Analog Circuits for Physical Cryptography

Q. Chen, G. Csaba, X. Ju, S. B. Natarajan, P. Lugli, M. Stutzmann, U. Schlichtmann, U. Rührmair
Institute for {Electronic Design Automation, Nanoelectronics, Computer Science VI}, Walter Schottky Institut
Technische Universität München
Munich, Germany
{qingqing.chen, csaba, xueming.ju, srinivasbn81, lugli, stutz, ulf.schlichtmann}@tum.de, ruehrmai@in.tum.de

***Abstract*—This paper presents circuit designs to be used as 'public-key' Physical Uncloneable Functions. We will introduce two specially designed circuits: 'skew' memories and massively parallel analog processor arrays (CNNs). We show that such circuits – which are possible to simulate but computationally outperform any feasible digital and hardware-assisted emulation – can be used to remotely and securely identify their owner and serve as a basis for various cryptographic protocols.**

***Index Terms*—physical cryptography, physical uncloneable function, public key cryptography, skew memory, cellular nonlinear network**

I. Introduction

Traditional cryptographic protocols often rely on unproven mathematical assumptions. This makes them vulnerable against increasingly powerful computers or the development of new breaking algorithms. Besides that, they are all based on the concept of a secret key (i.e. a secret number), which must be stored and safeguarded. Should this key become known, for example via viruses or side channel attacks, it can be replicated indefinitely, compromising the security of all information that was encrypted with it.

The idea of physical cryptography is to get rid of this secret key and substitute it with a real, physical system. Such objects are referred to as 'Physical Uncloneable Function (PUF)' in the literature [1, 2]. Roughly, an ideal PUF should have the following properties: (i) contain a very high amount of structural information; (ii) this information is reliably and stably extractable via so-called challenge-response pairs (CRPs) of the PUF; (iii) the rate at which the information can be extracted and/or the huge number of the CRPs, prevent full characterization and (iv) no computational model can imitate the PUF. The reader is referred to the literature for more precise definitions [1] and realization attempts for such systems [2, 3, 4, 5]. Most PUF-based security protocols still require a central database, where the characteristics of PUFs (a number of CRPs) is stored and against which the PUF objects could be verified to.

An extension of the PUF concept was recently introduced by members of our group [6]. Traditional PUFs rely on the amount / concealment of the internal structural information. This new concept of the public-key PUF requires a physical system to respond to a challenge much faster than any feasible software / 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. This makes the central database redundant. This novel type of PUF is referred to as SIMPL system (SIMulation Possible, but Laborious) in [6]. The goal of this paper is to present two integrated circuit designs for SIMPL systems.

As we suggest, SIMPL systems can be considered as special purpose computer hardware, which performs a specialized computing task faster than a large-scale supercomputer. Nevertheless, they still should be simulatable by desktop-size computers in a reasonable time. They furthermore have to be compact, cheap and individualizable, so that an arbitrary number of different SIMPL-circuits could be made.

We try to meet these seemingly non-reconcilable requirements in SIMPL systems by exploiting some of the well-known computational shortcomings of digital CPUs. Section II of this paper shows how the traditional bottleneck of operating massive amounts of data in parallel can be taken advantage of: We devise a special (skew) memory, which intentionally and deterministically fails at some writing operations. The memory performs those operations on addressed cells simultaneously, and built-in logic circuitry decides the next memory operation. Section III shows some specific details on the memory design and Section IV discusses the security issues. Section V outlines another circuit design for realizing SIMPL systems: Cellular Nonlinear Networks (CNNs) [7]. In such CNN manufacture-programmed templates create a circuit, which is very time-consuming to simulate.

II. Skew–Memory Based SIMPL System

A. Structure and Function

The sketch of the skew-memory based SIMPL system is shown in Figure 1.

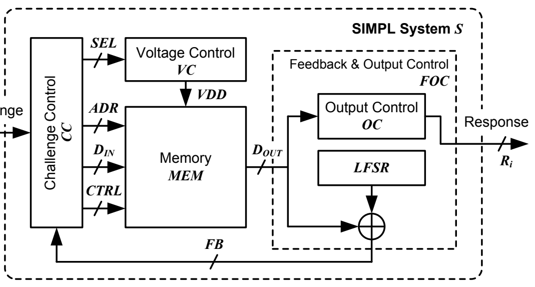


Figure 1. Schematic illustration of the skew-memory based SIMPL system.

The SIMPL system ***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 denoted as *FOC*.

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 memory for each operation. Even if millions of read and write operations are performed, the response R_i takes only tens of milliseconds to produce.

The memory *MEM* is called a skew memory because of its behavior. While a 'normal' memory faithfully stores the information written into it, a skew memory may behave differently. It may store a fixed '0' or '1' value regardless of the data written into it. It may show a 'supply-voltage dependent behavior', which means that the successfulness of write operations depends on the supply voltage *VDD*. For example, only when the supply voltage is $VDD \geq VDD_{funcmin} > VDD_{min}$, the data will be successfully stored in the cell. For $VDD < VDD_{funcmin}$, the cell's current content will not be changed.

The challenge control block *CC* 'scrambles' the input challenge and produces voltage select, address, write data and memory control signals for *VC* and *MEM*. After the first cycle, the output of *CC* will be dependent of the previous cycle and possibly from additional inputs. 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 *FOC*, the linear feedback shift register (*LFSR*) serves as a pseudo random number generator, whose output is XOR'd with the readout data D_{OUT} (like one-time pad, OTP [8]), and the result is fed back to *CC* to modify the input 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 the memory.

B. *Usage of the Skew-Memory Based SIMPL System*

A *D(**S**)* description of the circuit should contain the characteristics of the memory cells and the logic functionality of the *CC*, *VC* and *OC* units – based on these data the circuit behavior is possible to be emulated and the correctness of the behavior of ***S*** can be verified in a security protocol. If the operation of the ***S*** SIMPL system yields results faster than a previously specified t_{max} time limit, it proves that the data was provided by the actual ***S*** SIMPL system, not by another emulator (e.g. digital clone). The security of the SIMPL system is assured by its definite speed advantage over any feasible clone.

III. Design of the Skew Memory

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 designed it starting from the transistor level.

A. *Skew Memory Cell*

Since SRAMs are the fastest of all commercial semiconductor memories our design uses SRAM cells. A six-transistor SRAM cell is schematically illustrated in Figure 2, with sizes (width/length in micron) specified beside each transistor. This special sizing assures 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 discrete supply voltages VDD_{high} and VDD_{low}. The sizing of a skew cell guarantees correct 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.

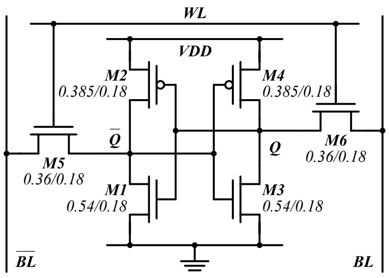


Figure 2. Skew SRAM cell in a 0.18-μm CMOS technology.

Our design is based on the TSMC 0.18 micron CMOS technology. Using the transistor sizing of Figure 2, 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 1.66 V will guarantee successful write operations and vice versa. We choose VDD_{high} = 1.8 V and VDD_{low} = 1.3 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 volt and 1.3 volt. The write operations at t=3 ns and t=7 ns are unsuccessful since *VDD* equals 1.3 V. Monte Carlo simulations with realistic process variations and mismatch show that 97% of the cells exhibit expected behaviors under VDD_{high} and VDD_{low}. Further optimization can be done in both pre- and post- layout design phases to improve reliability and yield.

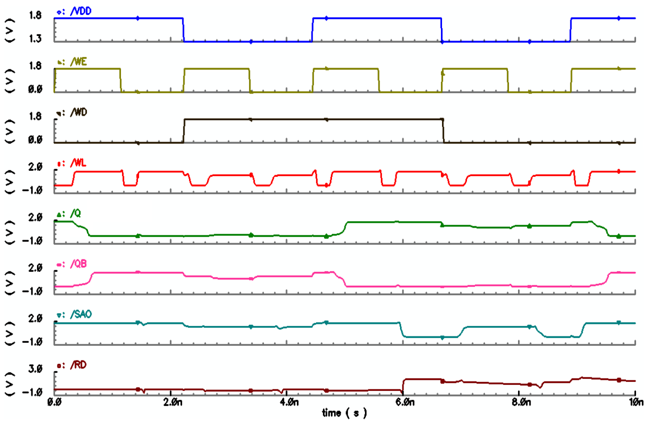


Figure 3. Operations of a skew SRAM cell with clock frequency of 900MHz. (WE: Write Enable, H/L=Write/Read; WD: Write Data; WL: Wordline; Q/QB: Cell Content; SAO: Sense Amplifier Output; RD: Readout Data)

B. *Individualization*

Individualization (adding "fixed 0" and "fixed 1" cells) of the SIMPL system can be done in the last fabrication step using laser fuses [9, 10].

Depending on where the skew cells are located, a practically infinite number of distinct memories can be fabricated: for example, assuming a 32×32 bit size array, the number of distinct designs is already larger than 10^{488}.

C. Design Parameters

The construction of the SIMPL system should exclude any computational shortcut, which a faker could use to emulate the circuit faster than a specified t_{max} time.

The bit-width of memory output D_{OUT} should be sufficiently large so that pre-computing a look-up table (LUT) for the next operation(s) is infeasible – such LUT would clearly offer a computational shortcut to emulate the behavior of the circuit. In addition, the *LFSR* should have a long cycle to avoid cyclic operations on the memory – the cycle period of the *LFSR* grows exponentially. The size of the LUT grows exponentially with the bit-width of memory output. A data bit-width of 32 bits can clearly prevent the lookup-table attack, because it would require a fast 128 gigabit ($32 \cdot 2^{32}$ bit) memory to store the pre-computed result of one computational step.

The size of the memory array (i.e. the number of rows by columns) should not be very small – otherwise different outputs of the actual cycle may lead to the same operation for the next cycle, which would decrease the required size of the LUT. We estimate that the required LUT size is over 64 gigabit, for a skew-based SRAM of 1 Mbit size (with 32 bit word-width).

Clearly, a large skew memory is also impractical. Larger memory arrays decrease the achievable operating frequency and increase the cost. We believe that a one-megabit size SRAM is a reasonable compromise between speed, cost and security level.

The speed advantage of the circuit largely stems from the fact that the supply voltage of the memory can be changed instantaneously. Larger number of supply voltage levels provides higher complexity and larger speed advantage. On the other hand more supply voltages increase the cost and decrease the stability of the operations against noise and process variations (because safe margin between different supply voltages are decreased), resulting in lower yield. Applying only two distinct supply voltages (as we did in the above design example) is still reasonable and results in reasonably high speed advantage.

Based on the above arguments, we choose 2 different supply voltages, 1 Mbit memory array and data width of 32 bit as an optimal design for the skew memory.

IV. Security Assessment

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 those possibilities infeasible.

A. Building an Exact Physical Clone

The skew-memory based SIMPL system can be re-fabricated in a silicon foundry. This is extremely expensive having one-time costs in the millions of US dollars range [11]. The system can be considered to be secure against attacks by consumers and individual hackers.

B. Building a Functional Physical Clone

One concept to build a functional physical clone of the skew memory is to combine a 'normal' (mass-manufactured) memory together with simple logic circuits to mimic the operations on skew cells. In this construction, the instantaneous change of supply voltage is substituted by a full read / write operation of the memory, which is at least a factor of two slower.

Another option is to use FPGA to replicate the behavior of the skew memory. Each FPGA logic block can be designed to act like a skew memory cell. This requires a very large (and very expensive) FPGA even for moderately sized (1 Mbit) skew memories – using multiple chips would further widen the speed gap between the emulator and the 'legal' monolithic circuit. The operating speed of a state of the art FPGA always lags behind optimized special purpose SRAM with at least a factor of five.

C. 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, every time that a write operation in $\boldsymbol{S}$ updates an address, whose content has been pre-readout, all the pre-computed results will need to be re-evaluated as all subsequent operations will be altered. The memory content that has been modified by pre-calculated write operations must be recovered to the corresponding cycle. We believe that the operation of our SIMPL system design is fundamentally sequential with no possibility to parallelize the computation.

Another possibility is to pre-compute and build lookup tables (LUTs), so that multiple cycles of $\boldsymbol{S}$ can be computed with a single emulation step. However, the storage needed by LUTs grows exponentially with the bit-width of the input. As long as each interface in the skew system is made with sufficient bus width, LUTs will be infeasible. Besides, larger LUTs also imply longer search time. 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 protocols proposed in [6].

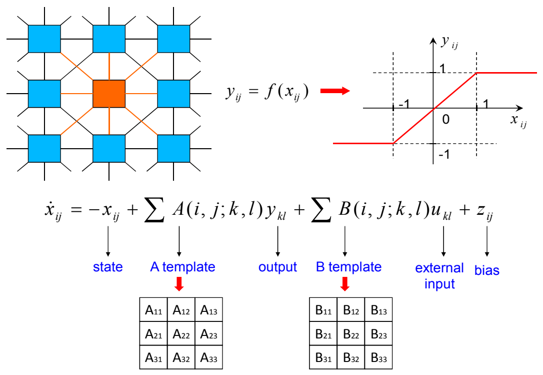
V. Purely Analog Designs: A Cellular Nonlinear Network Approach to SIMPL Systems

The computational power of the above discussed SIMPL system comes from the parallel and analog operation of a specially designed memory unit. The rest of the circuit is based on 'conventional' digital circuits.

Using a fast and fully analog circuit as SIMPL system could significantly increase the speed gap between the SIMPL system and any feasible functional clone or digital emulation. On the other hand, a predictably operating (computable) analog system can be challenging to design and build: analog circuits are susceptible to noise, manufacturing variations, temperature fluctuations, etc.

Cellular Nonlinear Networks (CNNs), a class of high-performance analog computing devices [12] may satisfy these conflicting design requirements.

CNNs are regular arrays of locally interconnected elementary processing units (cells). 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 inputs from neighboring cells. On an abstract level, the cell-behavior and the cell-to-cell relation is characterized by so-called templates, which are real-valued matrices [12]. The structure of a CNN circuit is illustrated in Figure 4.



$y_{ij} = f(x_{ij})$
y_{ij}
1
-1
0
1
-1
x_{ij}
$\dot{x}_{ij} = -x_{ij} + \sum A(i,j;k,l)y_{kl} + \sum B(i,j;k,l)u_{kl} + z_{ij}$
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}

Figure 4. Schematic layout of a CNN circuit and the equations describing its operation.

CNNs can be simulated by solving the time-dependent differential equations describing the time evolution of each cell: for example a thousand-cell CNN solves thousand coupled nonlinear ODEs 'by hardware'. Smaller CNNs can be programmable, so the templates / ODE coefficients can be set to a desired value; fixed-template CNN cells are much less complex and scalable up to several hundred thousand cells.

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 µm standard CMOS technology exhibited peak computing figures of 330 GOPS [13] (3.6 GOPS/mm^2 and 82.5 GOPS/W, projected for chip area and power consumption). In specialized tasks, it is known that CNNs can outperform digital computers by a factor of up to 1000 [7, 14].

Due to this extreme computing performance, CNN circuits are promising candidates for SIMPL systems. A CNN circuit can be programmed by the manufacturer for certain templates (image processing or partial differential equation solving [7]). The number of possible template combinations is extremely large [12], even excluding the trivially solvable or chaotic ones. Based on the public template values, the circuit can always be simulated, but no faker (who uses digital computers or programmable CNNs) will be able to match the speed of the original (custom-manufactured) CNN.

VI. Discussion & Conclusion

This paper discussed the implementations of a new cryptographic device termed SIMPL system, which had been proposed in [6]. These systems can be considered a public-key version of physical uncloneable functions (PUFs). SIMPL systems – being physical objects (just like PUFs) – overcome some of the difficulties associated with traditional cryptographic and security methods.

Our paper introduced two concrete candidates for SIMPL systems. The first skew-SRAM based circuit uses a specially designed memory to generate data, which are than 'scrambled' by logic circuits and fed back to the memory. This iteration is repeated many times to gain time advantage over a digital simulation. In the last chapter we outlined the possibility of using fully analog cellular arrays as SIMPL systems.

Acknowledgment

The authors are grateful to Bing Li and Ning Chen. This work was supported by the International Graduate School of Science and Engineering (IGSSE) and the Institute for Advanced Study (IAS) of the Technical University of Munich.

References

[1] B. Gassend, D. Clarke, M. van Dijk, S. Devadas, "Silicon physical random functions", Proceedings of the 9th ACM conference on Computer, 2002.

[2] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation", Proceedings of the 44th Design Automation Conference, IEEE, 2007.

[3] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk, S. Devadas, "Extracting secret keys from integrated circuits", IEEE Tansactions on Very Large Scale Integration (VLSI) Systems, vol. 13, pp. 1200-1205, October 2005.

[4] J. Guajardo, S. S. Kumar, G. Schrijen, P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection", Lecture Notes in Computer Science, Springer, 2007.

[5] S. S. Kumar, J. Guajardo, R. Maesyz, G. Schrijen, P. Tuyls, "The butterfly PUF protecting IP on every FPGA", IEEE International Workshop on Hardware-Oriented Security and Trust, June 2008.

[6] U. Rührmair, "SIMPL systems: on a public key variant of physical unclonable functions", Cryptology ePrint Archive, International Association for Cryptologic Research, June 1, 2009.

[7] G. Csaba, X. Ju, Q. Chen, W. Porod, J. Schmidhuber, U. Schlichtmann, P. Lugli, U. Rührmair, "On-chip electric waves: an analog circuit approach to physical uncloneable functions", Cryptology ePrint Archive, International Association for Cryptologic Research, May 29, 2009.

[8] J. A. Buchmann, Introduction to Cryptography, 2nd edition, pp. 123, Springer, 2004.

[9] R. T. Smith, J. D. Chlipala, J. F.M. Bindels, R. G. Nelson, F. H. Fischer, T. F. Mantz, "Laser programmable redundancy and yield improvement in a 64K DRAM", IEEE Journal of Solid-State Circuits, vol. SC-16, No. 5, October 1981.

[10] T. Chih-Wu, H. Hsien, "Laser fuse structure", US Patent 6404035, 2002.

[11] A. B. Kahng, "Bringing down NRE", IEEE Design and Test of Computers, May-June 2003.

[12] L. O. Chua, T. Roska, Cellular Neural Networks and Visual Computing, Cambridge University Press, 2002.

[13] A. Rodríguez-Vázquez, G. Linan-Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Galán, F. Jiménez-Garrido, R. Domínguez-Castro, S. Meana, "ACE16k: the third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs", IEEE Trans. on Circuits and Systems - I, 51(5), pp. 851-863, 2004.

[14] T. Roska: "Cellular wave computers for nano-tera-scale technology - beyond boolean, spatial-temporal logic in million processor devices", Electronics Letters 12th, vol. 43, No. 8, April 2007