A near-instantaneous and non-invasive erasure design technique to protect sensitive data stored in secure SRAMs

Jean-Philippe Noel, Manuel Pezzin, Jean-Frédéric Christmann, Lorenzo Ciampolini, Mikael Le Coadou, Mariam Diallo, Florent Lepin, Benjamin Blampey, Simone Bacles-Min, Romain Wacquez, et al.

A Near-Instantaneous and Non-Invasive Erasure Design Technique to Protect Sensitive Data Stored in Secure SRAMs

J.-P. Noel, M. Pezzin, J.-F. Christmann, L. Ciampolini, M. Le Coadou, M. Diallo, F. Lepin, B. Blampey[1], S. Bacles-Min, R. Wacquez[1,2] and B. Giraud
Univ. Grenoble Alpes, CEA, LIST, F-38000 Grenoble
[1]Univ. Grenoble Alpes, CEA, LETI, F-38000 Grenoble
[2]CEA Tech, Systèmes et Architectures Sécurisées (SAS), Centre CMP, Equipe Commune CEA Tech - Mines Saint-Etienne, Gardanne
jean-philippe.noel@cea.fr

***Abstract*—On-chip memories, and in particular SRAMs, are among the most critical components in terms of data security because they might contain sensitive data such as secret keys. Whenever a tampering event is detected, one should be able to erase efficiently and rapidly the full content of a memory holding such sensitive data, but current solutions based on simple power-off lead to very long erasure times. In this paper, we present a non-invasive design technique based on an innovative mechanism to remove electric charges from SRAM bitcells still powered on, before refreshing them with a new content not correlated with the previous one. The particularity of this novel hardware countermeasure is to be natively compatible with any SRAM circuit designed from pushed-rule foundry bitcells. We have designed and characterized an 8kB SRAM in 22nm FD-SOI process technology exploiting the proposed security strategy demonstrating an erase operation accomplished in the nanosecond time scale (versus 295µs with the conventional power-off solution) at the cost of an additional area of less than 5%. We have also shown that our solution is more efficient than a solution without prior erasure consisting in writing identical data to all memory addresses in a single clock cycle (1 ns). The use of the latter drops the ratio of zeroized addresses at 92%, while increasing the operating energy consumption by 2.1x under nominal operating conditions.**

I. Introduction

Today, SoC design faces new challenges such as being able to take full advantage of smart and secure cyber-physical systems (CPS) in the demanding world of IoT. Design efforts therefore focus on two distinct areas: (1) how to deliver high performance while minimizing energy consumption and (2) how to provide both cryptographic-quality "roots of trust" in silicon and resistance to physical side-channel attacks with minimal area overhead [1-2]. Among the main elements of a CPS, on-chip memories, and in particular SRAMs, are probably the most critical in terms of performance and vulnerabilities to attacks because they might contain sensitive data (such as secret keys) [3]. In this context, the detection of malicious tampering events (mechanical, electrical or optical) is then the first bulwark of the defense strategy aimed at effectively securing SRAMs, while maintaining high performance in nominal operating conditions [4]. As a response to a tampering event, two types of countermeasure can be applied: (1) software and (2) hardware. Software countermeasures are generally less expensive to implement and sufficiently effective against most threats. However, in some cases, this type of countermeasure is not fast enough. This is particularly the case for data remanence attacks (or cold-boot attacks) [5]. In this case, a well-known hardware countermeasure consisting in powering off the SRAM can be advantageously used [6]. This technique drastically reduces the erasure time (compared to software countermeasures), while being relatively easy to implement. Nevertheless, it may still be too slow for certain sensitive applications (such as key generation). To further reduce the erasure time, the use of an inverting charge pump is an efficient solution but at the cost of a relatively large silicon footprint. With the constant improvement of physical side-channel attack techniques combined with the increase of parasitic elements, which slows down the power-off of SRAMs in advanced CMOS technology nodes (below 28nm), faster, less invasive and more efficient countermeasures must be developed. An alternative solution is to intentionally tamper memory content by writing as quickly as possible a new content defined at design time (*e.g.* zeroization). This consists, for example, in writing new data address-by-address [5][7]. Nevertheless, this design technique presents two major drawbacks. First, the time of self-tampering depends on the memory size (*i.e.* the number of addresses to tamper). Second, the refreshed data is not uncorrelated from the previous one. This means that an attacker can exploit the information leak by side-channel analysis to retrieve the original data [8]. In secure CPS, the notion of temporality is a key factor because the

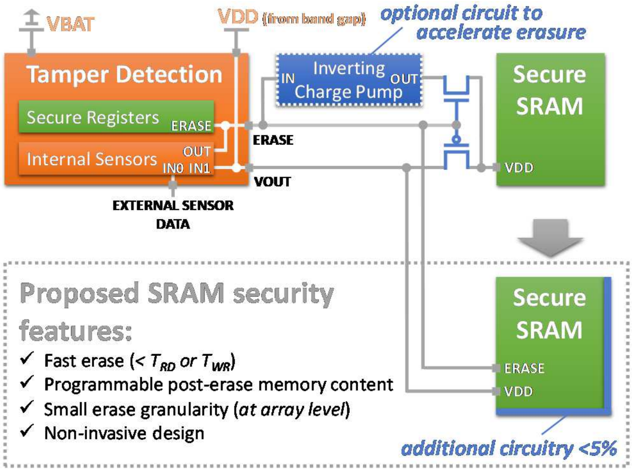


Figure 1 Integration scheme of a secure SRAM erasable after tampering event detections either by (top) powering it off or (bottom) using the proposed fast erase mechanism.

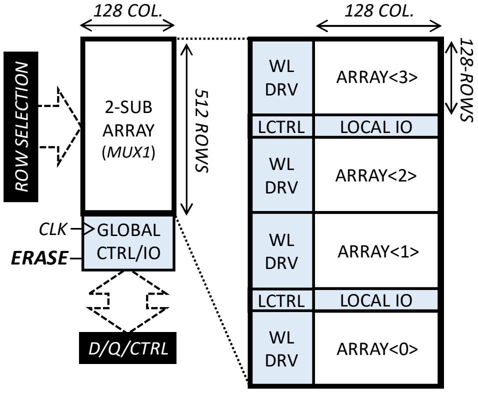


Figure 2 Proposed 8kB (512x128-bit) SRAM architecture based on 2 sub-arrays in MUX-1 configuration.

faster the system reacts to a tampering event, the less the attacker has the chance to recover data. For this reason, a hardware countermeasure must also generate as few information leaks as possible that can be exploited by attackers. In this paper, we propose a design technique that significantly reduces the erasure time, while decorrelating at the same time the memory content with the previous one to minimize the information leaks. In addition, particular attention has been paid to limit both the area overhead and the performance penalty, while remaining compatible with any conventional SRAM circuit designed from pushed-rule foundry bitcells.

The remainder of the paper is organized as follow: Section II describes the proposed non-invasive design technique for quickly erasing the sensitive data of a secure SRAM. Section III presents and compares the measurement results obtained from an 8kB SRAM testchip implemented in 22nm FD-SOI process technology. Finally, Section IV summarizes the paper and presents the application perspectives.

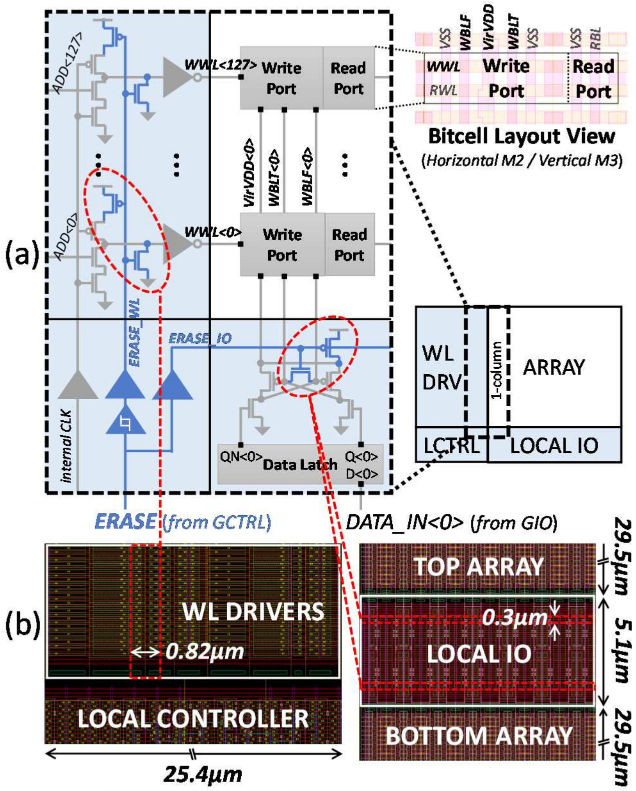
II. Proposed Fast Erase SRAM Circuit

A. *Integration scheme with a tamper detection*

Most of the time, the erasure of the memory content is triggered following the physical detection of a tamper event. For the response to this detection to be as effective as possible, in particular to counter cold-boot attacks, the memory must erase its content as quickly as possible. Figure 1 illustrates an example of the use of a tamper detection controlling the path of the supply voltage of the SRAM to be secured. Conventionally, the erasure of the sensitive data will be carried out by the tamper detection by switching the supply voltage to either ground or a negative voltage (-VDD) to accelerate the operation. In the first case, the erasure time is often very long (few µs to few ms), while in the second case the area overhead is significant.In both cases, the entire memory is erased (even non-sensitive data) but its content after initialization is not mastered. This could be problematic to protect the content of an SRAM used to generate PUFs. The proposed fast erase mechanism makes it possible to respond to all of these issues by integrating it at a lower cost at memory cut or compiler level.

B. *Circuit description & implementation*

Figure 2 represents the 8kB SRAM testchip manufactured in Globalfoundries 22nm FD-SOI technology. It is composed of 2 sub arrays of 256 addresses of 128 bits in MUX-1 configuration (*i.e.* 256 rows of 128 columns). Each of these sub arrays consists of two bitcell matrices of 128 rows of 128 columns. Access to each matrix is via a shared local IO (including the write and read circuitry connected to the BLs) and the WL drivers controlled by an internal clock signal supplied by the local controller. The special feature of this testchip is that it does not include address decoding circuitry to allow multiple selection of write WLs (WWLs). Finally, the bitcell used for its design is a pushed-rule foundry two-port (8T). Figure 3 (a) shows the implementation at schematic level of the proposed fast erase mechanism in the different elements of the sub-arrays, mainly at the level of WL and write drivers. Concerning the WL driver, two transistors per bitcell row have been added to allow simultaneous activation of all the WWLs when the erase signal is activated. Then, in the write driver of the local IO, two transistors have been also added to connect the write BLs (WBLT & WBLF) between them and to disconnect the bitcell column from its supply voltage (VDD). Figure 3 (b) shows the impact of integrating these modifications at layout level, while Figure 3 (c) quantifies it in terms of additional cost: 1% of the overall area of the 8kB SRAM and estimated at 3.2% for a memory capacity reduced to 256B. Figure 4 shows a comparison between the erase mechanisms by



(c)

Memory Capacity	Area Overhead		
	WL Drivers	Local IO	Total
512 x 128-bit (8kB)	3.3%	6.3%	1%
64 x 128-bit (256B)			3.2%*

*extrapolated from testchip floorplan

Figure 3 (a) Schematic and (b) layout views of the proposed fast erase mechanism (additional transistors in dark blue) in the sub array of the 1R1W 8kB SRAM in MUX-1 configuration and (c) the area overhead (also extrapolated for a 256B SRAM).

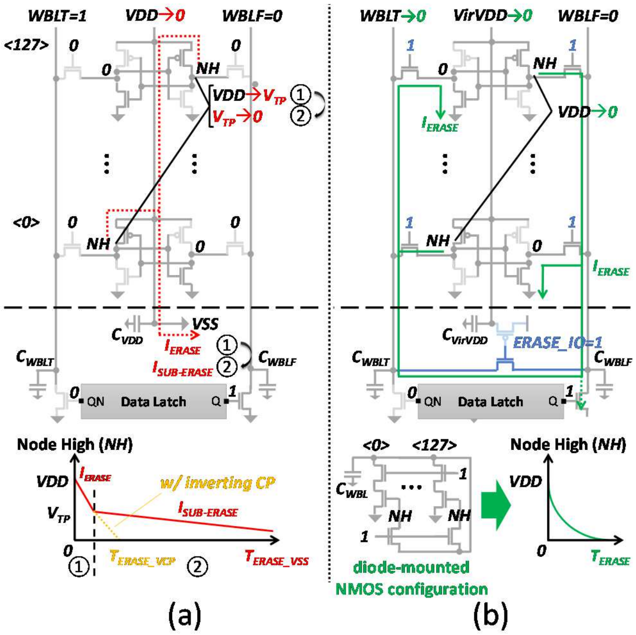


Figure 4 Erase mechanism when the SRAM is (a) powered-off and (b) driven by the proposed fast erase circuit (bitcell columns in diode-mounted NMOS configuration).

(a) by powering off and by (b) using the modifications depicted in Figure 3. In Figure 4 (a), when VDD is forced to ground, in each bitcell only one pull-up PMOS (PU) can discharge the internal node from '1' to '0'. Morever, as soon as the internal node (node high, NH) reaches a value close to the threshold voltage of the PU (V_{TP}), the latter operates in weak-inversion region and considerably slows down the erasure time. The most effective way to speed up the erase operation is thus to apply a negative supply voltage via an inverting charge pump. In Figure 4 (b), the activation of the erase signal triggers (1) the opening of all the WWLs then, with a certain delay, (2) the isolation of the supply voltage of each bitcell column and (3) the equalization of the WBLT/WBLF pairs. This puts all the pull down NMOS of each bitcell in the diode-mounted configuration, leading to an acceleration of the evacuation of the electric charges from all the internal nodes (NH) of the bitcells. Figure 5 illustrates that after the memory erasure phase, the bitcells are in an undetermined data state. In the following state (memory initialization), the WWLs are deactivated (controlled by *ERASE_WL* signal) only after the supply voltage of the write drivers and of each bitcell column is restored and once the WBLT/WBLF pairs are electrically disconnected. In this way, new data can be decorrelated from the previous ones (limiting the information leak) and initialized all-at-once for the full array from data hold in write latches.

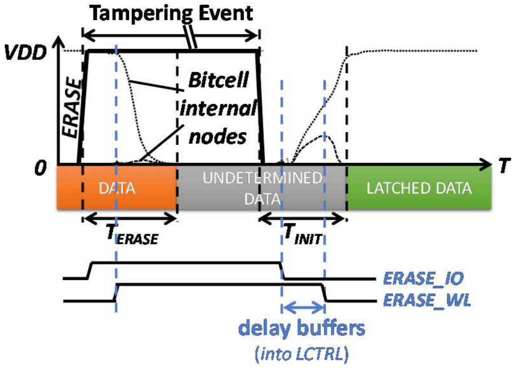
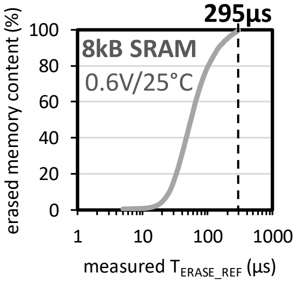


Figure 5 Diagram of the operating sequence of the fast erase mechanism triggered by a tampering event.



VDDA/P (V)	0.6	0.85
T_{ERASE_REF} (ns)	295.10^3	▨
T_{ERASE_FAST} (ns)	<150	
	0.5*	0.15*
T_{INIT_FAST} (ns)	0.7*	0.3*

*post-layout Spice simulations (*TT/25°C*)

Figure 6 Measured erasure time by powering off a standard 8kB SRAM and measured/simulated erasure/initialization time using the fast erase mechanism of the proposed 8kB SRAM.

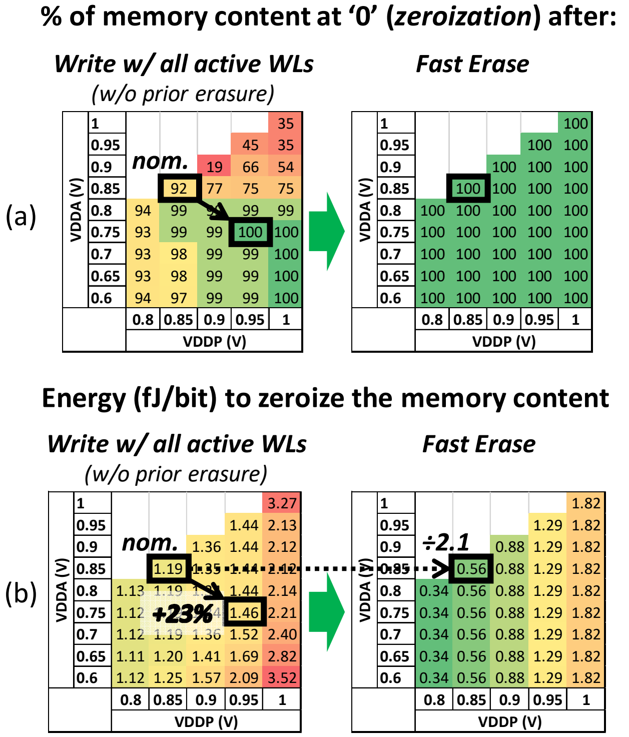
III. Measurement Results

A. *Erasure time*

Figure 6 shows the reference erasure time measured on a different 8kB SRAM manufactured in the same CMOS technology (using the same process options) and coming from a commercial memory compiler supported by the foundry. In this case, the SRAM was powered off for increasing time durations, allowing to acquire the cumulative distribution of the erased bitcell data, as shown in the left part of Figure 6. The resulting time of 295µs at 0.6V is significantly longer than that obtained with our solution, which was less than 150ns (the current testchip design did not allow us to measure erasure times less than 150ns, a limitation that will be drop in future releases). Post-layout Spice simulations at room temperature (25°C) and typical process have then been performed in order to improve the accuracy of the comparison. The erasure time is defined by the time interval that begins when the rising edge of the erase signal reaches 50% of VDD and ends when all internal nodes of the bitcells are equalized. The initialization time is defined by the time interval from the falling edge of the erase signal reaching 50% of VDD to the internal node (NH) of the slowest bitcell reaching 90 % of VDD. This condition is only valid if all the memory addresses have been correctly written with the value held in the write latch. As shown in the right part of Figure 6, the simulated erasure time of the proposed fast erase mechanism is 0.5ns and even decreases down to 0.15ns at 0.85V (nominal VDD). As for the initialization time, it is of the same order of magnitude ranging from 0.7ns to 0.3ns at 0.6V and 0.85V, respectively. This reduction is explained by the fact that the more the supply voltage increases, the faster the diode-mounted transistors evacuate electrical charges.

B. *Erasure efficiency & energy consumption*

Two methods are available in the proposed testchip to initialize the full memory content at '0' (zeroization): either the latch-based mechanism, described in section II, or a massively parallel write '0' operation performed by switching on all WLs. The worst-case energy cost of both methods has been evaluated by writing at first a solid '1' in a traditional way to all memory



(a) *Write w/ all active WLs (w/o prior erasure)*

VDDA (V) \ VDDP (V)	0.8	0.85	0.9	0.95	1
1					35
0.95				45	35
0.9			19	66	54
0.85		92	77	75	75
0.8	94	99	99	99	99
0.75	93	99	99	100	100
0.7	93	98	99	99	100
0.65	93	98	99	99	100
0.6	94	97	99	99	100

(a) *Fast Erase*

VDDA (V) \ VDDP (V)	0.8	0.85	0.9	0.95	1
1					100
0.95				100	100
0.9			100	100	100
0.85		100	100	100	100
0.8	100	100	100	100	100
0.75	100	100	100	100	100
0.7	100	100	100	100	100
0.65	100	100	100	100	100
0.6	100	100	100	100	100

(b) *Write w/ all active WLs (w/o prior erasure)*

VDDA (V) \ VDDP (V)	0.8	0.85	0.9	0.95	1
1					3.27
0.95				1.44	2.13
0.9			1.36	1.44	2.12
0.85		1.19	1.35	1.44	2.12
0.8	1.13	1.19	[illegible]	1.44	2.14
0.75	1.12	[illegible]	[illegible]	1.46	2.21
0.7	1.12	1.19	1.36	1.52	2.40
0.65	1.11	1.20	1.41	1.69	2.82
0.6	1.12	1.25	1.57	2.09	3.52

(b) *Fast Erase*

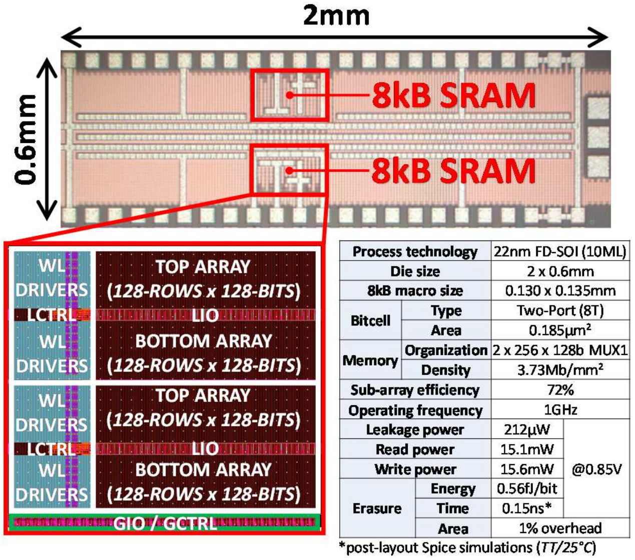
VDDA (V) \ VDDP (V)	0.8	0.85	0.9	0.95	1
1					1.82
0.95				1.29	1.82
0.9			0.88	1.29	1.82
0.85		0.56	0.88	1.29	1.82
0.8	0.34	0.56	0.88	1.29	1.82
0.75	0.34	0.56	0.88	1.29	1.82
0.7	0.34	0.56	0.88	1.29	1.82
0.65	0.34	0.56	0.88	1.29	1.82
0.6	0.34	0.56	0.88	1.29	1.82

Figure 7 Shmoo plot of measured (a) percentage of memory content at ‘0’ and (b) energy consumption using write with all active WWLs (left) and fast erase (right).

addresses (in 512 write cycles, *i.e.* 512ns at 0.85V). For the proposed fast initialization mechanism, a ‘0’ is recorded in the write latches without activating any WWLs. Finally, the erase signal is activated for 150ns. For the massive parallel write mechanism, all of the WWLs are activated during a write ‘0’ operation during a single cycle (1ns). Figure 7 (a) shows the percentage of correctly zeroized memory addresses for supply voltages ranging from 1 to 0.8V for the periphery (VDDP) and down to 0.6V for the bitcell arrays (VDDA) at room temperature. Under all these conditions, the rate of zeroizing is 100% using the fast erase mechanism, while it drops to 92% in the nominal condition (0.85V) using the parallel write, because the write drivers in local IO have not been oversized to drive 128 WWLs to spare area and power. A 100% of zeroizing can be achieved with this technique if VDDP is increased by 100mV (up to 0.95V), while decreasing VDDA by 100mV (down to 0.75V). This increases the erasure energy by 23% whereas this is already 2.1x higher in nominal condition than the energy consumed by the proposed fast erase mechanism (Fig. 7 (b)). Finally, Figure 8 shows the die micrograph and the testchip summary at 0.85 V.

IV. Conclusion

This paper proposes a near-instantaneous and non-invasive erasure design technique to efficiently protect the sensitive data stored in secure SRAMs. This novel hardware countermeasure, applied in response to a tampering event, has been developed to compensate for the relatively long erasure time when powering off SRAM, especially in advanced CMOS technology nodes. We have thus experimentally demonstrated that the erasure


2mm
0.6mm
8kB SRAM
8kB SRAM
WL DRIVERS
TOP ARRAY (*128-ROWS x 128-BITS*)
LCTRL
LIO
WL DRIVERS
BOTTOM ARRAY (*128-ROWS x 128-BITS*)
WL DRIVERS
TOP ARRAY (*128-ROWS x 128-BITS*)
LCTRL
LIO
WL DRIVERS
BOTTOM ARRAY (*128-ROWS x 128-BITS*)
GIO / GCTRL

Process technology		22nm FD-SOI (10ML)	
Die size		2 x 0.6mm	
8kB macro size		0.130 x 0.135mm	
Bitcell	Type	Two-Port (8T)	
	Area	0.185µm²	
Memory	Organization	2 x 256 x 128b MUX1	
	Density	3.73Mb/mm²	
Sub-array efficiency		72%	
Operating frequency		1GHz	
Leakage power		212µW	@0.85V
Read power		15.1mW	
Write power		15.6mW	
Erasure	Energy	0.56fJ/bit	
	Time	0.15ns*	
	Area	1% overhead	

*post-layout Spice simulations (*TT/25°C*)

Figure 8 Die micrograph and chip summary @0.85 V.

time when switching off supply voltage (0.6V@25°C) of an 8kB SRAM fabricated in 22nm FD-SOI process technology was 295µs versus an estimated 0.15ns (2.10^{6}x reduction) using the proposed fast erase mechanism. Furthermore, this solution limits the possible information leaks by completely decorrelating the new memory content with the previous one. An alternative erasure strategy based on a complete rewriting of the data without prior erasure has been demonstrated to be not as effective in a single clock cycle (1ns) because the rewrite is incomplete (92% of addresses zeroized) and increases energy consumption by 2.1x under nominal operating conditions. Finally, the proposed solution is also competitive both in terms of area (<5% overhead) and memory performance (no significant increase in leakage currents and timing) thanks to its native compatibility with any conventional SRAM circuit designed from pushed-rule foundry bitcells. In perspective, this design technique could be also advantageously used to design efficient and secure SRAM-based PUF or RNG providing cryptographic-quality “roots of trust” in silicon.

References

[1] P. Kocher, “Complexity and the challenges of securing SoCs”, DAC, pp. 328–331, 2011.

[2] A. Ehret *et al.*, “A Survey on Hardware Security Techniques Targeting Low-Power SoC Designs”, HPEC, pp. 1–8, 2019.

[3] S. Malliaros, C. Ntantogian and C. Xenakis, “Protecting Sensitive Information in the Volatile Memory from Disclosure Attacks”, ARES, pp. 687–693, 2016.

[4] D. Serpanos and D. Stachoulis, “Secure Memory for Embedded Tamper-proof Systems”, DTIS, pp. 1–4, 2019.

[5] W.-G. Ho *et al.*, “Area-efficient and low stand-by power 1k-byte transmission-gate-based non-imprinting high-speed erase (TNIHE) SRAM”, ISCAS, pp. 698–701, 2016.

[6] K. Wenjing *et al.*, “Novel security strategies for SRAM in powered-off state to resist physical attack”, ISIC, pp. 298–301, 2009.

[7] A. Srivastava and P. Ghosh, “An Efficient Memory Zeroization Technique Under Side-Channel Attacks”, VLSID, pp. 76–81, 2019.

[8] R. Giterman *et al.*, “Power Analysis Resilient SRAM Design Implemented with a 1% Area Overhead Impedance Randomization Unit for Security Applications”, ESSCIRC, pp. 69–72, 2019.