Design and Implementation of a Secure RISC-V Microprocessor

Kleber Stangherlin and Manoj Sachdev, *Fellow, IEEE*

***Abstract*— Secret keys can be extracted from the power consumption or electromagnetic emanations of unprotected devices. Traditional countermeasures have a limited scope of protection and impose several restrictions on how sensitive data must be manipulated. We demonstrate a bit-serial RISC-V microprocessor implementation with no plain-text data. All values are protected using Boolean masking. Software can run with little to no countermeasures, reducing code size and performance overheads. Unlike previous literature, our methodology is fully automated and can be applied to designs of arbitrary size or complexity. We also provide details on other key components, such as clock randomizer, memory protection, and random number generator (RNG). The microprocessor was implemented in 65-nm CMOS technology. Its implementation was evaluated using NIST tests and side-channel attacks. Random numbers generated with our RNG pass on all NIST tests. The side-channel analysis on the baseline implementation extracted the advanced encryption system (AES) key using only 375 traces, while our secure microprocessor was able to withstand attacks using 20M traces.**

***Index Terms*— Boolean masking (BM), clock randomization, dynamic logic, secure microprocessor, side-channel attack.**

I. Introduction

INCREASED cybercrime and decentralized currencies are escalating the importance of protecting digital assets. Malicious individuals use a number of techniques to extract secrets from integrated circuits (ICs). Traditional hardware security countermeasures only target a few most vulnerable elements, such as system buses, memory, and cryptographic accelerators. We demonstrate a bit-serial RISC-V microprocessor implementation with no plain-text data. All values are protected using Boolean masking (BM). Our architecture sets no constraints on how sensitive data must be manipulated. Software implementations may run with little to no countermeasures, reducing code size and performance overheads.

Unlike previous literature, our methodology is fully integrated into the ASIC digital design flow. We carefully selected a set of countermeasures that require no changes to the input register transfer level (RTL) code or the application software. Our techniques can be applied to digital designs of any size or complexity. Original circuit functionality and latency are not affected.

Our design uses BM at the logic level. BM splits each value into two or more shares. Individual shares are (ideally) uncorrelated to the original value. Physical probing of all shares is required to uncover the original data. Accurate tampering of the system requires precise modifications in all shares of a value, which is considerably more difficult.

At the transistor level, we use differential domino logic (DDL) to implement our logic gates. DDL is a differential input/output dynamic logic style with precharge/evaluation phases. During precharge, both outputs are driven to the same value, but only one of them toggles in the evaluation phase. DDL is used to suppress glitches and reduce data-dependent power consumption.

To allow meaningful comparison, we implemented three different versions of a bit-serial RISC-V microprocessor using the same RTL code as input. A baseline implementation with no countermeasures, another using BM, and a third that combines BM and DDL. This work also provides detailed information on the implementation of key components, such as a memory protection unit, a clock edge randomizer, and a random number generator (RNG).

We fabricated a testchip in 65-nm CMOS. We report the area cost of all countermeasures used, as well as the power consumption of each microprocessor and the RNG. The quality of random numbers is evaluated using NIST tests, autocorrelation, and the Shannon entropy. Side-channel leakage assessment is performed using an extensive database of recorded power traces—a total of more than 40M traces. We use state-of-the-art analysis algorithms for correlation power analysis (CPA) with dynamic time warping (DTW) preprocessing. Our results evaluate each countermeasure individually such that we can determine its relative effectiveness.

Our main contributions include the demonstration of a microprocessor where all values are protected with BM and precharge logic. This work serves as proof of concept for a system where software implementations can run with fewer countermeasures. Our RTL design and implementation scripts are publicly available [42]. We describe scalable and automated solutions that require no changes to the architecture being protected. Relevant problems, such as the high-throughput requirements for the RNG, clock randomization, and the details of address/data protection at memory interfaces, are discussed. Finally, this article also performed a security assessment of different countermeasures using distinct implementations of the same initial design.

II. Related Works

Secret keys can be extracted from the power consumption or electromagnetic emanations of unprotected devices [27], [36]. BM was first proposed in [15] followed by [24] and [46]. Threshold implementation (TI) was introduced as a provably

Manuscript received 10 May 2022; revised 15 July 2022; accepted 13 August 2022. Date of publication 19 September 2022; date of current version 24 October 2022. *(Corresponding author: Kleber Stangherlin.)*

The authors are with the Electrical and Computer Engineering (ECE) Department, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: khstangh@uwaterloo.ca; msachdev@uwaterloo.ca).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2022.3203307.

Digital Object Identifier 10.1109/TVLSI.2022.3203307

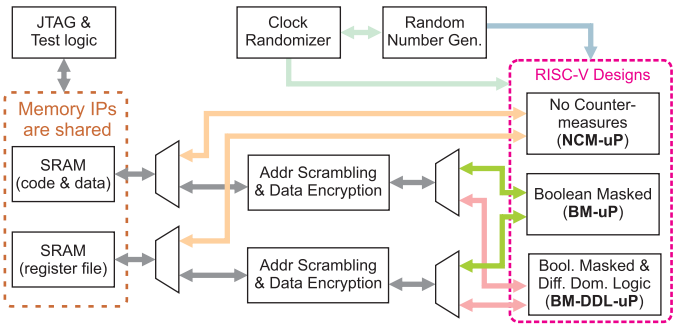


Fig. 1. Top-level block diagram of our testchip.

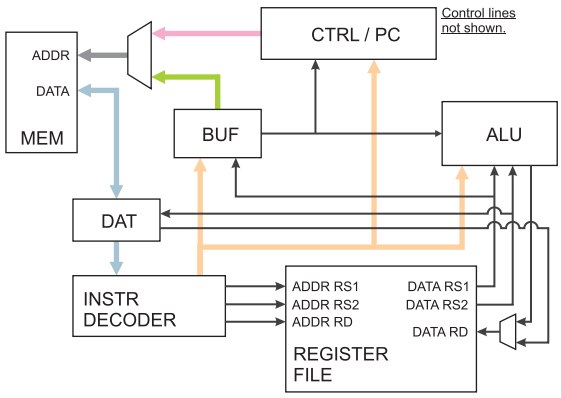


Fig. 2. Bit-serial RISC-V datapath. Control lines are not shown. Thin lines have a width of a single bit. Although not shown in this diagram, memory and RF are shared among the three microprocessors.

glitch-resistant masking technique [33]. Recent developments have been focused on reducing the area and randomness cost of implementing masked circuits [8], [17], [19], [20], [21], [22], [38]. Most of BM literature is centered on protecting nonlinear operations in cryptographic algorithms. Interest in larger scopes of protection has risen in recent years. A microprocessor arithmetic logic unit (ALU) is masked using TI [16]. Masking was applied to a binarized neural network [13] and RISC-V microprocessors [12], [18]. This work explores masking expressions that have lower costs and are easy to integrate into already existing designs [8].

Differential logic styles with the return to zero encoding make power consumption less data-dependent. For example, in [43], a sense amplifier-based logic (SABL) is used to reduce asymmetries of the dynamic logic. In [44], wave dynamic differential logic (WDDL) uses static CMOS gates to replicate the behavior of dynamic logic, but glitches still leak sensitive information. Other logic styles include [2], [7], [9], [32], [34], [35], and [41]. Our work differentiates in the applied methodology. Instead of "semicustom" techniques, our scripts are integrated with CAD tools and perform automatic replacement of relevant static CMOS cells by dynamic cells, without affecting place and route or static timing analysis.

Often, countermeasures are tailored to the hardware implementation of specific algorithms. For example, Kumar *et al.* [29] use randomized byte ordering, heterogeneous substitution boxes (SBOXes), and linear BM of mix columns to protect a advanced encryption system (AES) implementation. Other recent works have developed more generic protection mechanisms using fast, randomized, and voltage dithering in the voltage regulator [14], [25], [28], [40]. Comparing the effectiveness of countermeasures from different publications is not trivial. Changes in the acquisition system and the presence of combined countermeasures make it hard to draw any conclusions. Finally, it is possible that small drops in the supply voltage could create a misalignment in the power traces. The mentioned works lack a side-channel leakage assessment using alignment techniques, such as DTW.

High-throughput RNGs are essential for the effective masking of digital circuits. A large literature exists on harvesting random numbers from different entropy sources. Recent proposals include latch metastability [11], [31], [50], time to collapse in multimode ring oscillators (ROs) [49], or common mode analog comparators [3]. However, perhaps, the most well-understood entropy source is still phase jitter [5]. While cryptographic algorithms need high-quality random numbers with forward/backward secrecy [26], masking implementations may tolerate random numbers generated by lightweight RNG designs that focus on throughput, and that is where our RNG is positioned.

III. Hardware Architecture

A. Top-Level Design

Our design has three RISC-V microprocessors (see Fig. 1). Each microprocessor is designed to test a different set of security countermeasures. They share two SRAM memories for code, data, and register file (RF). The test logic selects which one of the three microprocessors will be operational, as well as the desired configuration for clock edge randomization and random numbers. The RISC-V implementation with no countermeasures (NCM-uP) has direct access to the SRAM memories, while others go through a memory protection unit. Netlist manipulation techniques described in this section are only applied to the BM-uP and BM-DDL-uP microprocessors. Other blocks are synthesized with the typical ASIC digital design flow using a commercial library of standard cells.

B. Bit-Serial RISCV Microprocessor

We used a bit-serial CPU, as shown in Fig. 2. It is based on a publicly available design.[1] This CPU design was chosen for its small area footprint. The internal datapath is one bit wide. The techniques described in this work are equally applicable to larger microprocessors. The main memory is a single port and uses 32-bit words to store both code and data. The RF is a dual port: one port for writes and another for reads with 2-bit words to avoid an additional read port. The ALU operates on a single bit of data per cycle. The fetch–execute–writeback process requires 36 cycles. Instructions need one or two phases to complete. Two-phase instructions, such as loads, stores, and branches, can use up to 70 cycles. Only three 32-bit registers exist. Register buffer (BUF) is a 32-bit register that holds data between phases, which can be addressed for branches, load/store, or data to be shifted. DAT is another 32-bit register, used for memory load and store operations,

[1] https://github.com/olofk/serv

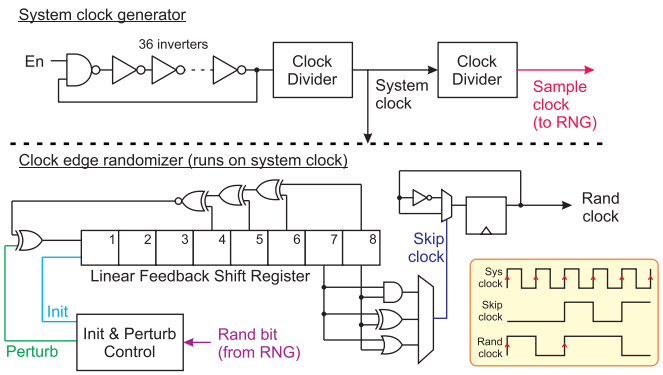


Fig. 3. Clock generation and randomization logic. Glitch suppression, test, and forbidden state prevention logic are not shown.

where data are shifted in from RS2 during store operations, and shifted out to RD during load operations. The last 32-bit register holds the program counter (PC). If the instruction is not a branch, the next address (PC + 4) is calculated one bit at a time, in parallel with ALU execution. All three 32-bit registers work similarly to a shift register during execution. For memory operations, parallel output/capture functionality is used. The instruction decoder (ID) parses instructions from DAT. It outputs immediate fields to BUF, controller/program counter (CTRL/PC), and ALU, one bit at a time, to calculate jump addresses and arithmetic operations. Control and status registers (CSRs), interrupts, multiplication, and divide instructions were not implemented. Using a 65-MHz clock, an AES encryption of 128 bits, without any software countermeasures, takes nearly 20 ms—SBOXes are not stored as lookup tables but computed during execution.

C. Clock Generation and Randomization

Power analysis attacks require a large number of power traces. Each trace records the power consumption, while the device is manipulating sensitive information with different input data. For example, AES encryptions with different plain texts but the same key. As discussed in Section IV-E, the effectiveness of power attacks is significantly higher if traces are aligned in time. Therefore, the insertion of random delays to purposely misalign power traces may be used as a countermeasure against power analysis attacks. To avoid changing software or existing hardware architectures, we insert random delays by manipulating the clock signal.

Our clock generation and randomization logic is shown in Fig. 3. A RO generates the clock using 36 inverters and a NAND gate. The layout of the RO was done manually. We used MOSFETs with the channel length of 4× the minimal value to achieve a nominal frequency of 226 MHz. Compensation for temperature and noise was not implemented. The hardware includes an adjustable clock divider from 2 to 256. Glitch suppression and test logic are not shown. To generate a random signal for clock edge randomization, we used an eight-degree linear feedback shift register (LFSR) in the XNOR form, polynom $x^8 + x^6 + x^5 + x^4 + 1$, cycle length $2^8 - 1$, and all ones excluded [1]. The output is taken from the two least significant bits; it indicates when an edge will be skipped.

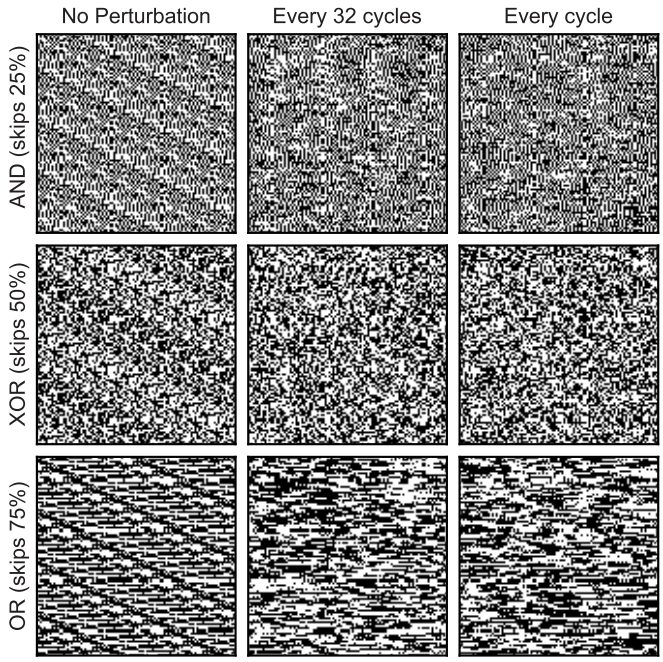


Fig. 4. Clock waveform visualization for nine different randomization settings. White/black pixels represent zero/one. Time passes from left to right and top to bottom. The white pixel on the left and the black pixel on the right represent rising edges.

A multiplexer was added to select how often clock edges are skipped. Valid options are 25%, 50%, or 75% for outputs from AND, XOR, and OR, respectively. For example, if the LFSR state is "10001101," both XOR and OR outputs would skip next clock edge if selected, while the AND output would introduce a clock edge.

Real applications require the hardware to complete operations under a time budget. We generate the clock randomization pattern using an 8-bit LFSR, which has a sequence length of 255. Therefore, within a 255-cycle window, the randomized clock will have a constant number of edges. This enables performance predictability with a resolution of 255 system clock cycles. Nevertheless, one may argue that a repeating clock randomization pattern introduces potential vulnerabilities. For this reason, we allow the 8-bit LFSR to be perturbed at a user-adjustable rate. The perturbation signal comes from the RNG; therefore, it has very large cycle length. If perturbed every clock cycle, we can assume that the randomized clock pattern will never repeat. However, under such a frequent perturbation rate, the LFSR is more likely to, sometimes, produce a run of 30 ones—causing the clock randomizer to skip 30 consecutive clock edges. To reduce the likelihood of such events, we suggest a moderate perturbation rate. For example, when perturbed every 32 cycles, the clock randomizer pattern will follow the 8-bit LFSR natural counting sequence for 31 cycles, until it is perturbed again.

Fig. 4 shows a grid of 3 × 3 subplots with a randomized clock waveform presented using image pixels. Rows have different clock skip rates, while columns have different perturbation rates. In each subplot, time is discretized as pixels. White/black pixels represent logic 0/1, respectively. Time passes from left to right and top to bottom (the clock

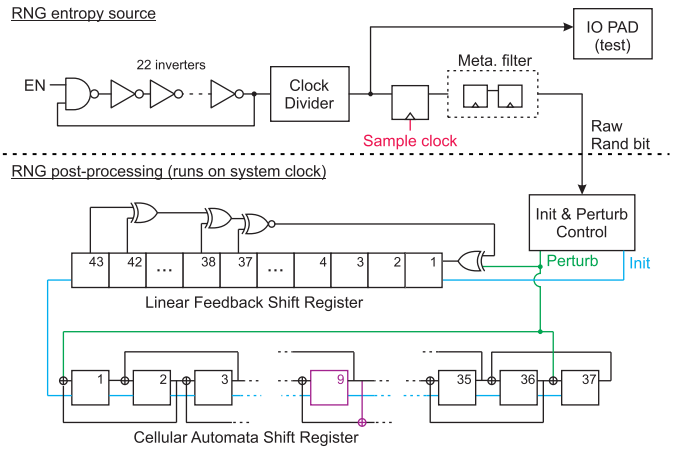


Fig. 5. RNG architecture. The sampling clock comes from the clock generation circuit.

waveform is folded in the image area). A transition from a white pixel (at the left) to a black pixel (at the right) denotes a rising edge. A pattern is clearly visible in the clock waveform when the LFSR is not perturbed. Perturbations every 32 cycles, and every cycle, remove noticeable patterns.

The clock skip rate affects the number of rising edges occurring in a period of time. Application engineers may assign different clock skip rates to software routines, depending on the sensitivity of data being manipulated and performance requirements. Also, care must be taken to avoid placing the LFSR into the forbidden state ("all ones," where updates no longer occur). The LFSR update cycle is skipped in case the next state is all ones. The initial state is generated by the RNG and loaded serially during power ON.

D. Random Number Generator

Random numbers are a critical component of secure ICs. Both clock randomization and BM require random numbers to work effectively. In particular, BM needs a large number of fresh (new) random numbers every clock cycle for remasking operations (see Section III-E). The entropy requirement for BM, however, is not as high as typical cryptographic uses, such as key generation. Therefore, our RNG focuses on high throughput to attend to the demand of our implemented countermeasures.

Our RNG design uses RO thermal noise as an entropy source. It accumulates thermal noise over a period of time, which produces phase jitter. The entropy of a sampled value is inversely proportional to the sampling rate. The RNG RO design, as shown in Fig. 5, uses the same techniques as the system clock RO but with a relatively prime number of stages to avoid frequency locking. The RNG RO has 22 inverters and a NAND gate with 366-MHz nominal frequency. The sampling clock is derived from the system clock. Sampling frequency depends on target entropy rate and technology noise characteristics. An IO PAD is connected to the RO output for noise characterization (see Section IV-C). A two-stage metastability filter running on the system clock was added to the raw random bit output.

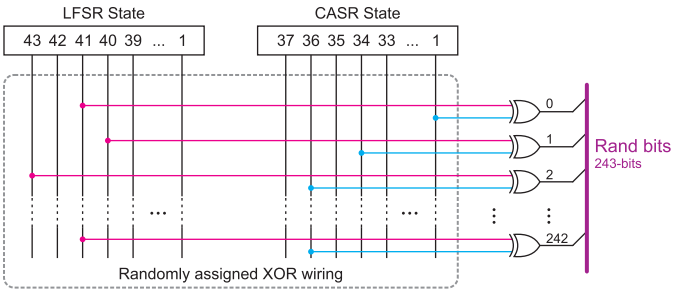


Fig. 6. Output XOR network for random numbers. XOR wiring is defined randomly at the design phase.

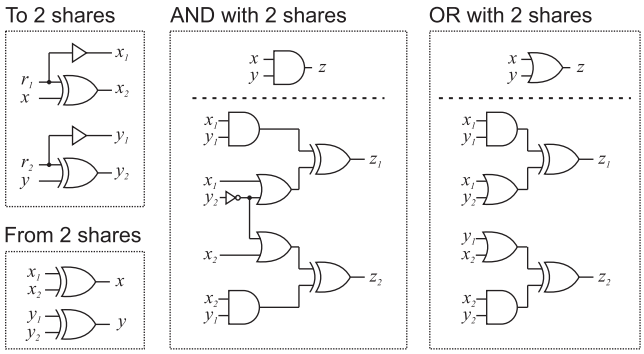


Fig. 7. BM expressions to perform nonlinear operations on shared values.

The throughput of raw random bits depends on sampling clock frequency, which is typically much lower than the system clock frequency. To increase throughput and remove possible entropy source biases, we added a postprocessing block. Our solution is based on [45]. We use an 43-degree LFSR in XNOR form, polynom $x^{43} + x^{42} + x^{38} + x^{37} + 1$, cycle length $2^{43} - 1$, and all ones excluded [1]. We also used a 1-D linear hybrid cellular automata shift register (CASR) of 37 cells, the cycle length of $2^{37}-1$, and all zeros excluded [10]. The hybrid CASR uses update rule 90 in all states, except in position 9, where rule 150 is used. Rule 90 and rule 150 are defined as $a_i(t+1) = a_{i-1}(t) \oplus a_{i+1}(t)$ and $a_i(t+1) = a_{i-1}(t) \oplus a_i(t) \oplus a_{i+1}(t)$, respectively [48]. The LFSR/CASR states are initialized with raw random numbers during power ON. Since the cycle lengths of LFSR and CASR are relatively prime, the combined cycle length is close to 2^{80}. Both LFSR and CASR update at every system clock cycle.

The LFSR feedback and the CASR first/last cells are XORed with the new raw random bit prior to the update. These perturbations to the natural counting sequence only occur when a new raw random bit is available, which depends on the sample clock frequency. Output is derived by XORing LFSR and CASR states. We use an XOR network, as shown in Fig. 6, which derives 243 outputs from LFSR/CASR state. Up to 1591 outputs are possible using two input XOR gates. More outputs require the XORing more than two state bits. The connection pairs are unique, randomly assigned, and defined at design time (they are static).

E. Boolean Masking

BM splits each value into two or more shares that are (ideally) uncorrelated to the original value. Computation

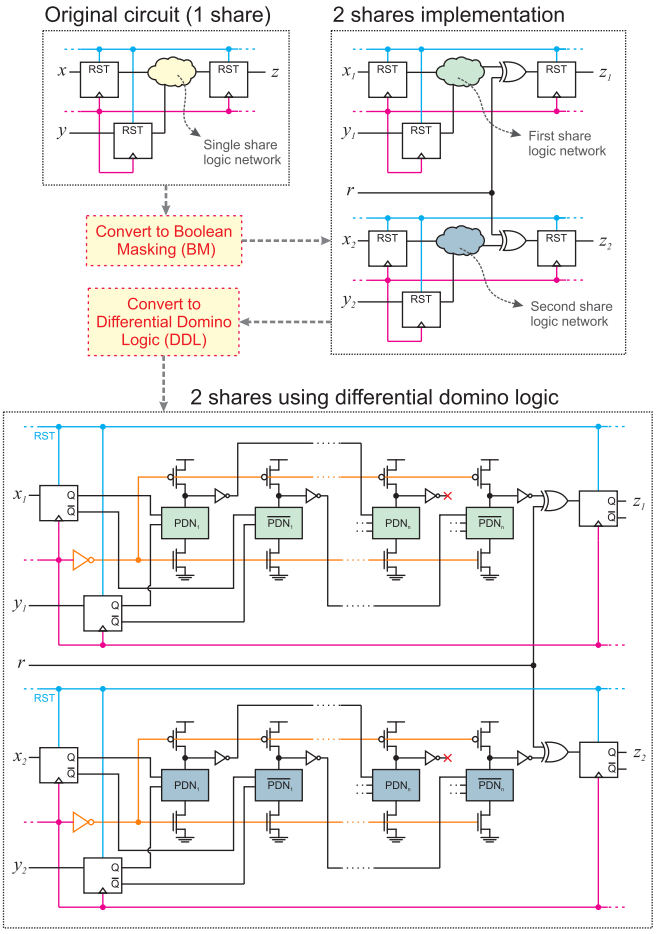


Fig. 8. Netlist conversion from original circuit to shared and then dynamic logic.

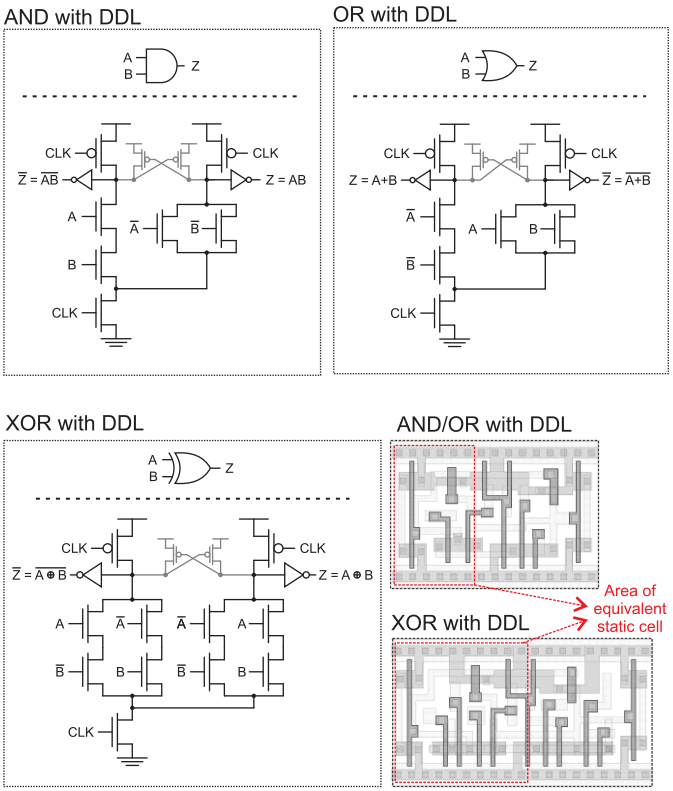


Fig. 9. Dynamic domino gates and associated layout.

using multiple shares requires careful logic manipulation, where each operation is replaced by its masked counterpart. We applied BM to the microprocessors designated as BM-uP and BM-DDL-uP. We used masking expressions from [8]. Although the masking expressions themselves are secure, further linear combinations can make them insecure [22]. Nevertheless, the used expressions have key characteristics that are very attractive for the construction of *fully masked* large-scale designs: 1) they are compact and 2) do not require fresh randomness at every operation. Fig. 7 shows the masked implementation of basic nonlinear logic gates (AND and OR) using masking expressions from [8]. Linear operations, such as XOR, shifts, and permutations, do not require any modification; they are applied to both shares equally. Fresh random numbers are required when converting from a single share to two shares. Conversions between two shares and a single share are only performed in the memory protection module, while data are still encrypted by session keys. Therefore, plain-text values are never exposed.

We developed a script that converts arbitrary circuit netlists into their masked counterparts. The script is integrated with CAD tools. Two restrictions were imposed: 1) the input netlist cannot use the clock and reset as data and 2) the reset must be asynchronous (see Fig. 8). The script splits every signal, except reset and clock, into two shares, replacing all basic operations with their masked counterparts. Circuit functionality and latency are not affected. All registers are duplicated and wired to accommodate the additional share of each state. An XOR gate is added before every register to refresh the masking with a new random number every cycle. No random number is ever reused. New top-level ports are automatically created to accommodate the additional share of all previously existing input/output ports. A new top-level input port bus for fresh random numbers is also created. It is driven by the RNG output network, as shown in Fig. 6, and each bit connects directly to a pair or remasking XOR gates.

The masked implementation of nonlinear operations requires signals to traverse between first/second logic networks. Such signals are not represented in Fig. 8. Also, our implementation does not mask the reset signal. If reset is synchronous, it can be masked together with logic.

F. Differential Domino Logic

Masking expressions for nonlinear operations (AND and OR gates) use signals from both shares (see Fig. 7). Glitches in that logic may momentarily expose protected values, reducing the effectiveness of BM [30]. Traditional solutions use registers to stop glitch propagation, which modifies circuit latency and requires significant design effort for existing designs. We reduce glitches by implementing our logic gates using DDL, a differential input/output dynamic logic style with precharge/evaluation phases [37]. During precharge, both outputs are driven to the same value, but only one of them toggles in the evaluation phase. Logic gate inputs are restricted to

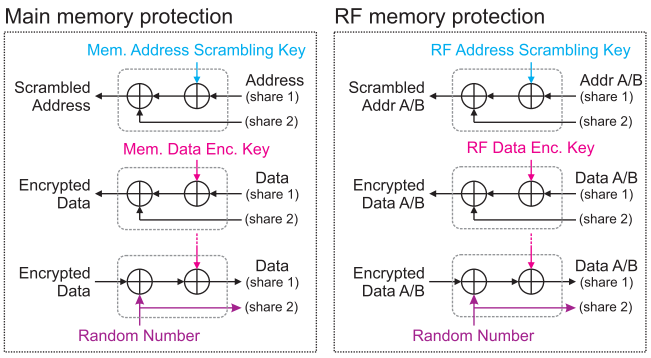


Fig. 10. Memory protection scheme.

a single 0 to 1 transition. Fig. 9 shows the transistor level schematic of AND, OR, and XOR using DDL with associated layout. Inverters are implemented by swapping the output wires. AND/OR gates have the same layout, which could be convenient for circuit obfuscation techniques, such as split foundry manufacturing [23].

Our netlist conversion script was adapted to use our DDL logic gates instead of the typical CMOS library cells. The output netlist has each wire split into two shares for BM. Moreover, each share is again split into two complementary signals for DDL. This technique was applied to BM-DDL-uP, as shown in Fig. 1. The layout of our DDL cells has the same height as the CMOS library cells so that we can conveniently mix DDL gates with static CMOS gates from the commercial library. Better area results are possible with taller DDL cells, but interoperability with commercial library cells would be extremely hard.

Our output netlist uses sequential elements from the commercial library. The remasking XOR gate preceding every sequential element also uses a cell from the commercial library—its output has no connection to dynamic cells, and glitches in that gate are unlikely to cause significant information leakage since one of the inputs is a random number. Inverted inputs to DDL gates are wired to register inverted output pins. Complementary signals were not granted their own sequential element—the number of registers does not quadruple with respect to the original design. Inverted outputs from the last DDL gate in a combinational path (prior to the remasking XOR) are left unconnected. This is a source of load imbalance that will leak information, but it avoids significant area costs.

We characterized multiple strengths of the DDL cells using a commercial tool. Our custom DDL library has two strengths of the AND/OR gate and three strengths of the XOR gate. Dynamic logic is not supported by the characterization tool, so we manually specified the relevant input/output timing arcs for delay characterization. We also wrote Verilog models for functional simulation, supporting delay annotation. The Liberty and Verilog files were used for timing analysis and design sign-off.

In order to maintain the methodology applicable to complex digital designs, we did not use routing constraints for differential signals. Therefore, wiring capacitance imbalances may be a source of information leakage.

TABLE I
AREA UTILIZATION IN 65-nm CMOS

Unit name	# of Instances	Area (μm^2)
Clock generation & randomization	203	922
Random number generation	674	2735
Microprocessor NCM-uP	731	3509
Microprocessor BM-uP	8232	22967
Microprocessor BM-DDL-uP	7633	44426
Main memory (code & data)	13	39505
Register file	19	10720
Memory protection and muxes	768	2653

Notes: XOR output network uses 242 XOR instances and account for 871-μm^2 of the area reported for the RNG.

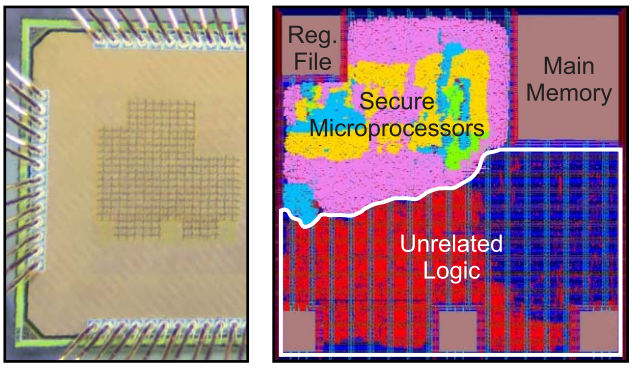


Fig. 11. Die photograph of the implemented chip in 65-nm CMOS technology and its layout. The layout highlights standard cells for NCM-uP (green), BM-up (yellow), and BM-DDL-uP (pink). Test logic, RNG, and clock generation/randomization are shown in light blue.

G. Memory Protection

A trivial solution for memory protection is to duplicate the memory size and store both shares of the Boolean masked data. To avoid a twofold increase in the SRAM size, we perform encryption using a session-based key, which is XORed with the data to alter its hamming weight. Moreover, we also scramble the address bus using another session-based key that is XORed with the address, changing the memory address layout. Session keys have different values at every session. The definition of a session is application-dependent, but, in our case, it corresponds to one execution of the target program in one of the microprocessors. Session keys remain stable for the duration of a session. Our session keys are provided via test logic. Fig. 10 shows the memory protection blocks. The order in which the XOR operations are performed is extremely important to avoid exposing plain-text values.

Single-port memory protection can be improved further. The encrypted data can be XORed with the address before it is written, making data encryption address-dependent. This requires a nonlinear expansion of the 10-bit address into a 32-bit word, which has extra area costs. It is also important to notice that our memory protection unit does not include firewall capabilities to detect unauthorized access. If present, such logic must be protected with BM and DDL.

IV. MEASUREMENT RESULTS

A. Chip Area Utilization

We fabricated a testchip in a 65-nm CMOS process, as shown in Fig. 11. Table I shows the number of instances

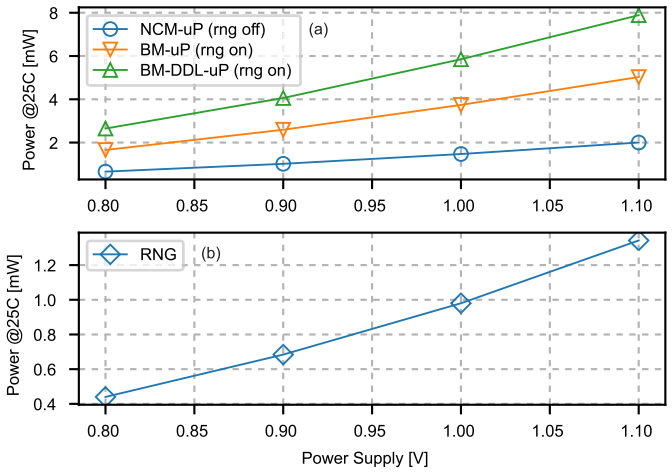


Fig. 12. Power consumption of (a) all RISC-V microprocessors and (b) RNG. Static components are not included.

TABLE II
NIST TESTS ON RNG OUTPUT

Test Name	ρ	Proportion
Frequency	0.554420	98/100
Block Frequency	0.042808	98/100
Cumulative Sums	0.964295	97/100
Runs	0.013569	99/100
Longest Run	0.289667	98/100
Rank	0.249284	98/100
FFT	0.474986	99/100
Non Overlapping Template	0.867692	96/100
Overlapping Template	0.115387	100/100
Universal	0.334538	97/100
Approximate Entropy	0.181557	99/100
Random Excursions	0.941144	67/69
Random Excursions Variant	0.619772	66/69
Serial	0.224821	98/100
Linear Complexity	0.017912	100/100

Notes: Cumulative Sums, Non Overlapping Template, Random Excursions, Random Excursions Variant, and Serial run multiple times in the NIST SP 800-22 Rev1a. Values in table refers to the worst pass proportion among all runs.

and area utilization of each block after design sign-off. The clock generation includes a 57-μm^2 RO and all clock edge randomization logic. The RNG includes a 37-μm^2 RO and all postprocessing logics (LFSR, CASR, and XOR network). The XOR network uses 242 XOR instances and accounts for 871 μm^2 of the reported area for RNG. The main memory uses a single-port 32-kbit SRAM (1024 $\times$ 32). RF uses a dual-port 1-kbit SRAM (512 $\times$ 2). The BM-uP implementation had an area increase of 6.5$\times$ compared to the baseline (NCM-uP), while the number of instances increased by 11.3$\times$. The BM-DDL-uP had an area increase of 1.9$\times$ compared to BM-uP, but the number of instances showed a small reduction due to the absence of inverters in differential logic styles.

B. Power Consumption

Dynamic power consumption was measured from 0.8 up to 1.1 V for the NCM-uP, BM-uP, BM-DDL-uP, and RNG. Nominal voltage is 1.0 V [see Fig. 12(a) and (b)]. We share the same power supply pin with other nonrelated digital circuits, which had their clock disabled during experiments. Moreover, static power was measured and excluded from all reported values. The RNG was not active during measurements of NCM-uP. The power reported for the RNG includes a system clock oscillator for sampling, RNG oscillator, LFSR/CASR postprocessing, XOR output network, and remasking XOR gates.

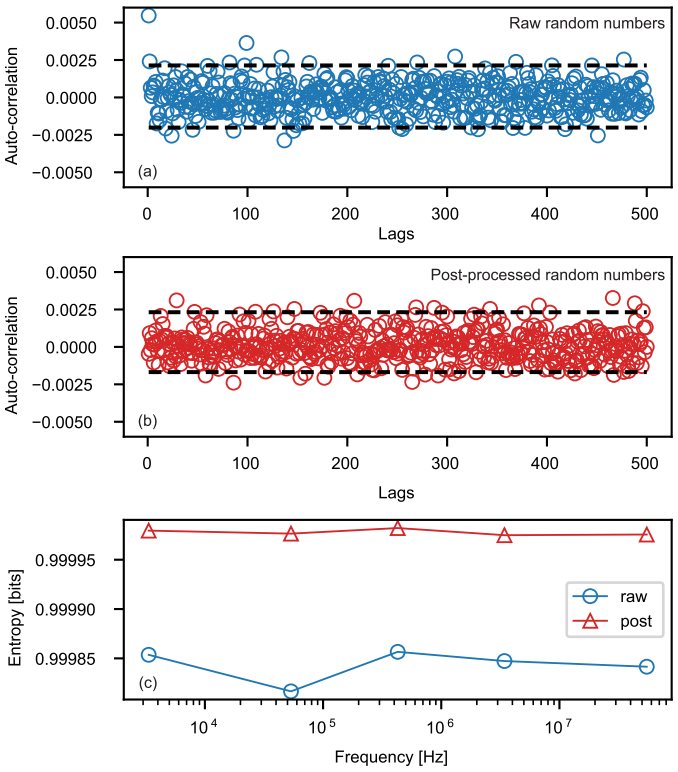


Fig. 13. Autocorrelation tests on (a) raw and (b) postprocessed random numbers. (c) Shannon entropy tests.

C. Quality of Random Numbers

The National Institute of Standards and Technology (NIST) published tests to assess the quality of random numbers [4]. Table II reports results for all tests. A total of 100 bitstreams with 1M bits each was used. All tests meet the suggested passing criterion of 80%. Fig. 13 plots the autocorrelation of 1M raw random bits [see Fig. 13(a)] and after postprocessing [see Fig. 13(b)] by LFSR/CASR logic. We used a sampling frequency of 3.4 MHz. Dashed lines represent the interval that contains 95% of the data. Fig. 13(c) plots the Shannon entropy for 1M bits at various sampling frequencies. Entropy was calculated using intervals of 5 bits.

Statistical modeling of phase jitter entropy sources was presented in [5]. The authors derived an expression for the minimum entropy rate. To use their expression, we measured the clock period standard deviation of our RNG oscillator over 120k cycles. Our testchip divides the clock by 256 and routes the signal to an output pin. The measured clock period mean was 716 ns, with 251 ps of standard deviation, which reflects the accumulated thermal noise in our RNG oscillator. Oscillators with high standard deviation values harvest more

TABLE III
COMPARISON OF RNG RESULTS

	This work	JSSC'16 [31]	JSSC'16 [3]	JSSC'22 [50]
Technology	65 nm	14 nm	65 nm	130 nm
Entropy	Jitter	Meta	Meta+Jitter	Meta
Bit rate (Gb/s)	12.8	0.225	3	0.002
Area (μm^2)	2735	1088	1609	5561
Power (mW)	0.992	1.5	5	-
Energy (fJ/bit)	0.078	6.67	1.67	-
Post-processing	LFSR/CASR	AES	None	VN8W

Notes: Power and energy reported for our work do not include the static component.

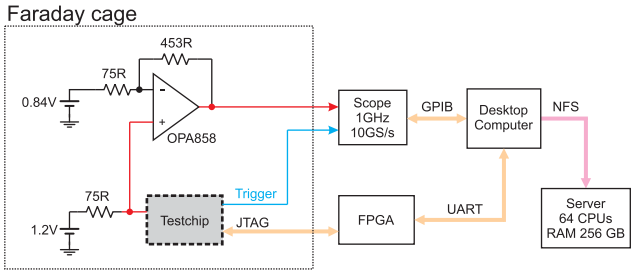


Fig. 14. Side-channel analysis acquisition setup.

entropy for the same sampling period. Similar jitter is also present in our system clock oscillator, and it was taken into account during timing analysis as clock uncertainty—it does not affect circuit functionality. The calculated minimum entropy for raw random numbers sampled at 3.4 MHz is 0.41 bits (per random bit). Using the model, 1.0 bit of entropy is achieved with the sampling frequency of 200 Hz or lower. Nevertheless, our application tolerates lower entropy for increased throughput. If a higher minimum entropy at increased throughput is required, additional ROs can be XORed before the sampling register. In that case, it is imperative to use ROs with a relatively prime number of stages to avoid frequency locking. Also notice that the model does not include phase jitter from our sampling clock, which could increase the calculated minimum entropy.

Our RNG produces 243 random bits in each clock cycle; therefore, its total throughput at 56 MHz is 12.8 Gb/s. Further details, such as area, power, and energy efficiency, are provided in Table III. This lightweight, throughput-focused RNG design meets our requirement for BM operations, but, arguably, other sensitive contexts, such as key generation, padding, and nonces, will require minimum entropy guarantees, slower sampling frequencies, and forward/backward secrecy in the postprocessing—which increases the hardware size significantly. We recommend the interested reader to see AIS 31 standard for details [26].

D. Experimental Setup for Side-Channel Analysis

Fig. 14 shows our experiment setup for side-channel analysis. We added a 75-Ω resistor in series with the testchip power supply. The signal is conditioned by a high-speed amplifier (OPA858, 5.5-GHz GBW) in a noninverting configuration with a gain of 7. A 0.84-V reference was connected to the gain resistor to avoid clamping of the output signal. The external chip power supply was set to 1.2 V instead of the nominal 1.0 V to account for the voltage drop in the series resistor. Our oscilloscope has an input bandwidth of 1 GHz. The acquisition system is orchestrated by a desktop computer that controls the oscilloscope using GPIB and communicates to the testchip using an FPGA. The FPGA writes the encrypted software using JTAG. All power traces use unique session keys for memory protection. The having sampling rate is 10 GS/s. We used an open-source Julia toolbox[2] to perform the CPA with and without DTW preprocessing. We attack only the first key byte during an AES encryption operation, targeting the SBOX output using a hamming weight leakage model. Our AES implementation uses multiplicative inverse over the Galois field to calculate the SBOX output (lookup tables are not used). The oscilloscope trigger is obtained from a testchip output set by the microprocessor software 36 cycles before the sensitive instruction. The number of samples logged in each power trace is adjusted depending on clock edge randomization settings to ensure that all relevant clock cycles are captured.

[2]https://github.com/Riscure/Jlsca

E. Correlation Power Analysis

CPA uses a set of power traces to extract secret information from a device. Traces record the device manipulating the secret value with different input data. For example, recorded power traces may include many AES encryptions using the same key but different plain texts. CPA attacks extract a few key bits in each iteration. The attacker chooses a key candidate and creates a power consumption hypothesis for each input data, using knowledge of the implemented algorithm, and a leakage model (typically Hamming weight). The correlation between the hypothesis vector and the recorded power traces will be the highest when the correct key is used.

Our side-channel analysis does not include test vector leakage assessment (TVLA) [6]. Such tests do not replace conventional key extraction attacks but provide a quick alternative to detect potential side-channel problems. Nevertheless, they require saving a long power trace that includes a series of encryptions, which would use an enormous amount of oscilloscope memory due to the low throughput of our hardware architecture. Future work shall use a faster architecture so that we can perform these tests.

Fig. 15 shows CPA results, plotting the correlation versus the number of traces used in the attack. Each plot has 256 lines, one for each key candidate. The correct key (red) is exposed when its correlation is the highest. In Fig. 15(a), the baseline implementation with no countermeasures (NCM-uP) had its key exposed with 375 power traces. Results with BM are shown in Fig. 15(b), where the key is exposed with 6200 power traces. If BM is combined with the dynamic logic implementation [see Fig. 15(c)], it takes 375k power traces, 1000× compared to the baseline, to expose the key.

Next, we investigated the effectiveness of the clock randomization on the baseline microprocessor (NCM-uP). We tested

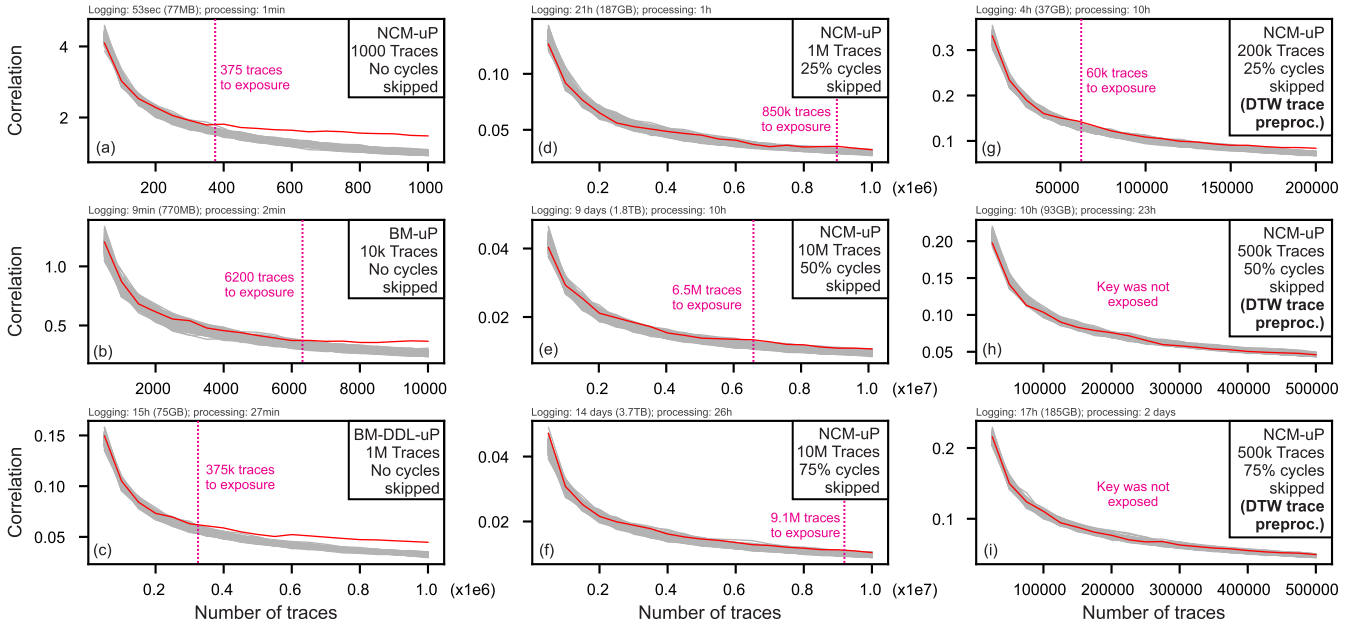


Fig. 15. CPA coefficient versus the number of traces for different sets of countermeasures. Results without clock edge randomization for (a) NCM-uP, (b) BM-uP, and (c) BM-DDL-uP; results for original netlist (NCM-uP) using clock edge randomization of (d) 25%, (e) 50%, and (f) 75%; and results using DTW preprocessing for trace alignment on NCM-uP with clock edge randomization of (g) 25%, (h) 50%, and (i) 75%.

three types of clock edge randomization. In Fig. 15(d), 25% of clock edges were skipped, and it took 850k power traces to expose the key. The required number of traces to expose the key increased dramatically to 6.2M and 9.1M, as we skipped 50% and 75% of the clock edges, as, respectively, shown in Fig. 15(e) and (f).

Preprocessing techniques, such as DTW, may be used to align power traces before a CPA attack [39]. The DTW algorithm originated from speech recognition systems to match spoken words to a database containing prerecorded words with different timing. We use a FastDTW variant that has complexity $O(\mathrm{Tk})$, where T is number of traces, and k is trace length [47]. In our experiments with DTW preprocessing, the radius parameter was set to 90. Larger radius enhances the representation accuracy of the original DTW algorithm but significantly increases computing time and memory usage.

Fig. 15(g)–(i) shows the CPA attack results with DTW preprocessing on the baseline microprocessor (NCM-uP), with clock randomization enabled. In Fig. 15(g), with 25% of clock cycles skipped, the key was exposed with only 60k traces, which is 14.2× fewer traces compared to CPA without DTW preprocessing, as shown in Fig. 15(d). However, it takes nearly 10× longer computation time. Similarly, Fig. 15(h) and (i), with 50% and 75% of clock cycles skipped, shows that the key was not exposed after a 500k traces CPA attack using DTW preprocessing, as opposed to the 6.5M and 9.1M traces required to expose the key without DTW.

We also assess the information leakage of all countermeasures combined. For that, we skip 50% of clock cycles at the BM-DDL-uP microprocessor, which implements both BM and DDL. Fig. 16 shows that CPA attacks using 20M traces without preprocessing [see Fig. 16(a)] and 2M traces with DTW [see Fig. 16(b)] were not enough to expose the key.

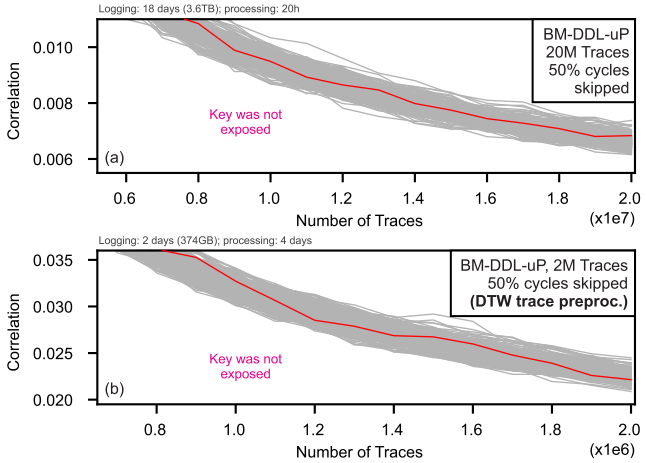


Fig. 16. CPA coefficient versus the number of traces for BM-DDL-uP with clock edge randomization of 50%. Results (a) without DTW trace preprocessing and (b) with DTW trace preprocessing.

F. Comparison With Prior Work

There are several implementations of Boolean masked microprocessors in the literature [12], [18], [34]. Table IV provides a comparison of our work with previous publications with respect to several key security features listed in the first column. It is clear from Table IV that our work covers a broad range of techniques and components necessary to implement a secure microprocessor.

Unlike other works, we used a CPU implementation with a datapath size of 1 bit. Such a decision was made due to area constraints in our testchip, but our methodology can be applied to any digital design of any size. In fact, our RTL and implementation scripts are publicly available to interested researchers [42]. However, comparing the effectiveness of countermeasures from different publications is not

TABLE IV
COMPARISON WITH OTHER SECURE MICROPROCESSORS

	This work	**CHES'07** [34]	**CARDIS'16** [18]	**DAC'19** [12]
Technology	65 nm	130 nm	FPGA	FPGA
Architecture	RISC-V	8051	RISC-V	RISC-V
Datapath	1 bit	8 bit	32 bit	32 bit
Clock rand	Yes	No	No	No
Entropy source	Jitter	No	No	No
PRNG	Yes	Not avail	Not avail	Not avail
Rand bits/cycle	243	1	–	–
Mem addr Scr	Yes	No	No	No
Mem data enc	Yes	No	No	Yes
Masking	[8]	MDPL [35]	DOM [21]	TI [33]*
Fully masked	Yes	Yes*	No	Yes*
Pre-charge logic	DDL	MDPL [35]	No	No
Methodology	Auto.	Not avail	Manual	Not avail
Traces to discl.	375	5 k	–	15 k
Max Traces	20 M	300 k	100 M	3 M
Attack types	CPA/DTW	DPA	TVLA	TVLA
Open-source	Yes [42]	No	No	No

Notes: (*) few details were provided in the publication.

trivial. Changes in the acquisition system and the presence of combined countermeasures make it hard to draw any definitive conclusions. It is also important to mention that the effectiveness of our countermeasures will likely increase when applied to larger designs. In addition to higher switching noise, the sensitive signals of larger designs will be relatively weaker compared to the total power consumption, requiring attackers to collect more power traces.

V. CONCLUSION

We demonstrated a bit-serial RISC-V microprocessor implementation with no plain-text data. Our design uses BM at the logic level and the dynamic domino logic at the transistor level. We selected a set of countermeasures that require no changes to the input RTL code. Unlike previous literature, our methodology is fully integrated with CAD tools and can be applied to digital designs of any size or complexity. We also provided details on other key components of secure ICs, such as clock randomizer, memory protection, and RNG. The random numbers generated with our RNG pass on all NIST tests. Side-channel analysis on the baseline implementation extracted the AES key using only 375 traces, while our secure microprocessor was able to withstand attacks using 20M traces.

REFERENCES

[1] P. Alfke, "Efficient shift registers, LFSR counters, and long pseudo random sequence generators," Xilinx, San Jose, CA, USA, Tech. Rep. XAPP 052, 1996.

[2] M. Avital, H. Dagan, O. Keren, and A. Fish, "Randomized multitopology logic against differential power analysis," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 4, pp. 702–711, Apr. 2015.

[3] S.-G. Bae, Y. Kim, Y. Park, and C. Kim, "3-Gb/s high-speed true random number generator using common-mode operating comparator and sampling uncertainty of D flip-flop," *IEEE J. Solid-State Circuits*, vol. 52, no. 2, pp. 605–610, Feb. 2017.

[4] L. E. Bassham *et al.*, "SP 800–22 rev. 1a. A statistical test suite for random and pseudorandom number generators for cryptographic applications," Nat. Inst. Standards Technol. (NIST), Gaithersburg, MD, USA, Tech. Rep. SP 800-22, 2010.

[5] M. Baudet, D. Lubicz, J. Micolod, and A. Tassiaux, "On the security of oscillator-based random number generators," *J. Cryptol.*, vol. 24, no. 2, pp. 398–425, 2011.

[6] G. Becker *et al.*, "Test vector leakage assessment (TVLA) methodology in practice," Cryptogr. Res., San Francisco, CA, USA, Tech. Rep., 2013.

[7] D. Bellizia, G. Scotti, and A. Trifiletti, "TEL logic style as a countermeasure against side-channel attacks: Secure cells library in 65nm CMOS and experimental results," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 11, pp. 3874–3884, Nov. 2018.

[8] A. Biryukov, D. Dinu, Y. Corre, and A. Udovenko, "Optimal first-order Boolean masking for embedded IoT devices," in *Proc. CARDIS*. Springer, 2017, pp. 22–41.

[9] M. Bucci, L. Giancane, R. Luzzi, and A. Trifiletti, "Three-phase dual-rail pre-charge logic," in *Proc. CHES*, 2006, pp. 232–241.

[10] K. Cattell and S. Zhang, "Minimal cost one-dimensional linear hybrid cellular automata of degree through 500," *J. Electron. Test.*, vol. 6, no. 2, pp. 255–258, 1995.

[11] L. T. Clark, S. B. Medapuram, and D. K. Kadiyala, "SRAM circuits for true random number generation using intrinsic bit instability," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 10, pp. 2027–2037, Oct. 2018.

[12] E. D. Mulder, S. Gummalla, and M. Hutter, "Protecting RISC-V against side-channel attacks," in *Proc. DAC*, Jun. 2019, pp. 1–4.

[13] A. Dubey, R. Cammarota, and A. Aysu, "BoMaNet: Boolean masking of an entire neural network," in *Proc. Int. Conf. Comput. Aided Design*, Nov. 2020, pp. 1–9.

[14] A. Ghosh, D. Das, J. Danial, V. De, S. Ghosh, and S. Sen, "Syn-STELLAR: An EM/power SCA-resilient AES-256 with synthesis-friendly signature attenuation," *IEEE J. Solid-State Circuits*, vol. 57, no. 1, pp. 167–181, Jan. 2022.

[15] L. Goubin and J. Patarin, "DES and differential power analysis the 'duplication' method," in *Proc. CHES*. Cham, Switzerland: Springer, 1999, pp. 158–172.

[16] H. Gross, "Sharing is caring-on the protection of arithmetic logic units against passive physical attacks," in *Proc. Int. Ws. Radio Freq. Identificat., Secur. Privacy Issues*. Cham, Switzerland: Springer, 2015, pp. 68–84.

[17] H. Groß, R. Iusupov, and R. Bloem, "Generic low-latency masking in hardware," in *Proc. TCHES*, 2018, pp. 1–21.

[18] H. Gross, M. Jelinek, S. Mangard, T. Unterluggauer, and M. Werner, "Concealing secrets in embedded processors designs," in *Proc. CARDIS*. Cham, Switzerland: Springer, 2016, pp. 89–104.

[19] H. Groß and S. Mangard, "Reconciling $d + 1$ masking in hardware and software," in *Proc. CHES*. Cham, Switzerland: Springer, 2017, pp. 115–136.

[20] H. Groß and S. Mangard, "A unified masking approach," *J. Cryptograph. Eng.*, vol. 8, no. 2, pp. 109–124, 2018.

[21] H. Groß, S. Mangard, and T. Korak, "Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order," Cryptol. ePrint Arch., Paper 2016/486, 2016. Accessed: Sep. 1, 2022. [Online]. Available: https://eprint.iacr.org/2016/486

[22] H. Gross, K. Stoffelen, L. D. Meyer, M. Krenn, and S. Mangard, "First-order masking with only two random bits," in *Proc. ACM Workshop Theory Implement. Secur. Workshop*, 2019, pp. 10–23.

[23] F. Imeson, A. Emtenan, S. Garg, and M. Tripunitara, "Securing computer hardware using 3d integrated circuit (IC) technology and split manufacturing for obfuscation," in *Proc. USENIX Secur. Symp.*, 2013, pp. 495–510.

[24] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks," in *Proc. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 2003, pp. 463–481.

[25] R. Jevtic, M. Ylitolva, C. Calonge, M. Ojanen, T. Santti, and L. Koskinen, "EM side-channel countermeasure for switched-capacitor DC–DC converters based on amplitude modulation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 6, pp. 1061–1072, Jun. 2021.

[26] W. Killmann and W. Schindler, "A proposal for: Functionality classes for random number generators, version 2.0," Bundesamt für Sicherheit in der Informationstechnik (BSI), Germany, Tech. Rep. AIS 20/AIS 31, 2011.

[27] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 1999, pp. 388–397.

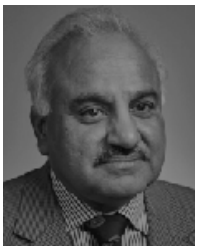
[28] R. Kumar *et al.*, "A time-/frequency-domain side-channel attack resistant AES-128 and RSA-4K crypto-processor in 14-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 56, no. 4, pp. 1141–1151, Jan. 2021.

[29] R. Kumar *et al.*, "A 4900-μm^2 839-Mb/s side-channel attack-resistant AES-128 in 14-nm CMOS with heterogeneous Sboxes, linear masked mixcolumns, and dual-rail key addition," *IEEE J. Solid-State Circuits*, vol. 55, no. 4, pp. 945–955, Jan. 2020.

[30] S. Mangard, T. Popp, and B. Gammel, "Side-channel leakage of masked CMOS gates," in *Proc. Cryptographers Track at RSA Conf.* Cham, Switzerland: Springer, 2005, pp. 351–365.
[31] S. K. Mathew *et al.*, "μRNG: A 300–950 mV, 323 Gbps/W all-digital full-entropy true random number generator in 14 nm FinFET CMOS," *IEEE J. Solid-State Circuits*, vol. 51, no. 7, pp. 1695–1704, Jul. 2016.
[32] M. A. Morrison, N. Ranganathan, and J. Ligatti, "Design of adiabatic dynamic differential logic for DPA-resistant secure integrated circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 8, pp. 1381–1389, Aug. 2015.
[33] S. Nikova, C. Rechberger, and V. Rijmen, "Threshold implementations against side-channel attacks and glitches," in *Proc. Int. Conf. Inf. Commun. Secur.* Cham, Switzerland: Springer, 2006, pp. 529–545.
[34] T. Popp, M. Kirschbaum, T. Zefferer, and S. Mangard, "Evaluation of the masked logic style MDPL on a prototype chip," in *Proc. CHES*, 2007, pp. 81–94.
[35] T. Popp and S. Mangard, "Masked dual-rail pre-charge logic: DPA-resistance without routing constraints," in *Proc. CHES*, 2005, pp. 172–186.
[36] J.-J. Quisquater and D. Samyde, "Electromagnetic analysis (EMA): Measures and counter-measures for smart cards," in *Proc. CARDIS*. Cham, Switzerland: Springer, 2001, pp. 200–210.
[37] J. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits*, vol. 2. Englewood Cliffs, NJ, USA: Prentice-Hall, 2002.
[38] O. Reparaz, B. Bilgin, S. Nikova, B. Gierlichs, and I. Verbauwhede, "Consolidating masking schemes," in *Proc. Annu. Cryptol. Conf.* Cham, Switzerland: Springer, 2015, pp. 764–783.
[39] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intell. Data Anal.*, vol. 11, no. 5, pp. 561–580, 2007.
[40] A. Singh, M. Kar, S. K. Mathew, A. Rajan, V. De, and S. Mukhopadhyay, "Improved power/EM side-channel attack resistance of 128-bit AES engines with random fast voltage dithering," *IEEE J. Solid-State Circuits*, vol. 54, no. 2, pp. 569–583, Feb. 2019.
[41] D. Sokolov, J. Murphy, A. Bystrov, and A. Yakovlev, "Improving the security of dual-rail circuits," in *Proc. CHES*, 2004, pp. 282–297.
[42] K. Stangherlin and M. Sachdev. *Design and Implementation of a Secure RISC-V Microprocessor (Code)*. Accessed: Sep. 1, 2022. [Online]. Available: https://github.com/cdrlabs-waterloo/2022-07-15
[43] K. Tiri, M. Akmal, and I. Verbauwhede, "A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards," in *Proc. E-SSCC*, Sep. 2002, pp. 403–406.
[44] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation," in *Proc. Design, Automat. Test Eur. Conf. Exhib.*, vol. 1, Feb. 2004, pp. 246–251.
[45] T. Tkacik, "A hardware random number generator," in *Proc. CHES*. Cham, Switzerland: Springer, 2002, pp. 450–453.
[46] E. Trichina, "Combinational logic design for AES subbyte transformation on masked data," IACR ePrint Arch., 2003. Accessed: Sep. 1, 2022. [Online]. Available: http://eprint.iacr.org/2003/236
[47] J. V. Woudenberg, M. Witteman, and B. Bakker, "Improving differential power analysis by elastic alignment," in *Proc. Cryptographers Track at RSA Conf.* Cham, Switzerland: Springer, 2011, pp. 104–119.
[48] S. Wolfram, "Random sequence generation by cellular automata," *Adv. Appl. Math.*, vol. 7, no. 2, pp. 123–169, 1986.
[49] K. Yang, D. Blaauw, and D. Sylvester, "An all-digital edge racing true random number generator robust against PVT variations," *IEEE J. Solid-State Circuits*, vol. 51, no. 4, pp. 1022–1031, Apr. 2016.
[50] R. Zhang, X. Wang, K. Liu, and H. Shinohara, "A 0.186-pJ per bit latch-based true random number generator featuring mismatch compensation and random noise enhancement," *IEEE J. Solid-State Circuits*, vol. 57, no. 8, pp. 2498–2508, Aug. 2022.

Kleber Stangherlin received the B.Sc. degree in electrical engineering from the Pontifical Catholic University of Rio Grande do Sul (PUCRS), Porto Alegre, Brazil, in 2010, and the M.Sc. degree in microelectronics from the Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, in 2013. He is currently working toward the Ph.D. degree at the University of Waterloo, Waterloo, ON, Canada.

He currently conducts research in hardware security at the University of Waterloo. He has more than six years of industry experience in designing security-focused integrated circuits. He had key contributions to the cryptographic cores and countermeasures used in the first EAL 4+ certified chip designed in the southern hemisphere.



Manoj Sachdev (Fellow, IEEE) is currently a Professor and an Interim Department Chair with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. He has contributed to over 180 conference and journal publications, and written five books. He holds more than 30 granted U.S. patents.

Dr. Sachdev is also a member of the Program Committee of IEEE Design and Test in Europe Conference. He is a fellow of the Engineering Institute of Canada. Along with his students and colleagues, he received several international research awards. He serves on the Editorial Board of the *Journal of Electronic Testing: Theory and Applications*.