MWA Skew SRAM Based SIMPL Systems for Public-Key Physical Cryptography

Qingqing Chen[1(✉)], Ulrich Rührmair[2], Spoorthy Narayana[1], Uzair Sharif[1], and Ulf Schlichtmann[1]

[1] Institute for Electronic Design Automation, Technische Universität München, Munich, Germany
{qingqing.chen, spoorthy.narayana, uzair.sharif, ulf.schlichtmann}@tum.de
[2] Horst Görtz Institute for IT Security, University of Bochum, Bochum, Germany
ruehrmair@ilo.de

Abstract. SIMulation Possible, but Laborious (SIMPL) systems are a novel cryptographic concept for physical cryptography that have been suggested in recent years. They can potentially solve inherent vulnerabilities of conventional public-key cryptography that is based on unproven mathematical hypotheses. The security of SIMPL systems rests on their physical unclonability and on the runtime difference between the real-time behavior of the unique SIMPL system and any adversarial simulation or emulation of it. One first circuit-based realization of SIMPL systems via so-called skew SRAMs has previously been discussed in the literature. This paper presents an approach to enhance the security of skew SRAM based SIMPL systems by introducing more complicated and parallel computing behavior taking place in the skew SRAM, which we call multiple-wordline-activation (MWA) skew SRAM. Simulations of the MWA skew SRAM show expected behavior complexity that can be taken advantage of in SIMPL systems to amplify the speed advantage over emulators (functional physical clones) or simulators (digital clones), which plays a key role in the security of SIMPL systems.

Keywords: Security · Cryptography · Physical cryptography · Public-key physical cryptography · Physical unclonable function · Simulation possible, but laborious system · SIMPL system · Public physical unclonable function, public PUF · Skew SRAM · MWA skew SRAM · Multiple wordline activation

1 Introduction

SIMulation Possible, but Laborious systems (SIMPL systems or just SIMPLs) [1, 2] are a novel cryptographic concept within so-called physical cryptography. Unlike physical unclonable functions (PUFs) [3–7], whose aim is to resolve inherent issues of conventional *private-key* cryptography, SIMPL systems and public PUFs [8] are new cryptographic primitives for typical *public-key* like scenarios.

A PUF is a physical function that maps challenges (inputs) to responses (outputs) depending on the physical phenomena taking place in the PUF structure. Based on the

M. Conti et al. (Eds.): TRUST 2015, LNCS 9229, pp. 1–15, 2015.
DOI: 10.1007/978-3-319-22846-4_16

challenge-response pairs (CRPs), a PUF is able to serve in various cryptographic protocols [9], e.g., challenge-response authentication. However, all these protocols require a piece of previously shared information that must be kept secret, restricting the application of PUFs as secret-key like primitives. Similar to a PUF, every SIMPL system also realizes an individual function that maps challenges to responses. Unlike PUFs, however, any SIMPL system possesses an individual public numeric simulation program. This allows everyone to simulate and calculate the responses of the SIMPL system, and therefore to verify the correctness of the responses received from the party who claims to be in possession of the original SIMPL system. As the SIMPL system is designed such that any adversarial simulation or hardware emulation of it is slower and more time consuming than the real-time behavior of the original SIMPL system, which means that only the person physically holding the original SIMPL system can compute the responses at a certain speed (equal to or faster than a threshold speed value that is publicly available, together with or given by the simulation program), everyone may verify the realness of the claimant's possession of the SIMPL system by checking the *quickness* (speed) of the responses with a timer and the *correctness* with the simulation program. It has been shown that the speed advantage or "speed gap" can be used for a number of different cryptographic protocols, including identification, key exchange, bit commitment, zero-knowledge protocols, and digital rights management applications [1].

1.1 SIMPL Systems and Public PUFs

SIMPL systems and public PUFs are actually equivalent cryptographic concepts. However, recent research on both primitives has shown a slightly different accentuation. Publications on public PUFs have mostly focused on nanoelectronic solutions, trying to achieve an *exponential* time gap between the public PUF hardware and any adversaries [10, 11]. While being scientifically highly interesting, such exponential time gaps between hardware and its simulation may be hard to achieve in practical, inexpensive, and stable implementations. Furthermore, an exponential time gap makes the simulation of the public PUF, which is one inherent protocol step, very time consuming, leading to practically inefficient schemes. For these reasons, recent investigations on SIMPL systems followed a different route [1, 2, 12, 13]: They examined practical, circuit-based implementations with a sufficiently large, but constant time gap. In order to avoid attacks, the SIMPL system implemented a non-parallelizable function, whose computation is closely tied to the maximal clock frequency of today's integrated circuits. It is well-known that this frequency cannot be raised indefinitely due to physical constraints imposed by semiconductor materials. This strategy promises to thwart many attacks while still maintaining practicality.

We follow and extend this route in this paper, attempting to achieve a substantial improvement over previous skew-SRAM-based designs of SIMPL systems: We try to lift the relative security margin from a small factor of around 2 to larger factors of $2n$ by a new, so-called multiple wordline activation (MWA) skew SRAM, in which n represents the number of wordlines that can possibly be activated simultaneously in skew SRAM write operations.

1.2 Implementation of SIMPL Systems via Skew SRAMs

A first implementation of SIMPL systems known as skew SRAM based SIMPL systems has been introduced in [2, 12, 13]. It uses specially designed ("skew") SRAM architectures to outperform their emulators/simulators when executing certain, specialized computing tasks that skew SRAMs are designed for. Figure 1 shows the structure of the skew SRAM based SIMPL system of [12, 13]. The system consists of four main blocks, namely a skew SRAM block (SS), a challenge control block (CC), a voltage control block (VC), and a feedback and output control block (FOC).

Similar to PUFs, SIMPL systems calculate responses R_i when fed with challenges C_i. The challenge control block CC "scrambles" the challenge and generates voltage select (*SEL*), read/write address (*ADR*), write data (D_{IN}) and other control (*CTRL*) signals, e.g., write enable, for the skew SRAM. References [12, 13] suggested realizing the challenge control block with a hash function. The skew SRAM block SS is designed similar to normal SRAMs, except that different types of cells (normal cells and skew cells, of which skew cells differ from normal, i.e., standard SRAM cells in the sizing of their transistors and therefore in their electrical characteristics, specifically their write behavior) are distributed in the array and write operations taking place in those specially designed skew cells may fail to modify their previously stored contents depending on the current supply voltage (controlled by voltage control block VC). To be specific, the skew cells discussed in [12, 13], which were designed for a 0.18-µm CMOS technology, will retain their previously stored values when the *VDD* to the skew SRAM block SS is 1.3 V, and the write operation will be successful (like in conventional SRAM cells) when the *VDD* is 1.8 V. Therefore, writing and reading successively in a cell of such a skew SRAM is effectively a computation process based on the *VDD* and D_{IN} signals, as well as the type and the current contents of the cell.

As the skew SRAM behavior cannot be outperformed by emulators or simulators [13], its speed advantage can be used to distinguish real SIMPL systems from faked ones. The speed advantage of the SIMPL system over emulators/simulators can be amplified if the data read out from the skew SRAM in a cycle is fed back into the challenge control, thus influencing the next cycle. This can of course be repeated multiple times. In [12, 13], the feedback loop is realized with a linear feedback shift register (LFSR) that XOR's the read-out data D_{OUT} of the skew SRAM block. More details of the operation and the design can be found in [12, 13].

By randomly allocating skew as well as normal cells in the skew SRAM array SS, a specific design is achieved. However, individualized SIMPL systems should not be realized by creating many different designs with different skew/normal cell distribution patterns, as it is too expensive to fabricate just one single or several chips out of one design for usual commercial applications. References [12, 13] suggested to randomly modify a portion of the skew and normal cells in the array to become fixed-"0" or fixed-"1" cells, which can be realized with laser fuses and "burn-in" fabrication steps. These fixed-"0" and fixed-"1" cells have their data nodes directly connected to *GND* and *VDD*, respectively. Their stored and read-out values will remain fixed no matter what operation is carried out. Depending on where the fixed-"0" and fixed-"1" cells are located, a practically infinite number of different SIMPL systems can be fabricated out

of one design, with properly chosen parameters including the skew SRAM array sizes and the portions of skew and normal, as well as fixed cells.

In public-key applications, the public key, i.e., the simulator part, implements a function that calculates response R_i when fed with the description D(**S**) of a SIMPL system **S** as well as the challenge C_i (the challenge C_i should be given by the verifier). The simulator should also be able to tell the response time limit t_{max} of the legal SIMPL system **S**, so that a verifier can check both the *quickness* and the *correctness* criteria discussed at the beginning of this section. To create such a public simulation program for a specific SIMPL system, one needs to know the accurate logic functionalities of each block of the system including the configurations of the skew SRAM block, i.e., allocations of normal, skew, as well as fixed cells. The simulation program just needs to implement the logic functionality of the SIMPL system in software, and, based on the specifications, calculate the response time limit t_{max} of the legal SIMPL system.

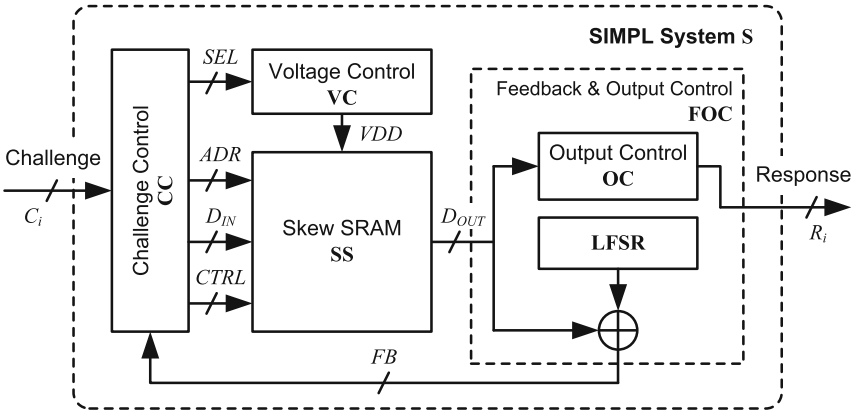


Fig. 1. Schematic illustration of the skew SRAM based SIMPL system [12, 13].

The security of SIMPL systems is determined by the speed/performance advantage which the systems have over all emulators or simulators of them. Specifically for skew SRAM based SIMPL systems, the security results from the speed advantage of skew SRAM write operations over emulators using conventional memories and logic, as well as simulators using standard computing systems [13], which can theoretically mimic each skew SRAM write operation in three sequential steps: (1) read out the contents and the type of addressed cells; (2) compute the result of the write operation; (3) write back the result into addressed cells. However, as technology develops, the speed advantage of a previously fabricated skew SRAM may decrease since emulators/simulators using new technologies become faster, making the secured service lifetime of a skew SRAM based SIMPL system shorter. To further enhance the security of skew SRAM based SIMPL systems and to ensure longer service lifetime, the concept of multiple wordline activation (MWA) skew SRAM is proposed in this paper. This concept attempts to write into multiple cells in each column of a skew SRAM array by activating multiple wordlines simultaneously in a write operation.

The rest of the paper is organized as follows: Section 2 proposes the security enhancement approach using MWA skew SRAM for SIMPL systems. Section 3 presents a design of MWA skew SRAM. Section 4 shows simulations of the design. Section 5 briefly assesses the security enhancement of the MWA skew SRAM and summarizes the paper.

2 MWA Skew SRAM Based SIMPL Systems

As discussed in Sect. 1, the long-term security of skew SRAM based SIMPL systems can be enhanced by increasing the speed advantage executing an operation in the skew SRAM over mimicking it through emulation/simulation. To enlarge the speed asymmetry, multiple wordline activation during write operations is introduced, which allows the attempt of writing the same data appearing on skew SRAM bitlines into multiple cells on simultaneously activated wordlines. Multiple wordline activation is only allowed in write operations, but not in read operations.

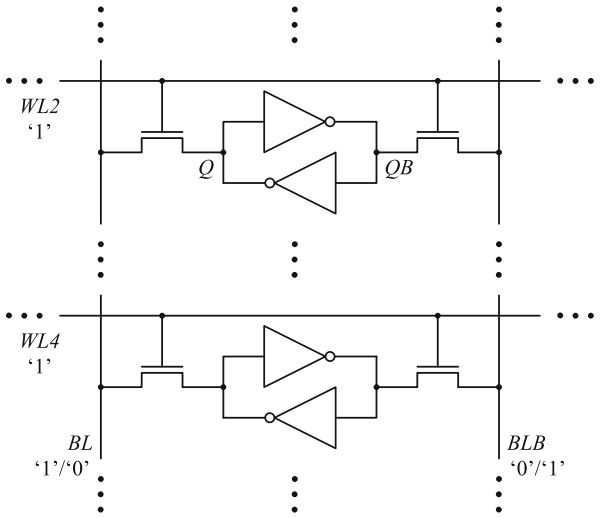


Fig. 2. Part of the schematic of an MWA skew SRAM array, with *WL2* and *WL4* activated at the same time. The two illustrated cells of the same column are activated simultaneously for write operations.

As an example, Fig. 2 shows when wordlines *WL2* and *WL4* are activated (become logic '1') at the same time in a write operation, the write data '1'/'0' appearing in bitline *BL*/*BLB* attempts to overwrite the internal data node *Q*/*QB*, respectively. Depending on the type of the addressed cells, the current supply voltage level, as well as their previously stored contents, the write operation may succeed or fail (i.e., previously stored contents stay unchanged, and are different from the write data). Since the activation of multiple wordlines connects the internal data nodes of multiple cells electrically through

access transistors and bitlines, the operations of these cells influence each other (e.g., if a '1' is to be written, and one of the two activated cells already contains a '1', then writing '1' into the other cell becomes easier), making the computation taking place in write operations even more complicated. As any combination (defined by the challenge control module, see Fig. 1) of wordlines can be activated simultaneously, this parallel write operation can be emulated or simulated only by evaluating the type and the current contents of all the addressed cells together. That would require the emulator/simulator to read all needed information from different addresses, which cannot be done in a single read operation using normal memories or standard computing systems (resources available for attackers making functional physical clones or digital clones discussed in [13]). And the same also holds for the writing of the computed results back into those addresses, which cannot be done in a single write operation. While the SIMPL system can effectively perform *reading*, *computing*, and *writing-back* in one single cycle, an attacker will have to read the necessary data one by one in multiple cycles, compute the results based on all readout information, and write back again in multiple cycles. That would make the attacker's emulator/simulator many times slower, and the factor is dependent on the number of wordlines that can possibly be activated at the same time. Further discussions about the security assessment of the MWA skew SRAM are carried out in Sect. 5.

As the security of the skew SRAM based as well as the MWA skew SRAM based SIMPL system relies on the speed advantage of the skew SRAM block, other blocks described in Sect. 1 may stay unchanged. Section 3 presents a specific design of our MWA skew SRAM that enhances the security of the SIMPL system based on skew SRAM concepts.

3 Design of an MWA Skew SRAM

As a proof of concept, an MWA skew SRAM design using 45-nm PTM nano-CMOS models [14] is presented below, with up to two simultaneously activated wordlines (in each write cycle, one or two wordlines are activated simultaneously for writing) and two different supply voltages (1.0 V and 1.3 V). Two types of skew cells (Type 1 skew cell S_1, and Type 2 skew cell S_2) are designed, which differ in their transistor sizing and thus in their write behavior. They are randomly distributed in the skew SRAM array, with some of them configured to be fixed cells in post fabrication steps [13]. Therefore, four types of fixed cells exist (F_1: S_1 fixed to '0'; F_2: S_1 fixed to '1'; F_3: S_2 fixed to '0'; F_4: S_2 fixed to '1'). All types of cells work normally as conventional SRAM cells do in read operations, but show relatively complicated behavior when data is attempted to be written into them, depending on the combination of cell types activated in the write cycle, the current supply voltage, the previously stored data in the activated cells, as well as the data to be written into them. The defined write behavior is presented in Table 1.

The left-most column specifies the type(s) of the activated cell(s) in write operations, with previously stored data in the parentheses. The upper-most two rows give the operating conditions (supply voltage VDD during the write cycle, and the data WD to be written). Non-bold numbers in the rest of the table are the stored data (result) of corresponding activated cells after the write operation.

Compared to the behavior definition of the skew SRAM cells (normal/skew/fixed cells) described in [12, 13], we name the MWA skew SRAM cells differently, since no cell behaves the same as normal SRAM cells under all conditions. For the MWA skew SRAM, two types of skew cells, which behave differently in write operations, are defined. As described in Table 1, the "$S_2(0)$, $S_2(1)$" line defines that when a Type 2 skew cell (with previously stored data '0') and another Type 2 skew cell (with previously stored data '1') from the same column are activated simultaneously in a write cycle under the condition that VDD is 1.3 volt and the write data is '1', the first S_2 cell succeeds to store the new data '1', while the second S_2 cell remains storing the previously stored data '1', which is the same as the write data. Figure 3 shows the schematic and carefully chosen transistor sizes of Type 1 and Type 2 skew cells, which fulfill the behavior specification of Table 1.

Table 1. Write behavior of MWA skew cells

Cell type[a] (prev. stored data)	VDD = 1.0 V[b]		VDD = 1.3 V	
	WD = 0[b]	WD = 1	WD = 0	WD = 1
S_1	0	1	0	1
S_2	0	1	0	-[d]
S_1, S_1	0, 0	1, 1	0, 0	1, 1
S_1, S_2	0, 0	1, 1	-, -	-, -
$S_2(0)$, $S_2(0)$	0, 0	0, 0	0, 0	0, 0
$S_2(0)$, $S_2(1)$[c]	0, 0	1, 1	0, 0	1, 1[c]
$S_2(1)$, $S_2(1)$	0, 0	1, 1	1, 1	1, 1
$F_x(0)$, S_x	0, 0	0, 0	0, 0	0, 0
$F_x(1)$, S_x	1, 1	1, 1	1, 1	1, 1
F_x, F_x	-, -	-, -	-, -	-, -

[a]The type(s) of cells that are activated simultaneously for write. For example, "S_1, S_2" means that a Type 1 skew cell and a Type 2 skew cell are activated simultaneously (both cells are from the same column, sharing the same bitline). "S_1" only means that one single Type 1 skew cell is activated. "F_x, S_x" stands for a fixed cell of any type and a skew cell of any type activated simultaneously. Data in the parentheses is the previously stored data in the corresponding cell, or the fixed value in case of a fixed cell. Without parentheses means that the previously stored data could be '1' or '0'.

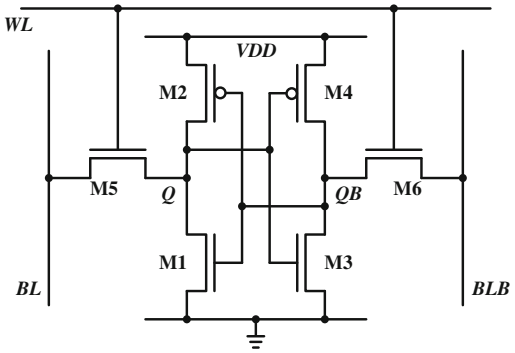
[b]"VDD" is the supply voltage of the current write cycle. "WD" is the data to be written in that cycle.

[c] "1, 1" here means the stored data of the first S_2 (with previously stored data '0') and the second S_2 (with previously stored data '1') cells after the write operation, respectively: Under the condition of VDD=1.3V and WD=1, '1' is successfully written into the first S_2 cell, while the second S_2 cell retains its previously stored data '1', which is the same as the write data.

[d]"-" means that previously stored data remains unchanged after the write operation

Fixed cells are simulated by adding connections between internal nodes *Q* to *VDD* (for Fixed '1' cells) or *GND* (for Fixed '0' cells) through resistors of 600 ohm (a typical value for antifuses using ONO, i.e., oxide-nitride-oxide technologies [15]).

As up to two wordlines need to be activated at the same time in write cycles, the challenge control block (see Fig. 1) and/or the address decoder of the skew SRAM need to be redesigned. A simple solution is to double the address (output of challenge control) bit width for the skew SRAM row decoder which controls the activation of wordlines, and duplicate the row decoder, with each taking half of the row address as input. The two row decoders may produce the same or different outputs, and each wordline is controlled by an "OR"-gate taking the corresponding output bits of the two row decoders as its inputs. Thus, one or two wordlines of the array will be activated simultaneously in each write cycle. In read cycles, the inputs of the two row decoders should be the same, so that only one wordline is activated in read cycles. This can be realized with simple logic based on the state of the *R*/*W* signal (read/write signal, which is part of the *CTRL* signal of Fig. 1).



	Type 1	Type 2
M1, M3	180 / 45	145 / 45
M2, M4	50 / 90	115 / 45
M5, M6	85 / 45	54 / 45

Fig. 3. Schematic of skew cells and transistor sizing (W/L in nanometer) of Type 1 and Type 2 skew cells using 45-nm PTM nano-CMOS models [14].

Other modules, i.e., voltage control and feedback & output control modules, of the skew SRAM based SIMPL system do not require any modification.

4 Simulation

We simulate an MWA skew SRAM column as shown in Fig. 4 to verify the design. In the MWA skew SRAM column, S1 and S2 cells are randomly distributed. Several cells are chosen to be fixed cells. All combinations of situations described in Table 1 are simulated for verification. Figure 5 shows part of the simulation that verifies, e.g., the "S1(1), S2(0)" situation with VDD = 1.3 V and WD = 1 of Table 1 at simulation time 420 ns. It can be seen that the write operation of the S2 cell fails (to overwrite its previously stored value '0', see signal "qs22") as is desired, while the operation of the

S1 cell (see signal "qs12") can be regarded as successful although WD is the same as its previously stored value. Through simulations, all functionalities under different situations defined in the specification of Table 1 have been verified.

To approximately evaluate the yield of the design, simulations of the MWA skew SRAM column of Fig. 4 considering global process variations were carried out to verify the design further. Gaussian distributed variations (with three standard deviations of 30 millivolt, i.e., about 16.7 %) of the threshold voltages of PMOS and NMOS transistors were considered. However, since the simulated circuit is not small and there are a lot of different situations (Table 1) to check, it is time-consuming to run a large amount of Monte Carlo simulations. Our solution was to sweep the process variation parameters and find the parameter space boundary first. Within the boundary, all the specifications of Table 1 should be met. This greatly reduced the number of required simulations. After that, we just need to check if a Monte Carlo sample falls in the yield region or not. Out of all 1,000,000 Monte Carlo samples, 997,334 (over 99.7 %) totally meet the specification of Sect. 3 in all cases.

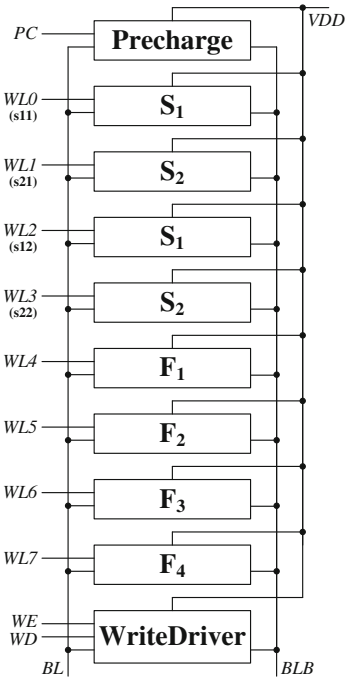


Fig. 4. MWA skew SRAM column design for simulation (PC: precharge; WLn: wordlines; WE: write enable; WD: write data; BL/BLB: bitlines; S1: Type 1 skew cell; S2: Type 2 skew cell; Fx: fixed cells)

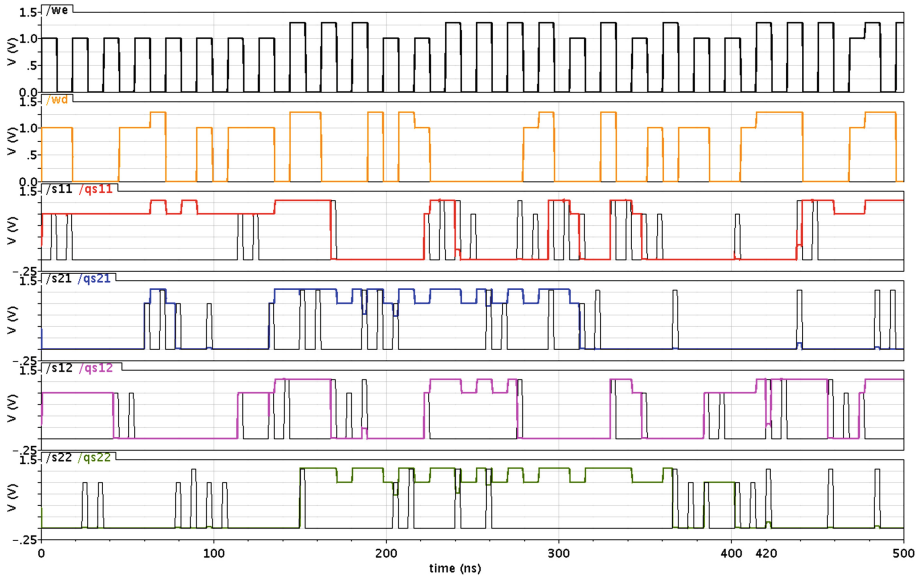


Fig. 5. Waveform of Type 1 and Type 2 skew cell operations. The voltage of logic '1' varies between 1.0 V and 1.3 V as VDD changes. Signals: we (write enable), wd (write data), s11/qs11 (WL/Q of an S1 cell), s21/qs21 (WL/Q of an S2 cell), s12 and qs12 (WL/Q of a second S1 cell), s22 and qs22 (WL/Q of a second S2 cell).

5 Discussion and Conclusion

This section briefly discusses the security improvement of the MWA skew SRAM based SIMPL system over the original described in [2, 12, 13] and concludes the paper.

5.1 Security Assessment

As in [12, 13], three basic possibilities for a faker to imitate the SIMPL system are discussed.

Exact Physical Clone. Similar to the conventional skew SRAM based SIMPL system, the MWA skew SRAM based SIMPL system is also refabricatable in a silicon foundry, if one-time costs of millions of US dollars are available for the faker [12]. However, if the value to be protected by the SIMPL system is much lower than that, which is the usual case for consumers or individual hackers and consumer application scenarios, fabricating an exact physical clone of the system is not only practically difficult, but also pointless from an economic perspective.

Furthermore, it is interesting to compare the attacker's need to fabricate a certain ASIC in silicon to the security level of secret binary keys, say, be they stored in classical memory or in alternative technologies like PUFs. In the past, such individual

keys can and have been read out by professional hacker teams on several occasions [16, 17]. In the case of satellite TV boxes, such attacks have reportedly been mounted by hacker teams hired by the direct competitors of the TV companies [18]. The supposedly protective keys can then be distributed quickly and conveniently over the dark net, since they are fully digital, and can be downloaded and used even by relatively untrained private consumers to commit fraud. The same does not hold for IP protection based on SIMPL systems, as attacks here require the professional fabrication of an ASIC in silicon. This offers a potentially game-changing novel security feature for large consumer markets.

Functional Physical Clone. Building functional physical clones using resources like "normal" (mass-manufactured) memories, logic IC components, PLDs and FPGAs that are available for consumers or individual hackers is discussed.

Figure 6 shows the basic structure of an emulator of the MWA skew SRAM using normal mass-manufactured memories and logic, which mimics the behavior of the MWA skew SRAM and attempts to catch up with it in computing speed. However, since the result of a write cycle is dependent on the type and the previously stored data of addressed cells, the emulator needs to first read out *celltype* and the previously stored data D_{prev} of the addressed cells from the configuration memory and the data memory, respectively, and then calculate (together with the write data D_{IN} and the voltage select signal *SEL*) in the logic block the new data (result of the write cycle) to be written back into the data memory. For an MWA skew SRAM emulator, the data for *celltype* as well as for D_{prev} for different MWA wordlines come from multiple addresses, making it

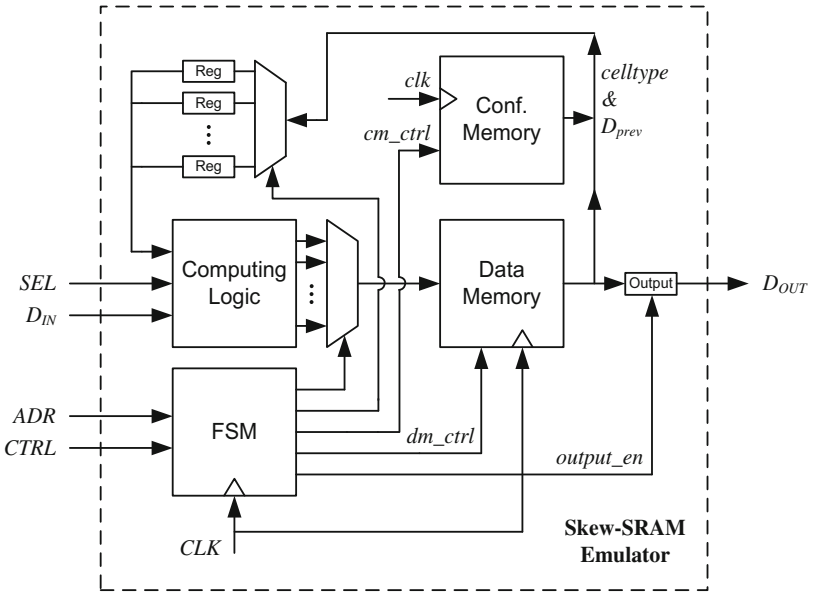


Fig. 6. Basic structure of an MWA skew SRAM emulator based on normal SRAMs and logic

impossible to read in a single cycle. Therefore, *celltype* and D_{prev} must be read out with multiple read operations (to be specific, n read operations where n is the number of simultaneously activated wordlines) and stored at the input of the logic block until all needed information is collected for calculation. For writing back, a similar procedure must be done (to be specific, in n write operations where n is the number of simultaneously activated wordlines), since writing back into multiple addresses simultaneously is not possible. Thus, write back data must be present at the output of the logic block until all data are written back into the data memory cycle by cycle. And a finite state machine (FSM) needs to be implemented to control all the operations, e.g., the cycle-by-cycle read out and write back should be controlled by the FSM through *cm_ctrl* and *dm_ctrl* that contains the addresses and other control signals for each operation cycle. Since a write operation in the MWA skew SRAM has to be substituted with multiple read and multiple write operations in sequence, this emulator is at least a factor of $2 \times n$ slower in write operations where n is the number of wordlines activated, if the configuration and the data memories work at the same speed as our MWA skew SRAM, and if the delay of the computing logic of the emulator is ignored. Realizing the same structure of Fig. 6 with FPGAs wouldn't be dramatically different in speed since the multiple read and write operations have to be done in sequence.

Another possible FPGA-based emulation approach is to replicate the behavior of the MWA skew SRAM block cell by cell. Emulating the logic behavior of a single cell defined in Table 1 is not complicated. However, to be noted is that determining the result of MWA write operations in a Type 1 or Type 2 skew cell (S_1 or S_2) requires some knowledge about the other simultaneously activated cell in the same column (on the same bitline). An emulated S_1 cell needs to know the cell type of the other activated cell, while an emulated S_2 cell needs to know both the cell type and the cell contents (previously stored data) of the other activated cell. Since any two cells in the same column may be activated at the same time, there must be a path between a skew cell and all the other cells in the same column to communicate the cell type (for both S_1 and S_2 cells) and the cell contents (for S_2 cells). That indicates a large amount of connections between cells. Directly connecting all the cells would be impossible, since it quickly uses up the wiring resources of FPGAs. A crossbar-like implementation [19] based on multiplexers and demultiplexers may solve the wiring resource problem, if the array size is not too large. However, that causes more delays as many stages of multiplexing are required for a large MWA skew SRAM array. Although emulating a single cell with FPGA is possible, another issue arises when emulating a complete MWA skew SRAM array. As the logic behavior of MWA skew cells are much more complicated than the original skew cells described in [12, 13], the number of configurable cells (look-up-tables or LUTs, and Flip-flops) required for emulating their behavior increases a lot. To verify that, we implemented a 2 Kbit (with 32 bit word-width) MWA skew SRAM array with a Xilinx Virtex-6 FPGA (fabricated with a 40-nm technology). Even in such a highly simplified case with only 16.7 % S_1 and 16.7 % S_2 cells randomly distributed in the array, already 84 % LUT resources of the FPGA were used. Implementing a 512 Kbit (an optimal array size suggested in [13]) MWA skew SRAM array with commercially available FPGAs would be simply impossible. Even if an emulator can be built by connecting hundreds of FPGAs, the communication delay between FPGA chips would further widen the speed gap between

the emulator and the "legal" monolithic SIMPL system. As the maximum operating frequency for MWA write operations achieved with the Xilinx Virtex-6 FPGA implementing a 2 Kbit array is only 21 MHz, implementing a full-size MWA skew SRAM array with hundreds of FPGA chips would end up with an even much slower emulator. By increasing the number of wordlines n that may be activated at the same time, the connections between skew cells and other cells in the same column will also at least linearly increase when implementing the emulator in the cell-by-cell way described above. And since the complexity of skew cell logic behavior also increases with n, emulating the MWA skew SRAM array will become even harder and slower as n increases.

Digital Clone. Building a digital clone using standard computing systems like PC and software could be the most cost-effective way to emulate the behavior of an MWA skew SRAM. However, due to the data dependency of a computing cycle on its previous cycles, parallelization of the computation taking place in the MWA skew SRAM is impossible. As discussed in [13], building a digital clone wouldn't produce a faster system compared to a functional physical clone. Even if the delay for computing logic (for calculating the new data to be stored) is ignored, just reading the data required for computing logic and writing back the new data already makes the emulator (implemented with an Intel Core 2 Quad CPU working at 2.5 GHz and a 3 GB RAM) at least 25 times slower [13]. By applying MWA designs, this speed gap will be further enlarged by at least a factor of n (where n is the number of simultaneously activated wordlines) in write operations, as the data bit-width of PCs is just comparable with that of skew SRAMs, and any combination of wordlines to be activated in a cycle may happen, today's memory architectures in standard computing systems are not able to parallelize or speed up the required cycle-by-cycle read out and write back procedure as described in building "functional physical clone(s)".

5.2 Conclusion and Future Scope

This paper presented a multiple wordline activation (MWA) skew SRAM design to improve the security of the original skew SRAM based SIMPL systems discussed in [12, 13]. By enabling parallel computations taking place in different cells controlled by multiple wordlines using the MWA skew SRAM design, the security of SIMPL systems based on that is enhanced by a factor linearly related to the number of simultaneously activated wordlines. Simulations of the enhanced skew SRAM show expected behavior complexity and satisfying yield considering manufacturing process variations. This makes our approach one of the first practically viable, circuit-based implementations of SIMPL system and public PUFs.

The security level can be further enhanced by increasing the number of supply voltages and/or the number of simultaneously activated wordlines. However, as the number of supply voltages and/or the number of simultaneously activated wordlines increase, the stability of the skew behavior against ambient noises, temperature changes, supply voltage ripples as well as process variations may decrease. A trade-off between security level and stability needs to be determined in future work.

References

1. Rührmair, U.: SIMPL systems: on a public key variant of physical unclonable functions. In: IACR Cryptology ePrint Archive, No. 2009/255 (2009)
2. Rührmair, U., Chen, Q., Stutzmann, M., Lugli, P., Schlichtmann, U., Csaba, G.: Towards electrical, integrated implementations of SIMPL systems. In: Samarati, P., Tunstall, M., Posegga, J., Markantonakis, K., Sauveron, D. (eds.) WISTP 2010. LNCS, vol. 6033, pp. 277–292. Springer, Heidelberg (2010)
3. Suh, E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: Proceedings of the 44th annual Design Automation Conference, DAC 2007, pp. 9−14. ACM Press (2007)
4. Chen, Q., Csaba, G., Lugli, P., Schlichtmann, U., Rührmair, U.: The bistable ring PUF: a new architecture for strong physical unclonable functions. In: Proceedings of the 2011 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2011, pp. 134−141. IEEE Press (2011)
5. Kumar, S.S., Guajardo, J., Maes, R., Schrijen, G.J., Tuyls, P.: The butterfly PUF protecting IP on every FPGA. In: Proceedings of the 2008 IEEE International Workshop on Hardware-Oriented Security and Trust, HOST2008, pp. 67–70. IEEE Press (2008)
6. Maes, R., Tuyls, P., Verbauwhede, I.: Intrinsic PUFs from flip-flops on reconfigurable devices. In: 3rd Benelux workshop on information and system security, WISSec 2008, **17** (2008)
7. Chen, Q., Csaba, G., Lugli, P., Schlichtmann, U., Rührmair, U.: Characterization of the bistable ring PUF. In: Proceedings of the 2012 Design, Automation & Test in Europe Conference & Exhibition, DATE 2012, pp. 1459–1462. IEEE Press (2012)
8. Beckmann, N., Potkonjak, M.: Hardware-based public-key cryptography with public physically unclonable functions. In: Katzenbeisser, S., Sadeghi, A.-R. (eds.) IH 2009. LNCS, vol. 5806, pp. 206–220. Springer, Heidelberg (2009)
9. Rührmair, U., Sölter, J., Sehnke, F.: On the foundations of physical unclonable functions. In: IACR Cryptology ePrint Archive, No. 2009/277 (2009)
10. Potkonjak, M., Meguerdichian, S., Nahapetian, A., Wei, S.: Differential Public Physically Unclonable Functions: Architecture and Applications. In: Proceedings of the 48th annual Design Automation Conference, DAC 2011, pp. 242–247. ACM Press (2011)
11. Meguerdichian, S., Potkonjak, M.: Matched public PUF: ultra low energy security platform. In: Proceedings of International Symposium on Low Power Electronics and Design, ISLPED 2011, pp. 45−50 (2011)
12. Chen, Q., Csaba, G., Ju, X., Natarajan, S., Lugli, P., Stutzmann, M., Schlichtmann, U., Rührmair, U.: Analog circuits for physical cryptography. In: Proceedings of the 12th International Symposium on Integrated Circuits, ISIC 2009, pp. 121–124. IEEE Press (2009)
13. Chen, Q., Csaba, G., Lugli, P., Schlichtmann, U., Stutzmann, M., Rührmair, U.: Circuit-based approaches to SIMPL systems. J. Circuits Syst. Comput. **20**(01), 107–123 (2011)
14. Predictive technology model, Nanoscale Integration and Modeling (NIMO) Group, ASU. http://ptm.asu.edu/
15. Trimberger, S.: Field-Programmable Gate Array Technology, p. 100. Springer, New York (1994)
16. Anderson, R.: Security Engineering. John Wiley, New York (2008)
17. Helfmeier, C., Boit, C., Nedospasov, D., Seifert, J.: Cloning physically unclonable functions. In: Proceedings of the 2013 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2013, pp. 1–6. IEEE Press (2013)

18. Biddle, P., England, P., Peinado, M., Willman, B.: The darknet and the future of content protection. In: Feigenbaum, J. (ed.) DRM 2002. LNCS, vol. 2696, pp. 155–176. Springer, Heidelberg (2003)
19. Sonntag, S., Reinig, H., Linz, S., Pitter, F., Ruhwandl, M.; XB07: a highly reusable crossbar architecture for multiprocessor system on chip (MPSoC). In: IP Based Electronic System Conference, IP 2007, pp. 307−311 (2007)