An EDA-Friendly Protection Scheme against Side-Channel Attacks

Ali Galip Bayrak*, Nikola Velickovic*, Francesco Regazzoni†, David Novo*, Philip Brisk‡ and Paolo Ienne*

* School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
{aligalip.bayrak, nikola.velickovic,
david.novobruna, paolo.ienne}@epfl.ch

† ALaRI - University of Lugano
CH-6900 Lugano, Switzerland
regazzoni@alari.ch

‡ University of California, Riverside
339 Engineering II, CA 92521, USA
philip@cs.ucr.edu

***Abstract*—This paper introduces a generic and automated methodology to protect hardware designs from side-channel attacks in a manner that is fully compatible with commercial standard cell design flows. The paper describes a tool that artificially adds jitter to the clocks of the sequential elements of a cryptographic unit, which increases the non-determinism of signal timing, thereby making the physical device more difficult to attack. Timing constraints are then specified to commercial EDA tools, which restore the circuit functionality and efficiency while preserving the introduced randomness. The protection scheme is applied to an AES-128 hardware implementation that is synthesized using both ASIC and FPGA design flows.**

I. Introduction

Security is an important design objective in modern technology. A variety of products, ranging from small embedded devices to mobile and desktop processors now integrate embedded cryptographic cores. An important weakness of these devices, which this paper addresses, is fragility against side-channel attacks, which focus on the physical implementation of the cryptographic core, rather than attacking the cryptographic algorithm itself. Side-channel attacks may exploit physical information leaked by the device, including power consumption [13], timing [12], EM radiation [7] or acoustics [21], allowing the attacker to recover the secret information stored on the device. Fig. 1, for example, conceptually illustrates a power-based side-channel attack [13] based on correlating real-time power measurements with the processed data and the secret key.

Both software and hardware countermeasures for side-channel attacks have been proposed in recent years. Software countermeasures [1], [6], [22] are limited by the fragility of the underlying hardware, and cannot offer as much protection as hardware countermeasures; however, hardware countermeasures can be costly in terms of area, energy, and performance [23], [24], [25], and may require non-traditional EDA toolchains and/or full-custom implementations for proper realization [23], [25].

To address these concerns, this paper introduces a new hardware countermeasure for protection against side-channel attacks, which incurs modest area and energy overhead, maintains full compatibility with existing commercial standard-cell design flows, is not tied to a specific CMOS technology node, is cryptographic algorithm agnostic, and can be introduced automatically, thereby simplifying the task of the hardware designer. The countermeasure randomizes the timing of circuit signals, which, in turn, randomizes the side-channel information that is leaked, such as power consumption, as shown in Fig. 2.

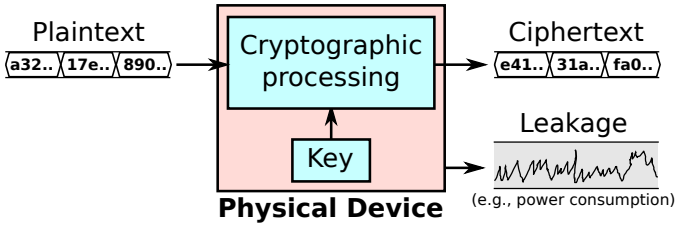


Fig. 1. Cryptographic algorithms are designed to be robust against mathematical attacks that exploit relations between the *plaintext* and *ciphertext* to recover the secret *key*. A side-channel attack, in contrast, attempts to exploit relations between physical information, such as power consumption, that the device inadvertently *leaks* during operation.

The countermeasure uses randomly-generated jittered clocks to increase the *uncertainty* of the signal arrival times at sequential elements in the circuit. The hardware generates M such random clocks in two steps:

1) It generates N phase-shifted clocks (of the same frequency) by inserting a δ time delay between two consecutive ones; and
2) It uses M clock multiplexers, each of which selects a different shifted clock per cycle.

This uncertainty is translated to commercial EDA tools as a new set of constraints. The tools can resolve problems related to the clock uncertainty, which has been introduced artificially, such as setup/hold time violations and glitches.

This method increases *non-determinism* of a circuit operation, which is a general approach taken by side-channel countermeasures. For example, power-based side-channel attacks try to establish a relationship between instantaneous

978-3-9815370-0-0/DATE13/©2013 EDAA

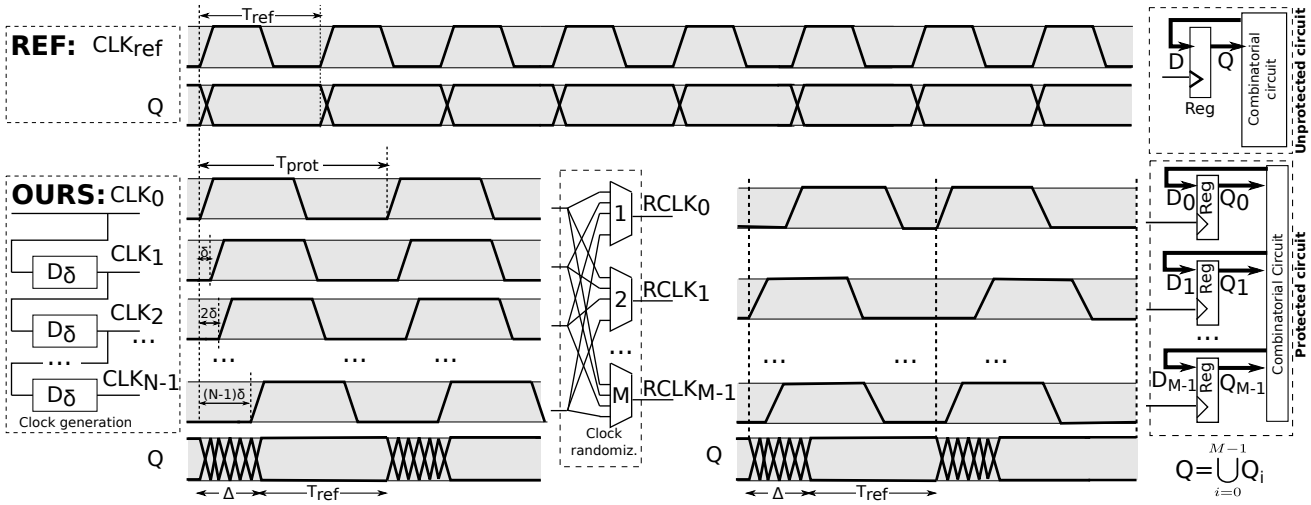


Fig. 2. Illustration of the countermeasure introduced in this paper. Sequential circuit elements are driven by clocks with randomly generated delays, which randomizes the timing of the signals. The tool generates N different clock signals ($CLK_{[0..(N-1)]}$) from a reference clock, with a delay of δ introduced between pair of clock signals, yielding delays as large as $\Delta = (N-1)\delta$. To compensate, the clock period is increased from τ_{ref} to $\tau_{prot} = \tau_{ref} + \Delta$. Next, M random clocks, $RCLK_{[0..(M-1)]}$ are generated using clock multiplexers, each of which selects one of the shifted clocks and updates its selection every clock cycle. These M random clocks are each sent to M sequential element groups, each of which contains at least one flip-flop, which varies the signal arrival time. Sections II and III describe methods to avoid violations and glitches caused by this randomization.

power consumption and the inputs (i.e., plaintext); the non-determinism introduced by our method increases the difficulty of uncovering such a relationship, as the instantaneous power consumption becomes randomized.

This paper makes the following contributions:

1) The protection scheme can be applied to any hardware implementation using any *standard-cell design flow*. Protection is applied at the RTL level, and does not require lower-level considerations, such as non-traditional standard cell libraries.
2) The design flow is fully automated, and can therefore be used by hardware designers who are *not* security experts.
3) The amount of protection depends on user-specified parameters N, M, and δ; hardware designers can trade-off security against other metrics such as performance and area, according to the design needs, by varying the parameter values accordingly. Past countermeasures have large constant overheads, rendering them infeasible under resource constraints.

II. Clock Randomization

Randomized clock generation is a two-step process. The first step is to generate N *phase-shifted clocks*, as shown in Fig. 2. The second step introduces M *clock multiplexers* to generate M *random clocks* which control the flip-flops in the circuit.

A. Shifted clock generation

Let CLK_0 be the regular (nonshifted) clock in the circuit. Given CLK_i, a shifted clock CLK_{i+1} is generated by introducing a delay element, which adds a delay of δ to CLK_i, $0 \leq i < N-1$; the delay elements are comprised of buffers and inverters. Commercial synthesis tools such as *Synopsys Design Compiler* can easily generate such delay units by specifying path delay constraints, such as '`set_min_path` δ `-from delay_in[i] -to delay_out[i]`", where `delay_in[i]` and `delay_out[i]` are CLK_i and CLK_{i+1}, respectively. After inserting the delay elements that meet the constraints, we then set the "`dont_touch`" attribute to this circuit to prevent the synthesis tools from optimizing away the delay elements later, as they are computationally redundant.

Our scheme does *not* require all N delay elements to have the same constant delay δ in order for the countermeasure to work. This makes the design more viable than other countermeasures that are based on strict timing requirements (e.g., those using complementary gates) that are difficult to satisfy in practice.

B. Random clock generation

The *shifted clocks* described in the preceding section are used to generate the *random clocks* described here. A separate clock multiplexing unit generates a random clock using the shifted clocks, $CLK_{0..(N-1)}$, and a $log_2 N$ bit random value, RND. Section II-C provides details on random number generation.

To prevent glitches at the multiplexer output, the select signal, SEL, is updated once per cycle during the *safe clock switching zone*, as shown in Fig. 4. During this period, all of the clock signals are low, and multiplexing among all zeroes guarantees safe and correct switching between clocks. The random input value, RND, is stored in a register clocked by the inverse of the final shifted clock, $\overline{CLK_{N-1}}$, and provides the output of the register as the select signal.

For this scheme to work, two conditions must be satisfied:

1) The clock period must be long enough to provide a safe

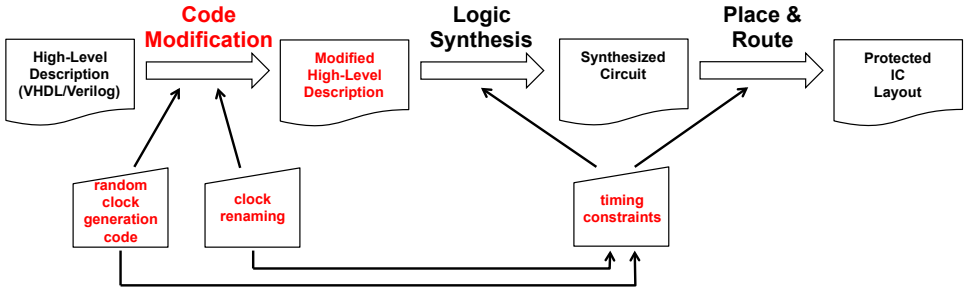


Fig. 3. Automatic design flow that introduces a protected hardware implementation. A *random clock generator* is introduced to the RTL circuit (Section II), which replace the original clock signals. The jitter information (Δ) is converted to a set of *timing constraints*, which are handled by a commercial EDA toolflow, which generates the *protected IC layout*. Logic simulation for verification proceeds as normal and is not shown here.

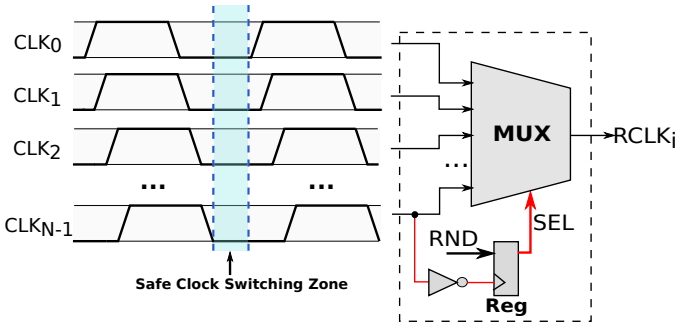


Fig. 4. The *shifted clocks*, $CLK_{0..(N-1)}$, and a $log_2 N$-bit random number, RND, generate a random clock, $RCLK_i$. There are M of these components, each having individual RND but sharing the same $CLK_{0..(N-1)}$. To suppress glitches the multiplexer select signal, SEL, is updated once per cycle during the *safe clock switching zone* when all clocks are low.

switching zone, i.e., the output of $RCLK_i$ should be ready before the rising edge of CLK_0.

2) The select signal should be updated after the last clock, CLK_{N-1}, falls; in other words, the path delay from the last shifted clock to the select input of the multiplexer, shown in red in Fig. 4, should be longer than the delay of the wires that drive the multiplexer's inputs.

After generating the random clocks, the set of flip-flops of the original circuit are partitioned into M mutually exclusive and collectively exhaustive sets; a separate clock is now driven to each set. The sets were chosen randomly in our experiments, and each set was chosen to have an equal number of flip-flops.

C. Random number generation

Random numbers can be read as pre-stored constant values, or generated at runtime using *Pseudo-* or *True-Random Number Generators* (PRNGs and TRNGs respectively). PRNGs are easy to implement at a low cost in terms of area and performance, but exhibit deterministic behavior that an attacker could exploit. TRNGs, in contrast, are based on non-deterministic physical phenomena. For the experiments, we implemented a TRNG based on jitter [11]: a system clock samples a high-frequency clock generated by ring oscillators. The designer must select appropriate random generator in this manner; a thorough analysis of random number generation in hardware is beyond the scope of this paper.

III. Design Flow

Fig. 3 shows the fully automatic design flow that introduces our hardware protection mechanism into an RTL description of a circuit. Given parameter values M, N, and δ, our script generates an HDL description of the random clock generation circuit (Section II) and integrates it into the hardware design. The script then randomly partitions the flip-flops and registers in the design, and updates the RTL so that each partition is driven by a different random clock (Section II-B).

To prevent timing violations, new timing constraints, which depend on the random clocks and the total jitter, $\Delta = (N-1)\delta$, are generated and propagated to the EDA tools. We model random clocks, $RCLK_{0..(M-1)}$, as separate clocks of the circuit with the same frequency and offset, each having an uncertainty of Δ; in *Synopsys Design Compiler*, we would use the `create_clock` and `set_clock_uncertainty` commands. This approach enables commercial EDA tools to produce an optimized circuit layout that respects our timing constraints while ensuring correct functionality of the random clocks. The toolflow automatically generates a protected implementation of the input circuit, which was originally unprotected.

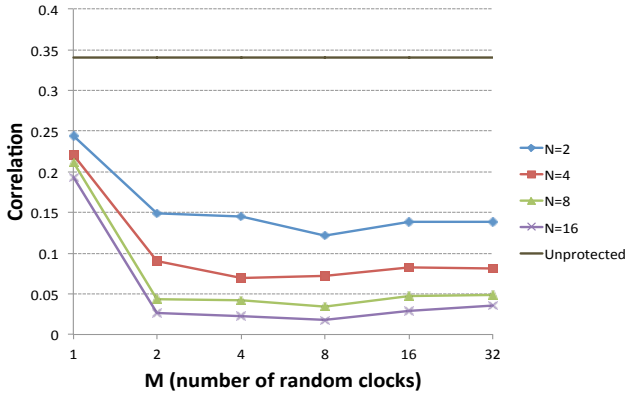
The user must appropriately select values for parameters N, M, and δ. In principle, this is a design space exploration problem, wherein the user can trade-off the amount of protection against area overhead, performance, and energy consumption. We provide a detailed example in the following section.

IV. Experimental Results

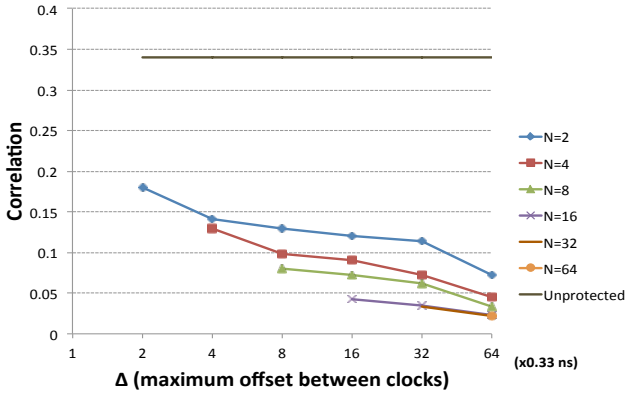
This section applies the countermeasure described previously to the AES-128 algorithm targeting both an FPGA and an ASIC design flow. Our results show how area, performance, and security scale as an empirical function of parameters M, N, and δ.

A. FPGA Experiments

1) Experimental Setup: We used the *Side-channel Attack Standard Evaluation BOard (SASEBO) G-II* board [20] connected to an oscilloscope with a 4 GHz sampling frequency and 600 MHz bandwidth, using a passive probe, to measure power consumption. The SASEBO board included

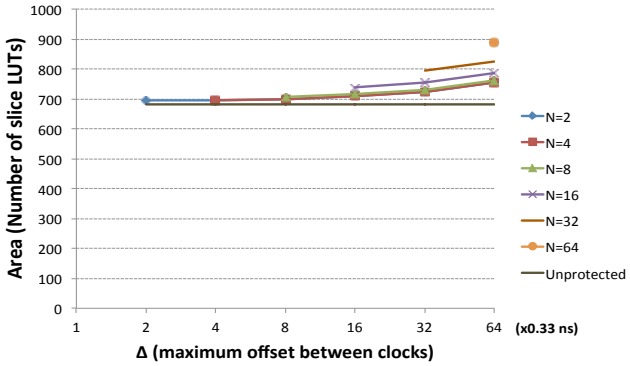


(a) Correlation as a function of M and N.

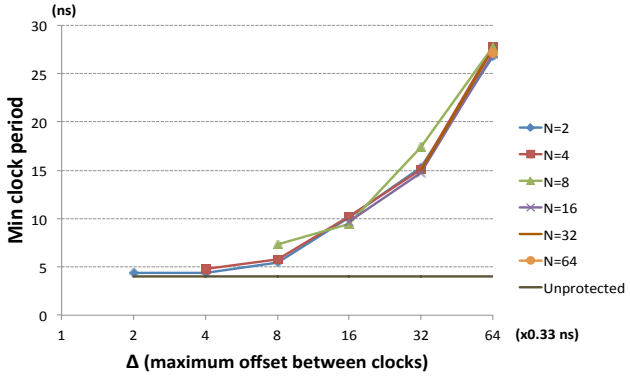


(b) Correlation as a function of Δ and N.

Fig. 5. Security experiments for AES-128 hardware implementation on a Xilinx Virtex-5 FPGA. Correlation represents the result of a correlation-based power attack; lower correlation values correspond to stronger security. Scaling factor on the x-axis of Fig. b (i.e., 0.33 ns) corresponds to the minimum delay of a single buffer.



(a) Area as a function of Δ and N.



(b) Minimum clock period as a function of Δ and N.

Fig. 6. Area and performance experiments on a Xilinx Virtex-5 FPGA. The area overhead of the protected hardware implementation is quite modest; the effect on clock period is more pronounced and depends quite significantly on Δ (maximum offset between clocks). Scaling factor on the x-axis (i.e., 0.33 ns) corresponds to the minimum delay of a single buffer.

two Xilinx FPGAs: a Virtex-5 (XC5VLX30) and a Spartan-3A (XC3S400A-4FTG256); we used the standard *Xilinx ISE* toolflow in our experiments.

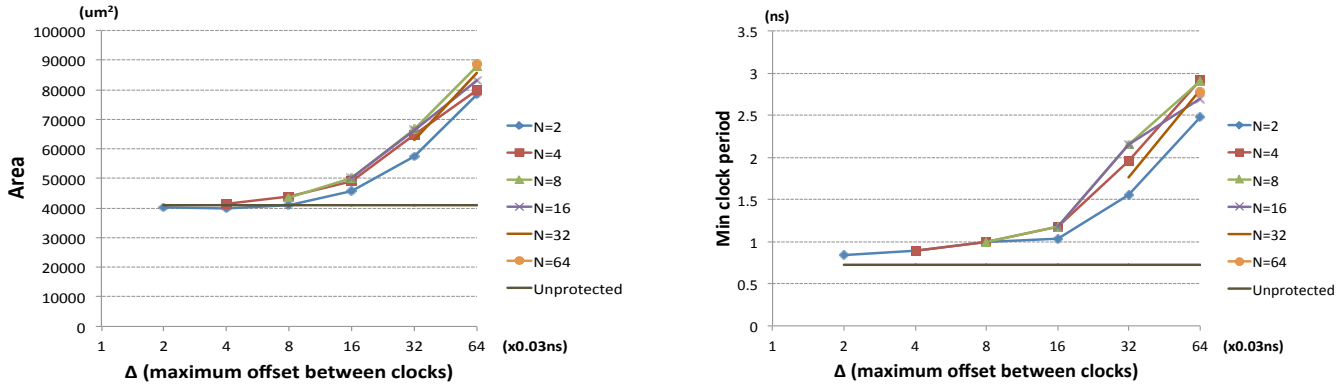
We varied the values of M, N, and Δ using this set-up and obtained $100,000$ power traces for each experiment. Each trace was obtained by running the circuit with a randomly generated input value and saving the instantaneous power consumption during execution of AES. After collecting the traces, we applied a correlation-based power analysis attack [3] in each experiment. We attacked the output of the S-Box operation and used the Hamming-Distance as the power model. To identify relevant points of power traces during each clock cycle, we applied the maximum extraction and integration compression methods [14], and reported the result of the more successful method for each experiment.

2) Results: Fig. 5(a) plots correlation (a proxy for security) as a function of M and N, with δ being held as a constant. Although the best overall results are obtained for M=8, the correlation observed for all values of $M > 1$ are comparable. For $M = 1$, integration and power trace alignment result in stronger attacks; however, increasing M render these techniques ineffective. Increasing N tends to reduce correlation, although the overall reduction yields diminishing returns, especially as N jumps from 8 to 16.

In the other experiments, we set $M = 8$ and vary N and Δ; as we will discuss later, Δ determines the area and performance cost of a circuit.

Fig. 5(b) shows that increasing Δ leads to observably lower correlations; similarly, increasing N up to 16 tends to reduce correlations, but increasing N to 32 or 64 has no discernable effect. The number of traces required to mount a successful attack increases by a factor of k^2 when there is a $k\times$ decrease in correlation [14]. Using this formula, setting $N = \Delta = 16$ increases the number of traces by almost a factor of $70\times$ compared to the unprotected implementation. Increasing Δ to 64 would increase this factor by more than $300\times$.

Fig. 6(a) shows FPGA area (slices) as a function of total delay (Δ). Compared to other hardware countermeasures, the



(a) Area as a function of Δ and N.

(b) Minimum clock period as a function of Δ and N.

Fig. 7. Area and performance experiments for the AES-128 implementation using STM 65nm CMOS standard cells. Compared to the FPGA, the area scales faster, while the clock period scales slower, as a function of Δ (maximum offset between clocks). The reason is the characteristics of buffers in 65nm CMOS technology, relative to other standard cells. The delay elements are required to generate shifted clocks and to prevent timing violations. Scaling factor on the x-axis (i.e., 0.03 ns) corresponds to the minimum delay of a single buffer.

area overhead of our proposed scheme is quite low (see Table I); the largest area overhead for the most secure system we evaluated was 15%.

Fig. 6(b) shows minimum clock period as a function of maximum offset between clocks (Δ). Here, the clock period increases dramatically as Δ increases, resulting in a slowdown of $7\times$ for the design point having the strongest security. For a more conservative data point, $N = \Delta = 16$, the slowdown is $2.3\times$ instead, with significant protection achieved.

Lastly, we measured the power consumption of protected and unprotected version of the AES-128 circuit running on the FPGA. The unprotected circuit consumes 297 mW, while the protected circuits consume as much as 360 mW, an overhead of 20% in the worst case.

B. ASIC experiments

1) Experimental setup: Our ASIC experiments used a 65nm STM CMOS standard cell library. We used *Synopsys Design Compiler* for synthesis, *Cadence Encounter* for placement and routing, *Mentor Graphics Modelsim* for simulations, and *Synopsys Nanosim* for power estimation. Area results are reported after placement and routing, performance results are obtained after simulations (considering post-place-and-route critical path delays), and power results are reported by *Nanosim* after running the same set of random inputs on each implementation.

2) Results: Fig. 7 shows area and clock period as a function of Δ. Compared to the FPGA implementation, the area scales faster, while the clock period scales slower. This is due to the characteristics of the buffers used in CMOS technology: relative to other gates, they have a small delay and a large area. Buffers are used to generate the shifted clocks and are also inserted by the EDA tools along register-to-register paths to fix hold-time violations that may be caused by the clock uncertainty that is inherent to our protection mechanism.

The power overhead is also more pronounced for ASIC technology compared to the FPGA: the unprotected implementation consumes 10.2 mW, while the most aggressively protected implementation consumes 17.5 mW; a moderately protected implementation, $N = \Delta = 16$, consumes 12 mW.

C. Discussion

Our results have demonstrated that clock period overhead (performance) is the primary factor that limits the effectiveness of our proposed countermeasure; the impact on area and power consumption is far less compared to other hardware countermeasures, as shown in Table I. Historically, power analysis attacks have focused on low-cost embedded devices which run at low speeds [14], such as smart cards, and the negative impact on clock period is not expected to be a significant impediment to widespread adoption of this countermeasure. Moreover, faster devices which have greater performance requirements can still benefit from more conservative applications of our countermeasure that limit the values of parameters N and Δ.

Our results indicate that our countermeasure can be applied to both ASIC and FPGA design flows, which establishes portability. We did not apply a security analysis to the ASIC design flow due to constraints on time and computational resources available to us: the number of traces required to attack the ASIC design would be on the order of hundreds of thousands [14], and detailed power simulation at the SPICE-level would take years to complete, even if parallelized across multiple machines.

V. Related Work

Security is an important design parameter, but does not come for free. Protected logic styles, such as *Sensed Amplified Based Logic (SABL)* [23], *Wave Dynamic Differential Logic (WDDL)* [24] and *Masked Dual-Rail Precharge Logic (MDPL)* [17], [18] are very popular hardware countermeasures; however, they have significant area and energy overhead, as shown in Table I, making them not so practical for protecting embedded devices. Furthermore, some of these countermeasures have security weaknesses; e.g., MDPL and

TABLE I
EXISTING COUNTERMEASURES.

Countermeasure	Ratio (protected/standard) Area	Clk Period	Energy	SDF[†]/Tech.[‡]
WDDL [24] (ASIC)	3.38	2.13	3.5	✔ /✔
MDPL [18], [10] (ASIC)	4–5	2	17.43 [19]	✔ /✔
iMDPL [17], [10] (ASIC)	18–19	3.33	-	✔ / ✔
SABL [23] (ASIC)	>2	<2	4.5	✘ / –
This work[(*)] (FPGA)	1.10	2.27	1.15	✔/✘
This work[(*)] (ASIC)	1.25	1.64	1.18	✔/✘

(†) Whether standard cell design flow is applicable.
(‡) Whether it needs to be revised across different technologies.
(*) For $N = \Delta = 16$.

iMDPL have shown to be vulnerable against power analysis attacks due to the early propagation effect [17] and routing imbalances between complementary mask trees [15].

Another common protection method is power trace misalignment, such as using a *randomized clock* [2], [8], [27], inserting *random delays* [4] and using *multiple clock domains* [9]. Randomized clock countermeasures use a single system clock which switches at random discrete time instants. Since these countermeasures do not change the overall power consumption behavior of the circuit, but only misaligns the power consumption traces, alignment preprocessing techniques such as *elastic alignment* [26] and *rapid alignment method (RAM)* [16] could be used to defeat such countermeasures. The random delay insertion method proposed by Bucci et al. [4] uses random number of delay buffers after each memory element on the data path. Since this countermeasure uses single system clock for all memory elements, the power consumption caused by the memory elements at the clock switches, which usually dominates the overall consumption, is aligned and could be exploited by the attacker. Gürkaynak et al. [9] proposed globally-asynchronous locally-synchronous implementation of AES composed of three blocks each having a local clock and a synchronization interface between them. The design is not generic; it is AES-specific and the blocks are hardwired.

Our design also uses the idea of power trace misalignment; however, since each element (or groups of elements), whether sequential or combinational, consumes power *independently* at random moments, the total power consumption of the circuit is randomized, and the overall circuit is robust against alignment techniques [5], [14], [16], [26] even if the attacker knows the amount of delays.

VI. CONCLUSIONS

This paper introduces a new hardware countermeasure to protect digital hardware implementations of cryptographic algorithms against side-channel attacks. The technique is fully automated and is compatible with commercial FPGA and standard cell ASIC design flows. We explored the design space of our technique to empirically derive the tradeoffs involved between protection (security), performance, area overhead, and energy consumption; these overheads are shown to be much smaller than competing countermeasures. Another benefit of our countermeasure is that it appears to be robust against process variations, which are expected to become increasingly-pronounced as CMOS technology scales.

REFERENCES

[1] M.-L. Akkar and C. Giraud, "An implementation of DES and AES, secure against some attacks," in *CHES '01*, 2001, pp. 309–318.
[2] K. H. Boey, Y. Lu, M. O'Neill, and R. Woods, "Random clock against differential power analysis," in *APCCAS '10*, 2010, pp. 756–759.
[3] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *CHES '04*, 2004, pp. 16–29.
[4] M. Bucci, R. Luzzi, M. Guglielmo, and A. Trifiletti, "A countermeasure against differential power analysis based on random delay insertion," in *ISCAS '05*, 2005, pp. 3547–3550.
[5] C. Clavier, J.-S. Coron, and N. Dabbous, "Differential power analysis in the presence of hardware countermeasures," in *CHES '00*, 2000, pp. 252–263.
[6] J.-S. Coron and L. Goubin, "On Boolean and arithmetic masking against differential power analysis," in *CHES '00*, vol. 1965, 2000, pp. 231–237.
[7] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *CHES '01*, May 2001, pp. 251–261.
[8] T. Güneysu and A. Moradi, "Generic side-channel countermeasures for reconfigurable devices," in *CHES '11*, 2011, pp. 33–48.
[9] F. Gürkaynak, S. Oetiker, H. Kaeslin, N. Felber, and W. Fichtner, "Improving DPA security by using globally-asynchronous locally-synchronous systems," in *ESSCIRC '05*, 2005, pp. 407–410.
[10] M. Kirschbaum and T. Popp, "Evaluation of a DPA-resistant prototype chip," in *ACSAC '09*, 2009, pp. 43–50.
[11] C. Klein, O. Cret, and A. Suciu, "Design and implementation of a high quality TRNG in FPGA," in *ICCP '08*, 2008, pp. 311–314.
[12] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems," in *CRYPTO '96*, 1996, pp. 104–113.
[13] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *CRYPTO '99*, 1999, pp. 398–412.
[14] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks - Revealing the Secrets of Smart Cards*. Springer, 2007.
[15] A. Moradi, M. Kirschbaum, T. Eisenbarth, and C. Paar, "Masked dual-rail precharge logic encounters state-of-the-art power analysis methods," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 9, pp. 1578–1589, 2012.
[16] R. A. Muijrers, J. G. Woudenberg, and L. Batina, "RAM: Rapid Alignment Method," in *CARDIS '11*, 2011, pp. 266–282.
[17] T. Popp, M. Kirschbaum, T. Zefferer, and S. Mangard, "Evaluation of the masked logic style MDPL on a prototype chip," *CHES '07*, pp. 81–94, 2007.
[18] T. Popp and S. Mangard, "Masked dual-rail pre-charge logic: DPA-resistance without routing constraints," *CHES '05*, pp. 172–186, 2005.
[19] ——, "Implementation aspects of the DPA-resistant logic style MDPL," in *ISCAS '06*, 2006, pp. 2913–2916.
[20] http://staff.aist.go.jp/akashi.satoh/SASEBO/pdf/SASEBO-GII_Spec_Ver1.01_English.pdf, November 2009, Side-channel Attack Standard Evaluation BOard SASEBO-GII Specification.
[21] A. Shamir and E. Tromer, "Acoustic cryptanalysis: On nosy people and noisy machines," http://www.cs.tau.ac.il/~tromer/acoustic/, 2004.
[22] S. Tillich, C. Herbst, and S. Mangard, "Protecting AES software implementations on 32-bit processors against power analysis," in *ACNS '07*, 2007, pp. 141–157.
[23] 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 *ESSCIRC '02*, 2002, pp. 403–406.
[24] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation," in *DATE '04*, 2004, pp. 246–251.
[25] Z. Toprak and Y. Leblebici, "Low-power current mode logic for improved DPA-resistance in embedded systems," in *ISCAS '05*, 2005, pp. 1059–1062.
[26] J. G. van Woudenberg, M. F. Witteman, and B. Bakker, "Improving differential power analysis by elastic alignment," in *CT-RSA '11*, 2011, pp. 104–119.
[27] Y. Zafar, J. Park, and D. Har, "Random clocking induced DPA attack immunity in FPGAs," in *ICIT '10*, 2010, pp. 1068–1070.