulting design artifacts provide a clear path forward for future efforts aimed at producing a fabrication-ready ASIC.

VII. Conclusion

As a part of this project, the team has successfully modified and synthesized an open-source Ibex Core such that it mitigates First-Order Side Channel Attacks, achieving an operating frequency of $\sim$ 21 MHz and an area of $\sim 190,000 \mu m^2$. This was accomplished through the careful implementation of both hiding and masking techniques integrated directly into the design at various levels of the RTL-to-GDS flow.

For the hiding techniques, ring oscillators were incorporated into the design to generate controlled electrical noise, which serves to obfuscate the correlation between power dissipation and the data being processed by the core. By introducing this noise into the power signature of the chip, it becomes significantly more difficult for an attacker to extract meaningful information through power analysis, as the signal-to-noise ratio of any captured power trace is substantially degraded. In addition to the hiding techniques, Boolean Masking was implemented within the core by systematically replacing all combinational gates with their secure, masked equivalents. This ensures that no single intermediate value computed within the core carries exploitable information about the secret data on its own, as every value is split into randomized shares that only reveal the intended result when combined. Special care was also taken during synthesis to ensure that EDA tool optimizations did not undo or simplify the masked logic, as described in earlier sections of this report. The effectiveness of these countermeasures was verified through Gate-Level Simulation (GLS) simulations and regression testing, confirming that the functional correctness of the core was maintained throughout the masking and synthesis process. For the physical implementation, pin placements and power planning have been completed, establishing the necessary groundwork to proceed with place-and-route on the design. With these milestones achieved, the project demonstrates a viable methodology for hardening an open-source RISC-V core against First-Order Side Channel Attacks within a standard RTL-to-GDS flow.

VIII. Acknowledgements

The authors would like to express their gratitude to Northrop Grumman for their invaluable insights and guidance throughout this project. The technical expertise and support provided by members of the Northrop Grumman team proved instrumental in enabling the team to be successful throughout the course of the project.

The authors would also like to extend their sincere appreciation to Dr. Mike Borowczak for his consistent encouragement and support throughout the duration of this project. His efforts in facilitating access to university resources, including the VLSI server and materials from prior iterations of Northrop Grumman's senior design projects, as well as his guidance in shaping the conceptual direction of this work, were essential to its completion.

References

[1] lowRISC, "Ibex: A small 32-bit risc-v cpu core previously known as zero-riscy," https://github.com/lowRISC/ibex, gitHub, Accessed: Sept. 4, 2025.

[2] A. Waterman, K. Asanović, and J. Hauser, "The risc-v instruction set manual, volume ii: Privileged architecture," RISC-V International, Document Version 20211203, 2021.

[3] H. Gross, S. Mangard, and T. Korak, "Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order," 2016, accessed: 2025-10-21. [Online]. Available: https://eprint.iacr.org/2016/486.pdf

[4] A. Biryukov, D. Dinu, Y. L. Corre, and A. Udovenko, "Optimal first-order boolean masking for embedded iot devices," 2018, accessed: 2025-10-20. [Online]. Available: https://orbilu.uni.lu/bitstream/10993/37740/1/Optimal_Masking.pdf

[5] Z. He, X. Deng, B. Yang, K. Dai, and X. Zou, "A sca-resistant processor architecture based on random delay insertion," in *2015 International Conference on Computing and Communications Technologies (ICCCT)*, 2015, pp. 278–281.

[6] Unknown, "Presentation paper no. 04 at the workshop on secure risc-v architecture design (secrisc-v'20)," in *Proceedings of the* Workshop on Secure RISC-V Architecture Design (SECRISC-V'20), 2020, accessed: Nov. 25, 2025. [Online]. Available: https://secriscv.org/2022/previous/2020/materials/presentations/presentation_paper_04.pdf

[7] K. Stangherlin and M. Sachdev, "Design and implementation of a secure risc-v microprocessor," 2022. [Online]. Available: https://arxiv.org/abs/2205.05095