Journal of Circuits, Systems, and Computers
Vol. 20, No. 1 (2011) 107–123
© World Scientific Publishing Company
DOI: 10.1142/S0218126611007098

CIRCUIT-BASED APPROACHES TO SIMPL SYSTEMS*

QINGQING CHEN[†,‡,§], GYÖRGY CSABA[‡,¶], PAOLO LUGLI[‡,‖]
and ULF SCHLICHTMANN[†,**]

[†]*Institute for Electronic Design Automation,*
Technische Universität München,
Arcisstr 21, D-80333 Munich, Germany

[‡]*Institute for Nanoelectronics, Technische Universität München,*
Arcisstr 21, D-80333 Munich, Germany
[§]*qingqing.chen@tum.de*
[¶]*csaba@tum.de*
[‖]*lugli@tum.de*
[**]*ulf.schlichtmann@tum.de*

MARTIN STUTZMANN

Walter Schottky Institut, Technische Universität München,
Am Coulombwall 3, D-85748 Garching, Germany
stutz@wsi.tum.de

ULRICH RÜHRMAIR

Institute for Computer Science VI,
Technische Universität München,
Boltzmannstr 3, D-85748 Garching, Germany
ruehrmai@in.tum.de

This paper presents circuit-based approaches to SIMPL Systems (SIMulation Possible, but Laborious Systems), which could be regarded as a "public-key" version of Physical Unclonable Functions. The use of these systems can help us to avoid some of the potential vulnerabilities of conventional cryptography, such as its dependency on secret binary keys. Two specially designed circuits for SIMPL systems are discussed: "skew" memories and massively parallel analog processor arrays known as Cellular Nonlinear Networks. We argue that these circuits are able to serve as SIMPL systems in practice, and discuss their security against numerical and physical attacks.

Keywords: Green design; physical cryptography; public key cryptography; physical unclonable function; SIMPL systems; skew memory; cellular nonlinear network.

*This work was supported by the International Graduate School of Science and Engineering (IGSSE) and the Institute for Advanced Study (IAS) of the Technische Universität München.

1. Introduction

According to the “Green Electronics Survey”,[1] electronic companies should try to prolong the actual lifecycle of their products by extending the design life and encouraging consumers to use the products for longer time spans. However, in the field of electronic security industry, it has been happening so often that a security-related product has to end its lifespan much earlier than its expected obsolescence because of early breakings or the discovery of unexpected security problems. To mitigate the effect that these no-more-secure electronic wastes could bring to the environment, one may, as suggested in Ref. 1, upgrade the product by modifying the software, or by substituting some relevant hardware components. However, these options are not always available or practical. In most cases, insecure products have to be called back and disposed, or must be directly thrown away by consumers. New ones, in which the security problems have been fixed, are distributed subsequently. However, new problems may float to the surface some time later as long as they are still based on the same conventional algorithmic cryptographic primitives and do not solve their vulnerabilities to physical attacks, e.g., side channel attacks. A versatile and more active solution to make the products “greener” would be to eliminate the vulnerabilities to these attacks from the beginning. To explain this, we need to start with some discussions about conventional cryptography.

Conventional cryptographic algorithms and protocols have originally been designed to defend against brute force and numerical cryptanalytic attacks. Although they often rely on unproven mathematical assumptions, the cryptographic primitives are generally working fine up to today, provided that the employed secret keys are unknown to the attackers. This requires that the keys must be stored and safeguarded well, which is by no means an easy task in mobile and often networked hardware systems. Beyond purely numerical attacks, fraudsters can apply physical attack strategies, including invasive and side channel attacks. Full or partial knowledge of the key can be obtained by observing, for example, the timing,[2] power consumption[3,4] or electromagnetic leaks[5] of the circuits, and then analyzing such side channel information. Should the secret key become known, the security of all protocols built on it is ruined: any information that was encrypted with it is compromised, and the key can be replicated arbitrarily to mount future impersonation attacks in identification protocols.

To tackle some of the known shortcomings of conventional cryptography, physical cryptography has been proposed, which takes into consideration the security of the physical implementation of cryptographic systems and physical attacks. One concept that has recently been studied in detail are “Physical Unclonable Functions (PUFs)”.[6–12] A PUF is a physical function that maps challenges to responses based on complex physical phenomena taking place in the PUF structure. It should be easy to evaluate a PUF, i.e., to determine its response to a single challenge, but it should also be hard to completely characterize a PUF, i.e., to read out its responses to all possible challenges within short time.[6] Furthermore, it should be hard to predict the responses of the PUF numerically, i.e., without possessing physical access to it. With these properties, PUFs are able to serve in various protocols, for example, in challenge-response based authentication.[7] However, these protocols still require a previously shared piece of information that must be kept secret throughout the protocol, which makes PUFs a secret-key like primitive. For applications where public-key type protocols are required or practically advantageous, another PUF-like primitive that is also naturally immune to cryptanalysis and physical attacks has been proposed in Refs. 13 and 14. This new concept can be regarded as "public-key" PUF. It requires a physical object to respond to a challenge much faster than any feasible software or hardware emulation would do. However, it still must be possible to (slowly) emulate the behavior of the system based on its publicly available simulation parameters. Based on the speed asymmetry between the physical object and the emulator, various public-key like protocols described in Ref. 13 can be executed with this novel type of PUF. This special sort of PUF was termed a SIMPL (SIMulation Possible, but Laborious) system. This paper presents two circuit-based implementations for SIMPL systems.

Section 2 of this paper introduces a first implementation[14–16] of SIMPL systems, which is based on a specially designed memory. We intentionally design some memory cells ("skew cells") such that they will deterministically fail at some write operations under specific supply voltages. Carrying out a lot of successive write and read operations on this special memory can be regarded as a special computation, which can only be realized with logic operations on a standard computing system.

Section 3 introduces another implementation[14–17] based on Cellular Nonlinear Networks (CNNs). This fully analog-circuit based approach builds its speed advantage on its real-valued and massively-parallel computing power that can outperform any numerical emulator when performing the same computing tasks.

2. Skew-Memory Based SIMPL System

2.1. *Structure and function*

Figure 1 shows the sketch of the skew-memory based SIMPL system.

The SIMPL System $\boldsymbol{S}$ consists of four main blocks, namely a skew memory (MEM), a challenge control block (CC), a voltage control block (VC), and a feedback and output control block (FOC).

After an initialization process that attempts to fill the memory with, e.g., all "0"s, the system receives the challenge C_i, carries out a number of successive write and read operations on the memory, while possibly changing the supply voltage VDD of the skew memory for each operation. Even if millions of write and read operations are performed, the response R_i takes less than tens of milliseconds to produce.

The memory MEM is called a skew memory because of its skew behavior. While a normal memory faithfully stores the information written into it, a skew memory may

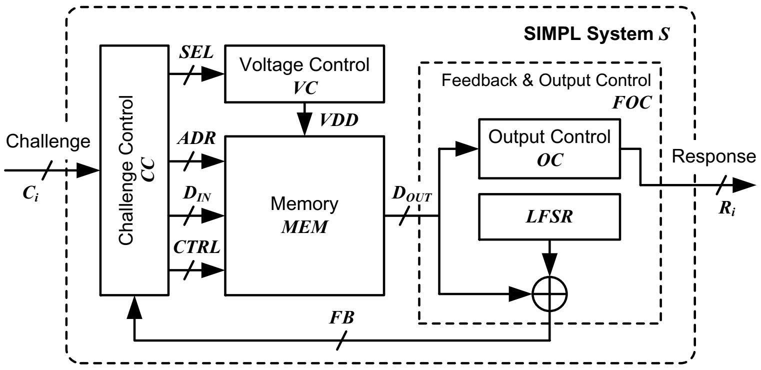

Fig. 1. Schematic illustration of the skew-memory based SIMPL system.

behave differently. It may store a fixed "0" or "1" value regardless of the data that have been attempted to be written into it (even the memory initialization process cannot modify the fixed cells). It may show a supply-voltage dependent behavior, which means that the success of write operations depends on the supply voltage VDD controlled by the voltage control block. For example, only when the supply voltage is $VDD \geq VDD_{\text{funcmin}} > VDD_{\text{min}}$ will the data be successfully stored in the cell. For $VDD < VDD_{\text{funcmin}}$, the cell's current content will not be changed. Different types of memory cells, i.e., fixed "0" cells, fixed "1" cells and supply-voltage dependent cells (different cells may have different VDD_{funcmin} values) are disorderedly distributed on memory arrays. It should be noted that only write operations are skewed. All read operations should correctly output the content of the addressed cells.

The challenge control block CC "scrambles" the input challenge and produces voltage select, address, write data and possibly memory control (e.g., "W/R", write or read) signals for VC and MEM. After the first cycle, the output of CC will be dependent on the feedback FB from the previous cycle. The CC unit could implement a hash function.

The voltage control block VC receives voltage select signals from CC and switches the supply voltage of MEM to the next scheduled value.

Inside the feedback and output control block FOC, the linear feedback shift register (LFSR) serves as a pseudorandom number generator, whose output is XOR'd with the readout data D_{OUT}, and the result is fed back to CC to modify the ADR, D_{IN}, SEL and possibly the $CTRL$ signals for the next operation on MEM. The output control OC generates the response R_i, which is a function of the data sequence read out from MEM.

2.2. *Public counterpart and usage of the system*

As a public counterpart of the skew-memory based SIMPL system, the description $D(\boldsymbol{S})$ of the system should contain all the information that is necessary for building a software emulator of the system. The information should contain the characteristics and the arrangement of the memory cells, the (logic) functionality of the CC, VC and OC units, as well as the seed value of the LFSR — based on these data the circuit behavior is possible to be emulated and the correctness of the behavior of $\boldsymbol{S}$ can be verified in a security protocol.[13] If the operation of the SIMPL system $\boldsymbol{S}$ can yield the response R_i faster than a previously specified time limit $t_{\max}$ (it must be ensured that emulations of the SIMPL system cannot determine R_i in a time $t < t_{\max}$), it proves that the data was provided by the actual SIMPL system $\boldsymbol{S}$, not by an emulator, e.g., digital clone. The security of the SIMPL system is assured by its clear speed advantage over any feasible clone.

2.3. *Design of skew-memory based SIMPL systems*

The detailed design (that is, the number of different supply voltages, the memory size and the data bit-width of the memory) of a skew-memory based SIMPL system is restricted by technological limits and security requirements. Proper choices of design parameters play an important role in the realizability and the security of the system.

As an example, this subsection describes some details of our design based on a TSMC 0.18 micron CMOS technology and discusses decisions regarding design parameters of the system.

The design of the skew memory gives the SIMPL system its speed advantage and uniqueness property. This is also the most unusual component of our design, so we started from the transistor level design of a skew memory cell.

2.3.1. *Skew memory cell*

Since Static Random Access Memories (SRAMs) are the fastest of all commercial semiconductor memories, our design is based on SRAM structures.

A six-transistor SRAM cell is schematically illustrated in Fig. 2, with sizes (width/length in micron) specified beside each transistor. This special sizing assures

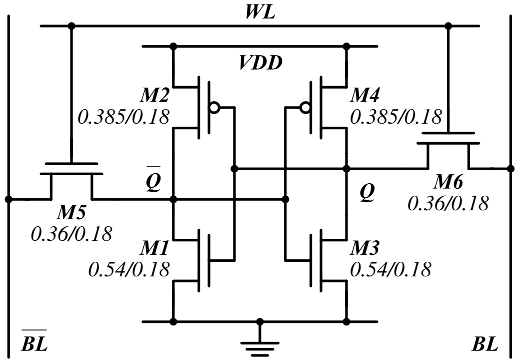

Fig. 2. Skew SRAM cell in a 0.18-μm CMOS technology.

a supply-voltage dependent but fully predictable and stable operation as long as proper supply voltages are applied.

We design the circuit to use two different supply voltages VDD_{high} and VDD_{low}. The sizing of the skew cell guarantees successful read operations under both VDD_{high} and VDD_{low}, and correct write operations under VDD_{high}. However, write operations under VDD_{low} are consistently unsuccessful.

In this design, the nominal minimum functional supply voltage (for write operations) VDD_{funcmin} is between 1.65 V and 1.66 V. Theoretically, any supply voltage that is higher than this VDD_{funcmin} will guarantee successful write operations and vice versa. We choose $VDD_{\text{high}} = 1.8\,\text{V}$ and $VDD_{\text{low}} = 1.3\,\text{V}$, which results in reasonable yield, considering unavoidable manufacturing variations in the transistor parameters. Figure 3 shows a sequence of write and read operations by switching VDD between 1.8 V and 1.3 V with the clock frequency of 900 MHz. The write operations which are attempted around $t = 3$ ns and $t = 7$ ns are unsuccessful since VDD equals 1.3 V.

The transistor width/length parameters shown in Fig. 2 are one possible sizing that can produce the special behavior required. By carefully tuning the transistor sizes, we obtained more skew cell designs with different minimum functional supply voltages.

2.3.2. *Skew memory block and individualization*

By randomly allocating skew memory cells that possibly have different minimum functional voltages in an array, we obtain the main part of the skew memory block.

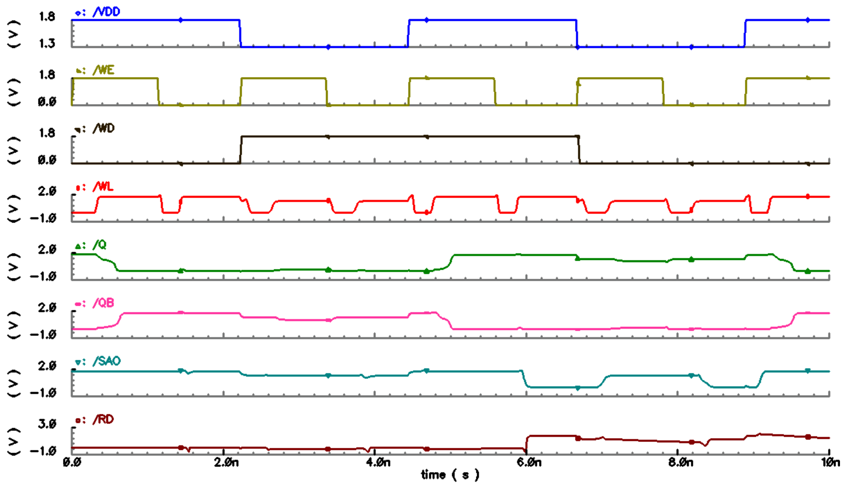

Fig. 3. Operations of a skew SRAM cell with clock frequency of 900 MHz (WE, write enable, H/L = write/read; WD, write data; WL, wordline; Q/QB, cell content; SAO, sense amplifier output; RD, readout data).

Like a normal SRAM, the memory block consists of memory arrays and peripheral logic such as address decoders, sense amplifiers and others.

In practice, besides the skew cell shown in Fig. 2, we include in the memory array another cell design with $VDD_{\text{funcmin}} \leq VDD_{\text{min}}$, which is actually a normal SRAM cell (both write and read operations are always successful since the whole supply voltage span is above the minimum functional supply voltage). The normal cells and the skew cells are randomly distributed in the memory array. Clearly, using more levels of supply voltages and more different skew cells would increase the complexity of the system. However, this would also increase the cost and decrease the reliability of operations against noise and process variations (because safety margins between different supply voltages are decreased), resulting in lower yield. Applying two distinct supply voltages and using two kinds of memory cells as we did results in acceptable reliability and does not lose the security brought by the skew behavior.

To choose a proper size and a proper data bit-width of the memory, we need to think about the security of the system. The address bit-width and the data bit-width of the memory should be sufficiently large so that pre-computing a look-up table (LUT) for the whole system is infeasible. Let us assume that the challenge control implements a bijective function, which simplifies our estimation of the minimum LUT required for the emulation of the system. We assume that the challenge is received only in the first cycle, since it is the simplest and the most practical design in reality (the rest of the paper is based on this assumption, unless specified); the bit-width of the challenge is w_C; the bit-width of the memory data (i.e., D_{IN} and D_{OUT}) is w_D; the final response R_i is the D_{OUT} of the last read operation; and the number of cycles to output R_i has been fixed. Theoretically, the minimum size of the LUT required for the system would be $w_D \times 2^{w_C}$. For a current process technology, 64 could be a suitable value for w_D. A memory of 512 Kbit size with $w_D = 64$ requires an address bit-width of 13. Even if we do not consider the voltage select and memory control signals, this would already require an LUT of $64 \times 2^{64+13}$ bit $= 2^{37}$ TByte, which is clearly infeasible. If the number of cycles to output the response R_i becomes a variable n_{cycle}, which is transferred with the challenge, an even larger LUT will be required to carry out LUT attacks. For a different n_{cycle} value, a different LUT with the size of $w_D \times 2^{w_C}$ will be required, since an attacker cannot compute D_{OUT} of later cycles out of the D_{OUT} of the current cycle unless he knows the current content of the memory also. If the protocol allows the n_{cycle} variable to be in the integer interval $[n_{\text{cyclemin}} .. n_{\text{cyclemax}}] = \{n_{\text{cycle}} \in \mathbb{Z} | n_{\text{cyclemin}} \leq n_{\text{cycle}} \leq n_{\text{cyclemax}}\}$, the required LUT size for storing all the possible challenge-response pairs would be $w_D \times 2^{w_C} \times (n_{\text{cyclemax}} - n_{\text{cyclemin}} + 1)$.

The individualization of the circuits will finally enable them to serve in the cryptographic protocols described in Ref. 13. This can be done by adding fixed "0" and fixed "1" cells in the last fabrication steps using, e.g., laser fuses.[18,19]

Depending on where the fixed "0" and fixed "1" cells are located, a practically infinite number of distinct memories can be fabricated out of one design. For example, assuming a 32×32 bit size array, which is still much smaller than our 512 Kbit array, the number of distinct designs is already larger than 10^{488}. Even if we fix the number of fixed "0" and fixed "1" cells to, e.g., 100 (this would make it impossible to modify a skew memory to behave like another legal one using, e.g., an FIB (Focused Ion Beam)), the number of distinct designs is still larger than 10^{277}. However, embedding the cell arrangement information in the description $D(\boldsymbol{S})$ of the system would require a sequence of 1 Mbit for a 512 Kbit memory array (two bits are required for a single cell, which has four possible types), which seems to be not quite economical to identify a system. To reduce the length of $D(\boldsymbol{S})$, the arrangement could be made according to the output of a pseudorandom number generator (PRNG). By doing this, the cell arrangement can be described with just a short seed value and the algorithm of the PRNG.

2.3.3. *Peripheral blocks* (CC, VC *and* FOC)

The CC block should have three main properties:

(1) It should be able to calculate the output, i.e., address, data input and voltage control signals for the next cycle so quickly that it will not become a main speed bottleneck of the system. If CC is slower than the memory, it can compromise the speed advantage gained by the skew memory;
(2) A small change, e.g., a one-bit change on the input should produce a great impact on the output, and any bit of the output can possibly be changed. This requirement ensures that the feedback from FOC can have great impact on the next operation on MEM, and therefore ensures high dependency of later operations on previous ones;
(3) It should map the expected inputs as evenly as possible over its output range, so that each memory cell can have an equal opportunity to be accessed, and the write data D_{IN} will not be biased, etc.

Under these requirements, a carefully designed hash function would be a suitable candidate for the implementation of CC.

The VC block should implement a fast voltage switch circuit, so that it will not hinder the system speed.

Inside the FOC block, OC could simply output the D_{OUT} of the last or the last several read operations or implement a function (e.g., a hash function) of the last or the last several readout data. The LFSR is used to lower the probability that a sequence (possibly very long) of operations forms cyclic behavior or generates cyclic outputs. It could implement a 64-bit Fibonacci LFSR with an XOR-based feedback, and the seed value can be an arbitrary binary sequence except all-zeroes.[a] This seed value does not need to be unique for different circuits, since the individuality does not

[a] The all-zeroes state is considered illegal for XOR-based Fibonacci LFSRs since the LFSR would remain "locked-up" in this state.

(although it can) come from the seed value, and the security does not rely on the seed value either. With a 64-bit Fibonacci LFSR, which has a period of $2^{64} - 1$, the probability of cyclic behavior is practically removed. Besides, LFSR also generates FB for write operations, in which no D_{OUT} is produced.

Based on the above arguments, we choose two different supply voltages, two kinds of memory cells with different minimum functional supply voltages (of which, one is between VDD_{high} and VDD_{low}, and the other is lower than VDD_{low}) and a 512 Kbit memory array with the data bit-width of 64 bits as an optimal design for the skew memory. For this design, the address space for the memory is 2^{13}, which makes a number around 100,000 a suitable number of cycles to output D_{OUT}. If the number of cycles n_{cycle} is too small, e.g., less than 5000 in this case, it will be very likely that many of the write operations do not really affect the final D_{OUT}, since write operations do not produce output for the feedback loop. If the addressed cells in write operations are not read out at some later time point, the write operations are essentially irrelevant. Since the LFSR and CC design as we suggested guarantees that the addresses for memory operations are uniformly distributed, $n_{\mathrm{cycle}} = 100{,}000$ ensures that almost all the cells of our 512 Kbit skew memory that have been addressed in write operations are also read out at least once (over 12 operations per address on average).

2.4. *Security assessment*

Although some discussion about the security of the skew-memory based SIMPL system has been carried out in previous subsections, this subsection will systematically analyze the security of the system from the perspective of faking the system.

There are three basic possibilities for a faker to pose as the owner of the SIMPL system without actually being in possession of it. The construction of the circuit should make all these possibilities infeasible.

2.4.1. *Building an exact physical clone*

Based on the design described above, the skew-memory based SIMPL system can still be re-fabricated in a silicon foundry. This is extremely expensive having one-time costs in the millions of US dollars range,[20] therefore the system can be considered secure against attacks by consumers and individual hackers.

2.4.2. *Building a functional physical clone*

One concept to build a functional physical clone of the skew memory is to combine "normal" (mass-manufactured) memories together with simple logic circuits to mimic the operations on skew cells. Figure 4 shows the basic construction of this attack. In this construction, the configuration (that is, the type of cell: either fixed "0", fixed "1", skew cell, or normal cell) of the memory cells are stored in a Configuration Memory. In our case, the Configuration Memory has twice the size of the skew

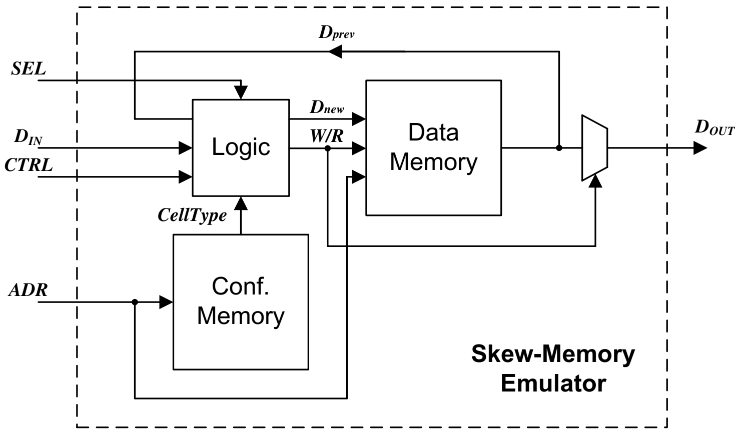

Fig. 4. A skew-memory emulator construction based on normal SRAMs and logic.

memory, since each cell has four possible types, and the content of the cells are stored in another normal memory called the Data Memory. Physically these two memories could be a single memory with at least three times the size of the skew memory they attempt to emulate. Read operations of a skew memory can directly be emulated on the data memory, since the read behavior is not "skewed". To emulate a write operation of the skew memory, the cell configuration *CellType* and the cell content D_{prev} need to be read out first. The new data D_{new} (could be the same as the previous data) will be evaluated through the logic block, and will then be written back to the data memory. Since a write operation needs to be substituted by a full read and write operation pair, this construction will be at least a factor of two slower for write operations if the configuration and the data memories have the same speed as our skew memory and if the delay of the logic operations can be neglected. However, the fastest mass- produced commercial SRAM by Cypress and Samsung Semiconductor in early 2010 has only a maximum operating frequency of 550 MHz while custom-designed SRAMs, e.g., L1 caches for Intel processors (of about 512 Kbit size, which is the same as our skew memory in size) have reached the operating frequency of 3 GHz. Since the current operation of the skew memory is always dependent on the output of the previous operation and successive operations have quite low probability of accessing successive addresses, the advanced features of today's SRAMs, e.g., pipelined and burst operations cannot be applied. Both the emulator and our skew memory will suffer from the initial latency (which is usually 2−3 clock cycles) in each operation. The actual speed of the emulator will be lower than 250 MHz, and the skew memory can operate at the speed of at least 1 GHz. Considering the speed advantages gained from the structure and the memory speed, the functional physical clone described above will lag behind by at least a factor of eight. Using FPGAs for the above structure will end up with a similar result because of the same reasons.

Another option that uses FPGAs is to replicate the behavior of the skew memory cell by cell. The emulation of a single skew cell requires on average at least three configurable cells (each including a LUT and a Flipflop) of an FPGA. This requires a very large (and very expensive) FPGA even for moderately sized (512 Kbit) skew memories. However, FPGAs with this size still do not exist in early 2010 (FPGAs by Xilinx provide at most about 800,000 configurable cells) — using multiple chips would further widen the speed gap between the emulator and the "legal" monolithic circuits. Even if we assume such huge FPGAs exist, the emulation on FPGAs will still be much slower than a custom-designed skew memory. To prove this, we emulated a small (32 Kbit size, 32 bit word-width) skew memory with a 10-to-1024 address decoder on an FPGA fabricated with a 65 nm technology (even in such a highly simplified case, we used up the resources of the FPGA). The maximum operating frequency the design can achieve is only about 200 MHz. A full size emulator will be even slower. Therefore, we estimate that the operating speed of such an emulator on a state-of-the-art FPGA will always lag behind optimized special purpose SRAMs by at least a factor of 10.

2.4.3. *Digital clone*

Parallel computing of the response of the SIMPL system $\boldsymbol{S}$ on a multi-core computing system is the most straightforward possibility to close the speed gap between $\boldsymbol{S}$ and its emulator.

However, the data dependency of later operations on the output or effect (on the memory content) of earlier operations gives no chance of parallelization. The only trick that a faker can try is that the write operations, of which the addressed cells are never read out in later operations, can be ignored. Except this, all other operations have to be carried out sequentially (although the order can be modified to some degree — a write operation on an address can be delayed until this address is about to be read out), since the read address must be calculated from the output of the previous read operation, and the new content of the memory must be evaluated (for write operations) if the same address is going to be accessed by a later read operation. This implies that the system should carry out fewer "meaningless" write operations. However, the solution should not be to significantly decrease the number of write operations, since write operations are the main source of the skew system's speed advantage. Two possibilities exist: a direct solution is to carry out a lot of successive read operations in the end to make all or almost all previous write operations "meaningful" to the final output. The disadvantage is that this would require a lot of extra operations (about 10% more, assuming that the address bit-width equals to 13 and totally 100,000 operations are to be carried out). Another possibility is to make a large proportion (but not all, otherwise the first attack described in Sec. 2.4.2 can be accelerated by leaving out the read operations since the D_{OUT} would be already available in the previous write operation which equals to D_{new}) of the write operations each followed by a read operation on the same address as the write operation. Both proposals can be realized by changing the CC block a bit (fix the "W/R" control signal to "read enabled" for the last, e.g., 2^{13} cycles, or use one or several bits of the FB input to indicate whether the address should be updated).

Another possibility for a digital clone is to pre-compute and build LUTs, so that multiple or all cycles of $\boldsymbol{S}$ can be computed with a single emulation step. However, as is discussed in Sec. 2.3.2, the storage needed by LUTs grows exponentially with the bit-width of the input (challenge C_i or $ADR|D_{\mathrm{IN}}|SEL$). Building LUTs for interim cycles will require even larger storage, because the memory content will also (partly) become part of the input, while for the first cycle the memory content is a constant. Besides, larger LUTs also imply longer search time. Therefore pre-computing LUTs for all the operations or a part of the operations are both infeasible because of the impractical storage requirement and low speed of searching in a huge memory space.

To estimate the speed advantage of a skew memory over a digital clone, we implemented a digital clone of a 512 Kbit skew memory with the data bit-width of 32. On a system with an Intel Core (TM) 2 Quad CPU working at 2.5 GHz and a 3 GB RAM, the emulation of skew memory operations takes at least 25 ns (CPU time) per operation, which is at least 25 times slower than a custom-designed skew memory.

In summary, armed with the skew behavior and by carefully choosing the design parameters, the skew-memory based design fulfills the definition of SIMPL systems, and will be able to serve in the protocols proposed in Ref. 13.

3. Cellular-Nonlinear-Network Based SIMPL System

The speed advantage of the above discussed implementation over possible attackers comes from the special computing ability of skew memory cells, which is amplified through a feedback. The Cellular-Nonlinear-Network (CNN) based implementation builds its advantage on the real-valued and massively parallel computing power that can outperform any numerical emulator when performing the same special tasks.

3.1. *Cellular nonlinear networks*

CNNs are analog computing arrays with regularly arranged and locally interconnected elementary processing cells.[21] Each cell is characterized by a state variable, and its time evolution is described by an ordinary differential equation (ODE). The time evolution of the state variable depends on the cell's own internal state and on the outputs of neighboring cells. On an abstraction level, the cell behavior and the interactions between neighboring cells is characterized by two templates, which are real-valued matrices.[21] Figure 5 shows the structure of a CNN, the general form of the ODE describing its time evolution and an example of the output function.

CNNs can be simulated by solving the time-dependent differential equations describing the time evolution of each cell (e.g., a thousand-cell CNN solves thousand coupled nonlinear ODEs "by hardware").

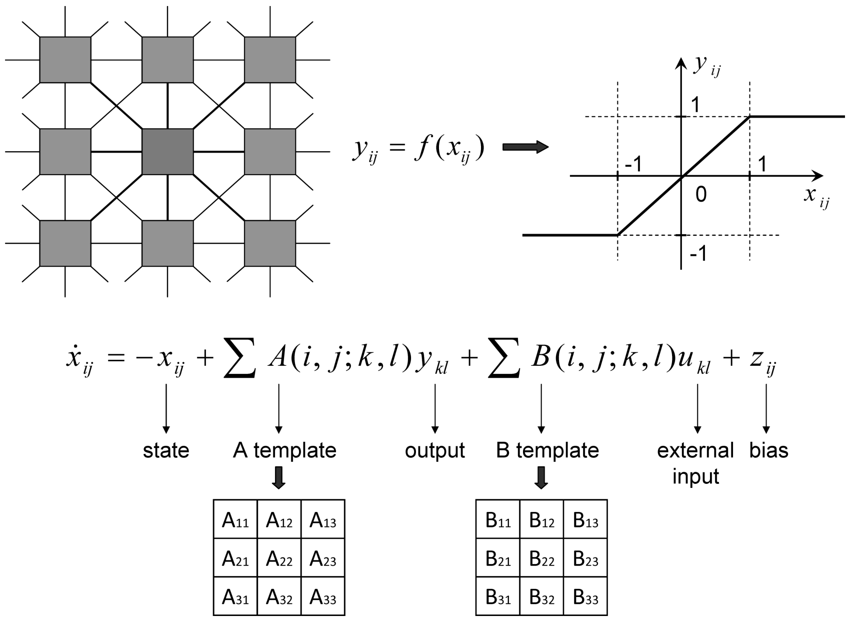
$y_{ij} = f(x_{ij})$
y_{ij}
1
-1
1
0
x_{ij}
-1
$\dot{x}_{ij} = -x_{ij} + \sum A(i,j;k,l)y_{kl} + \sum B(i,j;k,l)u_{kl} + z_{ij}$
state
A template
output
B template
external input
bias
A_{11} A_{12} A_{13}
A_{21} A_{22} A_{23}
A_{31} A_{32} A_{33}
B_{11} B_{12} B_{13}
B_{21} B_{22} B_{23}
B_{31} B_{32} B_{33}

Fig. 5. Schematic layout of a CNN circuit and the equations describing its operation (bottom: time evolution ODE; right upper: output function).

Due to their analog and highly parallel architecture, CNNs have a remarkable computing power and efficiency. Already in 2004, a state-of-the-art programmable, commercially available CNN in a $0.35\,\mu$m standard CMOS technology exhibited peak computing figures of 330 GOPS (3.6 GOPS/mm^2 and 82.5 GOPS/W, projected for chip area and power consumption).[22] In specialized tasks, it is known that CNNs can outperform digital computers by a factor of up to 1000.[21,23] Due to this extreme computing performance, CNN circuits can be promising candidates for SIMPL systems, as long as their implementations are individualizable, characterizable and reliable in operation.

3.2. *Implementation*

Figure 6 shows a basic circuit implementation of a CNN cell.[14,16,17] The resistance of the resistors in the circuit represents the templates the cell implements.

The individualization of CNN circuits can be achieved by setting unique templates through hard-programming the resistances. To do this, laser fuses that have been proposed for the individualization of skew memories can also be used here. Another approach to individualize CNN circuits is to take advantage of the inherent fabrication variations of, e.g., the resistors. This approach would enhance the security level of the system because it would arm the system with unclonability. However, since fabrication variations are usually much smaller than the mismatch

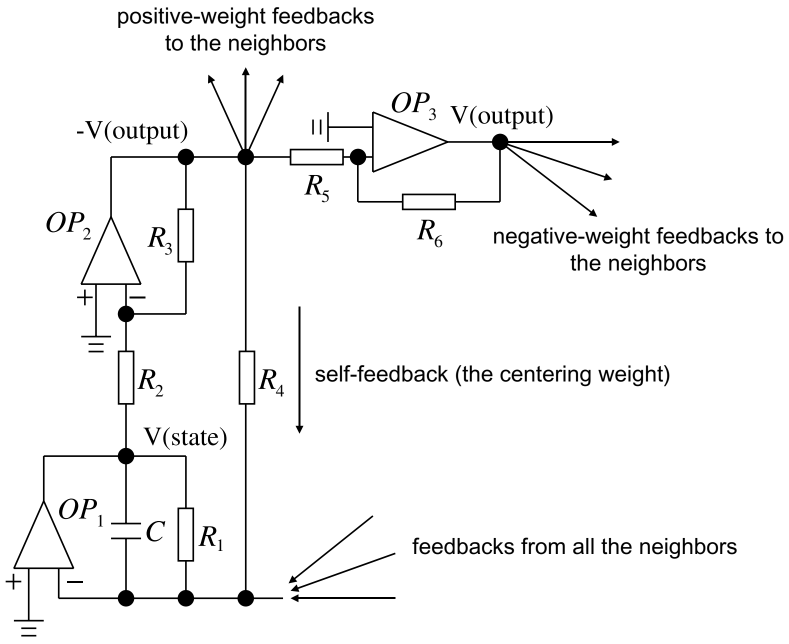
positive-weight feedbacks to the neighbors
-V(output)
OP_3
V(output)
R_5
R_6
OP_2
R_3
negative-weight feedbacks to the neighbors
R_2
R_4
self-feedback (the centering weight)
V(state)
OP_1
C
R_1
feedbacks from all the neighbors

Fig. 6. Circuit design of a CNN cell.

that hard-programming can produce, the sensitivity of the final output to operating noises would become relatively a bigger challenge. Besides, the characterization would become an extra step which may require, e.g., some built-in self-measuring circuit.

Reliability is a natural challenge for SIMPL systems using analog computing powers. Since CNNs are a general structure, a lot of properties including sensitivities to noises can generally be tuned through its output functions and time evolution equations. However, this may at the same time reduce its computing complexity and therefore security. A balance between complexity and reliability could be made to optimize the CNNs for SIMPL systems.

3.3. *Security assessment*

Dedicatedly designed CNN circuits for SIMPL systems have huge natural speed advantages over digital computers.

Field-Programmable Analog Arrays (FPAA) could be regarded as programmable CNNs in our case, which can be programmed to behave like CNNs. However, since FPAAs are built for flexible uses, their speed and array size are relatively limited because of the complexity of the designs. Custom-manufactured CNNs will outperform these programmable CNNs especially when very large arrays are required in the computing task.

CNN based approaches to SIMPL systems could possibly be made technologically secure by introducing fabrication variations for CNN individualization. However, this would require extra efforts to characterize each CNN during or after fabrication.

4. Discussion and Conclusion

This paper discussed the implementations of a new physical cryptographic primitive termed SIMPL systems, which had been proposed in Ref. 13. These systems can be considered a public-key version of Physical Unclonable Functions (PUFs). While PUF-based protocols[7] still require a previously shared piece of secret information (typically a set of challenge-response pairs) between the communicants, and therefore have some aspects of secret-key like protocols, a SIMPL system (as a secret "key"), together with its software counterpart (as the public "key") that can (slowly) simulate the functionality of the SIMPL system hardware, is able to serve in various public-key like cryptographic protocols introduced in Ref. 13. This new physical cryptographic primitive is "greener" than conventional cryptographic primitives with respect to the security against physical attacks, and may extend the actual lifespan of security systems based on it.

The implementations of SIMPL systems should be able to provide the speed advantages over any feasible emulation of the system. This paper discussed two concrete implementation candidates for SIMPL systems. The first skew-SRAM based implementation obtains its speed advantage over emulators from the "skew" behavior of specially designed SRAM cells, and this speed advantage is amplified with a feedback loop. We discussed the skew-SRAM based implementation in detail and assessed its security against different clone attacks. In the end we outlined another implementation based on CNNs, which uses the computing ability of analog cellular arrays to create speed advantages over numerical emulators.

Acknowledgments

The authors are grateful to Srinivas B. Natarajan, Xueming Ju, Bing Li and Ning Chen.

References

1. M. Rautner and C. Harrell, *Green Electronics Survey*, ed. S. Erwood (Greenpeace International, Amsterdam, 2008), pp. 11 and 14.
2. P. Kocher, Timing attacks on implementations of Diffie−Hellman, RSA, DSS, and other systems, *Proc. 16th Annual International Cryptology Conference on Advances in Cryptology*, ed. Koblitz (Springer, London, 1996), pp. 104−113.
3. P. Kocher, J. Jaffe and B. Jun, Differential power analysis, *Proc. 19th Annual International Cryptology Conference on Advances in Cryptology*, ed. M. Wiener (Springer, London, 1999), pp. 388−397.

4. M. Alioto, L. Giancane, G. Scotti and A. Trifiletti, Leakage power analysis attacks: A novel class of attacks to nanometer cryptographic circuits, *IEEE Trans. Circuits Syst. I: Regular Papers* **57** (2010) 355–367.
5. K. Gandolfi, C. Mourtel and F. Olivier, Electromagnetic analysis: Concrete results, *Proc. 3rd Int. Workshop on Cryptographic Hardware and Embedded Systems*, eds. Ç. K. Koç, D. Naccache and C. Paar (Springer, London, 2001), pp. 251–261.
6. B. Gassend, D. Clarke, M. van Dijk and S. Devadas, Silicon physical random functions, *Proc. 9th ACM Conf. Computer*, Washington, DC, USA (2002), pp. 148–160.
7. G. E. Suh and S. Devadas, Physical unclonable functions for device authentication and secret key generation, *Proc. 44th Annual Design Automation Conf.*, San Diego, CA, USA (2007), pp. 9–14.
8. D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk and S. Devadas, Extracting secret keys from integrated circuits, *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **13** (2005) 1200–1205.
9. J. Guajardo, S. S. Kumar, G. Schrijen and P. Tuyls, FPGA intrinsic PUFs and their use for IP protection, *Proc. 9th Int. Workshop on Cryptographic Hardware and Embedded Systems*, Springer, Berlin/Heidelberg, Vienna (2007), pp. 63–80.
10. S. S. Kumar, J. Guajardo, R. Maesyz, G. Schrijen and P. Tuyls, The butterfly PUF protecting IP on every FPGA, *IEEE Int. Workshop on Hardware-Oriented Security and Trust*, Anaheim, CA, USA (2008), pp. 67–70.
11. R. Maes, P. Tuyls and I. Verbauwhede, Intrinsic PUFs from flip-flops on reconfigurable devices, *Proc. Benelux Workshop Information and System Security*, Eindhoven (2008).
12. R. Helinski, D. Acharyya and J. Plusquellic, A physical unclonable function defined using power distribution system equivalent resistance variations, *Proc. 46th Annual Design Automation Conf.*, San Francisco, CA, USA (2009), pp. 676–681.
13. U. Ruhrmair, SIMPL systems: On a public key variant of physical unclonable functions, *Cryptology ePrint Archive* (International Association for Cryptologic Research, 2009).
14. U. Rührmair, Q. Chen, M. Stutzmann, P. Lugli, U. Schlichtmann and G. Csaba, Towards electrical, integrated implementations of SIMPL systems, *Cryptology ePrint Archive* (International Association for Cryptologic Research, 2009).
15. Q. Chen, G. Csaba, X. Ju, S. B. Natarajan, P. Lugli, M. Stutzmann, U. Schlichtmann and U. Rührmair, Analog circuits for physical cryptography, *Proc. 12th Int. Symp. Integrated Circuits*, Singapore (2009), pp. 121–124.
16. U. Rührmair, Q. Chen, M. Stutzmann, P. Lugli, U. Schlichtmann and G. Csaba, Towards electrical, integrated implementations of SIMPL systems, *Information Security Theory and Practices: Security and Privacy of Pervasive Systems and Smart Devices: Proc. 4th IFIP WG 11. 2 International Workshop*, eds. P. Samarati, M. Tunstall, J. Posegga, K. Markantonakis and D. Sauveron (Springer, Berlin/Heidelberg, 2010), pp. 277–292.
17. G. Csaba, X. Ju, Z. Ma, Q. Chen, W. Porod, J. Schmidhuber, U. Schlichtmann, P. Lugli and U. Rührmair, Application of mismatched cellular nonlinear networks for physical cryptography, *12th IEEE Int. Workshop on Cellular Nanoscale Networks and their Applications*, Berkeley, CA, USA (2010), pp. 1–6.
18. R. T. Smith, J. D. Chlipala, J. F. M. Bindels, R. G. Nelson, F. H. Fischer and T. F. Mantz, Laser programmable redundancy and yield improvement in a 64 K DRAM, *IEEE J. Solid-State Circuits* **16** (1981) 506–514.
19. T. Chih-Wu and H. Hsien, Laser fuse structure, US Patent 6404035 (2002).
20. A. B. Kahng, Bringing down NRE, *IEEE Des. Test Comput.* **20** (2003) 110–111.
21. L. O. Chua and T. Roska, *Cellular Neural Networks and Visual Computing: Foundations and Applications* (Cambridge University Press, Cambridge, UK, 2002).

22. A. Rodríguez-Vázquez, G. Linan-Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Galán, F. Jiménez-Garrido, R. Domínguez-Castro and S. Meana, ACE16k: The third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs, *IEEE Trans. Circuits Syst. I: Regular Papers* **51** (2004) 851–863.
23. T. Roska, Cellular wave computers for nano-tera-scale technology — beyond boolean, spatial-temporal logic in million processor devices, *Electron. Lett.* **43** (2007) 427–429.