GALS Design of ECC Against Side-Channel Attacks – A Comparative Study

Xin Fan[1], Steffen Peter[2] and Milos Krstic[1]
[1]IHP, Im Technologiepark 25, Frankfurt (Oder), Germany
[2]CECS, University of California - Irvine
{fan, krstic}@ihp-microelectronics.com; st.peter@uci.edu

Abstract — **Elliptic Curve Cryptography (ECC) represents the state-of-the-art of public-key cryptography. It is very computation intensive and hardware consuming for ASIC implementation. In this work, an ECC processor based on the Globally Asynchronous Locally Synchronous (GALS) design is presented. Attention has been paid on the resistances of GALS design against side-channel attacks (SCAs). The pausible clocking scheme, with random hopping of clock frequencies, is applied as a countermeasure of SCAs with low overhead on hardware. A comparative study between the synchronous and the GALS designs of ECC, in terms of the SCA resistance, processing efficiency, and hardware costs, is further elaborated.**

Keywords – ECC, SCA, GALS

I. INTRODUCTION

Security of data exchange and processing define one of the most critical issues for nowadays embedded systems. Cryptographic blocks are typically integrated, *e.g.*, in smart cards and wireless sensor nodes, for this reason. This, however, imposes at the same time the requirements of security and efficiency on hardware design. Among various schemes, the Elliptic Curve Cryptography (ECC) has proven to be powerful for providing sufficient complexity of computation on decryption. Given a medium key size of *130*-bits, it is found to be very challenging to break ECC algorithms in practice with affordable hardware resources [1].

However, mathematical decryption is not the only approach to break a cryptography device. It has been demonstrated, over 20 years ago, that the variations of some physical features of a crypto-device can expose its operating state information, thus leading to disclosure of the secret key. This type of attacks is generally termed Side Channel Attacks (SCAs) [2]. Operation time, power dissipation, and electromagnetic radiation all can be exploited. Limited knowledge on cryptanalysis is needed in many cases for applying SCAs. Implementations of a crypto-algorithm, which is secure from the mathematics perspective, *e.g.*, ECC, can be trivially breakable for SCAs.

Among different SCAs, power analysis has been given the most intensive attention. It relies on the operation and data dependency of a crypto-device in terms of the power dissipation. Two categories can be generally identified: the Simple Power Analysis (SPA) and the Differential Power Analysis (DPA). The SPA is applicable if the 0- and 1-bits of the cryptographic key result in distinct encryption operations. By observing the measured power profiles, the key bits can be derived directly. Statistical analysis is further resorted by DPA to characterize the power traces for different key bits. Experiments have already revealed the potential of DPA to detect even marginal data-dependent variations of power [3].

Imposing timing uncertainty on cryptographic operations will mask the power traces by noise. SPA and DPA therefore both can be hampered. Globally Asynchronous Locally Synchronous (GALS) design is suggested as a countermeasure for this reason [4]. It breaks down crypto-processing into a group of independent clock regions. This de-synchronizes the execution of crypto-operations fundamentally. Data synchronization is performed for communication crossing clock domains. The non-deterministic latency of synchronization, however, results in extra variations on operation time. Both effects obscure the power traces in GALS design and, consequently, challenge the power analysis. In [5] the authors explored to further boost the timing uncertainty by introducing GALS pipeline on the inner iterations of crypto-operations.

So far, most studies of GALS design are carried out on the conventional symmetric-key crypto-algorithms, *i.e.*, AES/DES (Advanced/Data Encryption Standard) [4] [5]. This, however, confines the generality of the performed evaluation. Moreover, the existing GALS schemes, such as the pipelining of crypto-operations [5], give rise to prohibitive overhead on silicon area (X3 – X25) and power dissipation (10-30%), comparing with the synchronous design. It is thus in demand to advance GALS solutions for the applications of more sophisticated algorithms of cryptography, *e.g.*, ECC.

In this paper we present the GALS design of ECC against SCAs. Our work is based on an optimal synchronous design of ECC on $GF(2^{233})$ [6]. The pausible clocking scheme reported in [7] [8] is employed for the GALS ECC adaptation. Reliable synchronization and high data-throughput essentially account for its key figures of merit in application. The major metrics of GALS design, *e.g.*, system partitioning scheme, asynchronous interconnect structure, and clock randomization strategy, have been addressed. Based on the experimental results of the synchronous and the GALS ECC implementations, a comparative study is further presented, covering a broad spectrum of SCA resistance, processing efficiency and hardware costs. To the best of our knowledge, this is the first work on the GALS design of ECC and its performance evaluation.

II. SYNCHRONOUS DESIGN OF ECC – OVERVIEW

ECC is an asymmetric cipher algorithm which uses algebraic operations on two-dimensional elliptic curves. The coordinates of elliptic curves are computed usually in specific finite fields. ECC is able to provide a high level of security with a relatively short key size. The binary extension finite fields ($GF(2^m)$) with a key size of $m = 233$ bits is employed in our work. According to [9], this corresponds to a RSA security of 2048 bits.

The computationally most complicated operation of ECC is the Elliptic Curve Point Multiplication (ECPM). The ECPM is a scalar multiplication of a point (P) on the EC and an integer k, an m-bit-long secret key which is to be protected. The trap-door property of ECMP ensures that, for known P and Q ($Q = kP$), the factor k is untraceable. Processing one bit of k calls for an inner loop of the ECPM. When the multiplication algorithm by Lopez and Dahab (LDA) is used, each inner loop has six m-bit field multiplications, five field squaring, and also three field additions [10]. A total of $m-1$ iterations of inner loop need to be performed for the ECPM on $GF(2^m)$.

The top-level block diagram of our synchronous ECC design is shown in Fig. 1 [6]. The main crypto-components include an m x m-bit multiplier (*MULT*), and an m-bit arithmetic logic unit (*ALU*) to execute squaring and addition. A controller (*CTRL*) is employed to schedule the crypto-operations of ECC. It signifies the *MULT* and *ALU* for proper processing at each clock cycle. An additional m-bit register bank is deployed to load internal parameters and to allow for external access.

The synchronous ECC processor was synthesized using the IHP *130*-nm CMOS standard cell library. The breakdowns of power dissipation and silicon area, in accordance with the post-synthesis netlist, are presented in Table I. The *MULT* is found to dominate the power of synchronous ECC design. In contrast, the *ALU* is negligible in both area and power. The *CTRL*, along with the register banks, is comparable in silicon area with the *MULT*, yet with much smaller power consumption.

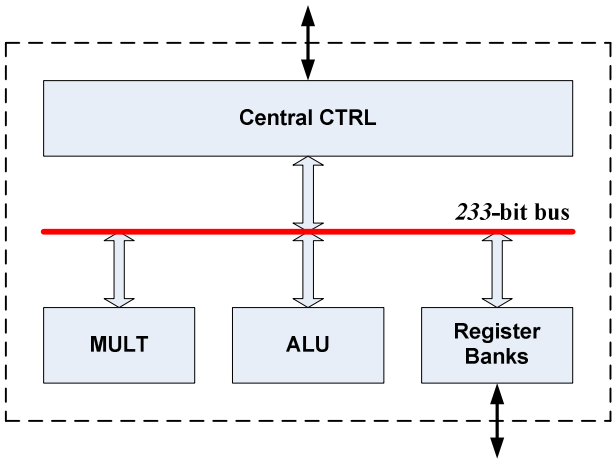


Fig. 1 Block diagram of the synchronous ECC processor

Tab. I Power and area breakdowns of SYNC ECC processor

	CTRL	ALU	MUTL	TOTAL
Power (*mW*)	1.71 (28%)	0.42 (7%)	3.97 (65%)	6.10
Area (mm^2)	0.11 (45%)	0.02 (10%)	0.11 (45%)	0.24

III. GALS DESIGN OF ECC

Previous work on the GALS design of cryptography shows significant hardware overhead on both power and silicon area over the synchronous design [4] [5]. In terms of the embedded applications, however, design efficiency usually is at least of equal importance to the robustness against SCAs. Two design issues fundamentally determine the efficiency of a GALS system: design partitioning and asynchronous interconnect. Moreover, attention is deserved by the frequency randomization of GALS clocks to hamper SCAs in cryptographic devices.

A. System Partitioning

Based on the power and area breakdowns among functional units, a partitioning scheme of the GALS ECC design has been explored as depicted in Fig. 2. Three individual clock domains, each consisting of a functional unit, are introduced. By timing each of the GALS local clock at an individual frequency, the switching activity, and thereby the induced dynamic power, of different functional units will be desynchronized accordingly. This complicates the characterization and alignment of power traces measured on the GALS ECC design, and thus challenges the power analysis attacks.

B. Asynchronous Interconnect

The dataflow of the ECC processor can be characterized in three aspects. First, rather frequent communications have to be carried out between the *MULT/ALU* and the *CTRL*: six round-trip data transfer between *MULT* and *CTRL*, and five between *ALU* and *CTRL*, within each of the ECPM inner-loop iterations. Second, the data transfer is executed in a quite large bit width. As seen in Fig. 1, the data bus has a width of 233 bits. Third, all the transfers are scheduled by the *CTRL* unit explicitly. That is, the *CTRL* decides the operation time to send/receive an item of data to/from the *MULT* and/or the *ALU* units.

The distinct features of ECC dataflow actually necessitate an elegant implementation of asynchronous interconnect across clock domains. The brute-force synchronization, which is done by double cascaded flipflops as used in [5] for the AES cryptography, is infeasible in our case. Despite its low hardware overheads, it would give rise to a significant performance loss, due to the multiple clock cycles suffered by each of the inter-clock-domain data transfers. The dual-clock FIFOs, which represent the most popular solution nowadays towards the GALS design of high performance, barely fit our demands as well. To offer a high data-throughput over the asynchronous clock domains, the FIFO should be sufficiently deep. This implies a prohibitively high cost of hardware, especially in silicon area, considering the large data width (233-bit) of the ECC design.

The above discussions account for our effort of applying the pausible clocking scheme for the asynchronous interconnect in GALS ECC design, as depicted in Fig. 3. For each of the GALS blocks a local clock generator is employed. It is in principle a gated ring oscillator, with an adjustable clock phase for the safe synchronization of input data. Communication across clock domains is executed by four handshake channels, each consisting of a pair of I/O ports (IP/OP). To accommodate the *CTRL*-unit scheduling of dataflow in ECC, two types of data channels are utilized: the push-type channels to send data from *CTRL* (the green arrows in Fig. 3), and the pull-type channels to get data from *MULT/ALU* (the blue arrows).

C. Clock Randomization

The clock frequency of each GALS block is independently configured by programming the delay of a ring oscillator in the corresponding clock generator. A well-known prototype design of the pausible clock generator can be referred to [11]. It allows for the static configuration of clock frequency, *i.e.*, configuring the delay length of a ring oscillator in the idle (reset) mode. For the resistance of SCAs, however, dynamic frequency adjusting, preferably to support the cycle-by-cycle frequency hopping, is required for the GALS design of ECC.

A pausible clock generator, which offers random frequency hopping (RFH) on its output clock, has been developed in our work. Fig. 3 illustrates the top-down hierarchical view. Rather than the design of [11], two cascaded delay lines, which can be independently configured, are employed in the ring oscillator. The maximum clock frequency achievable for a GALS block determines the length of the first–stage delay line: it is constant and is programmed via a JTAG interface prior to enabling the ECC. As a contrast, the second-stage delay line is dynamically configured in the running time of ECC. Its delay varies in each clock cycle, according to the output value of a random number generator (RNG). Frequency hopping of the generated clock is introduced as a result.

It is, however, crucial to avoid glitch on the clock signal due to frequency hopping. The dynamically configured delay line (DCDL) has been implemented deliberately for this purpose. It is made up of a series of delay slices, which are identical in the structure as shown in Fig. 3(d). Each slice has a control signal, *cc*, to decide its working status: if *cc = 1*, it is activated; otherwise, deactivated. Once a delay slice is deactivated, all its subsequent slices get bypassed as well, thereby adapting the delay line. Note that, given the low inputs of a delay slice, *i.e.*, *fin = bin = 0*, its outputs will keep to be low, regardless any transient pulses on *cc*. The DCDL therefore allows to be configured, yet without giving rise to glitch, when its output goes low. This has been achieved by triggering the RNG at the falling edges of the generated clock, as depicted in Fig. 3(b).

IV. Performance Evaluation

The GALS ECC design was synthesized using the IHP 130-nm CMOS standard cell library. Its performance has been then evaluated based on the post-synthesis netlist and simulations. In the following a comparative study between the synchronous and GALS ECC designs, in terms of the SCA resistance, hardware costs, and processing efficiency, will be addressed. Also included here is a short discussion on the experimental results derived in our work.

A. Resistance Against SCA

The power profiles induced by the synchronous and GALS designs of ECC were estimated using the *Synopsys PrimeTime*. Fig. 4 illustrates an example of the obtained dynamic power for comparison. Three design cases have been taken into account: the synchronous baseline design, the GALS design with plesiochronous clocking and the GALS design with clock RFH. In all cases exactly the same six inner-loop iterations of ECPM are covered, which correspond to the processing of six bits of key. The processing time of each key bit k_i, according to the netlist simulations, is manually back-annotated onto the power traces (the slash lines) for the clarity of discussion. The waveforms of clock signals applied in each case are also sketched.

First addressed in Fig. 4(a) is the power profile induced by the synchronous design of ECC. Given a certain value of k_i, the synchronous design requires a fixed time of execution for each of the inner-loop iterations: 56 clock cycles when $k_i = 1$ and 54 clock cycles when $k_i = 0$. Two clock cycles have been saved in case of $k_i = 0$. This accounts for a marginal speedup of the ECC processing. On the other side, however, the distinctive patterns of dynamic power have been introduced: a remarkable drop on power dissipation occurs for $k_i = 1$. The key bit involved in each of the inner-loop iterations can be accordingly inferred by the characterization of power traces. That is, the implemented synchronous ECC is vulnerable even to SPA.

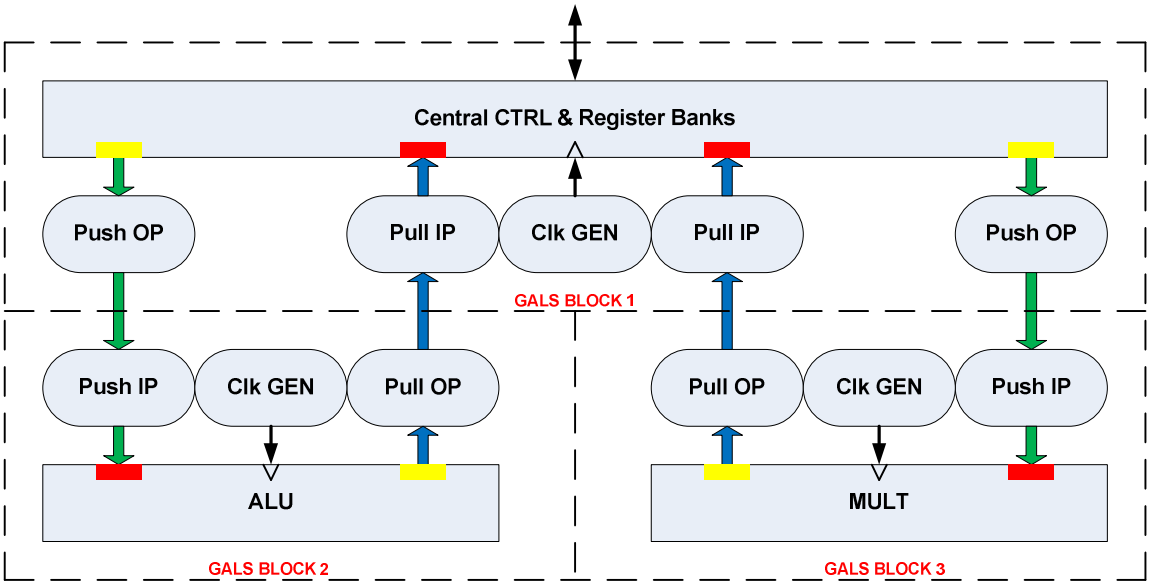


Fig. 2 Top-level block diagram of the GALS ECC processor

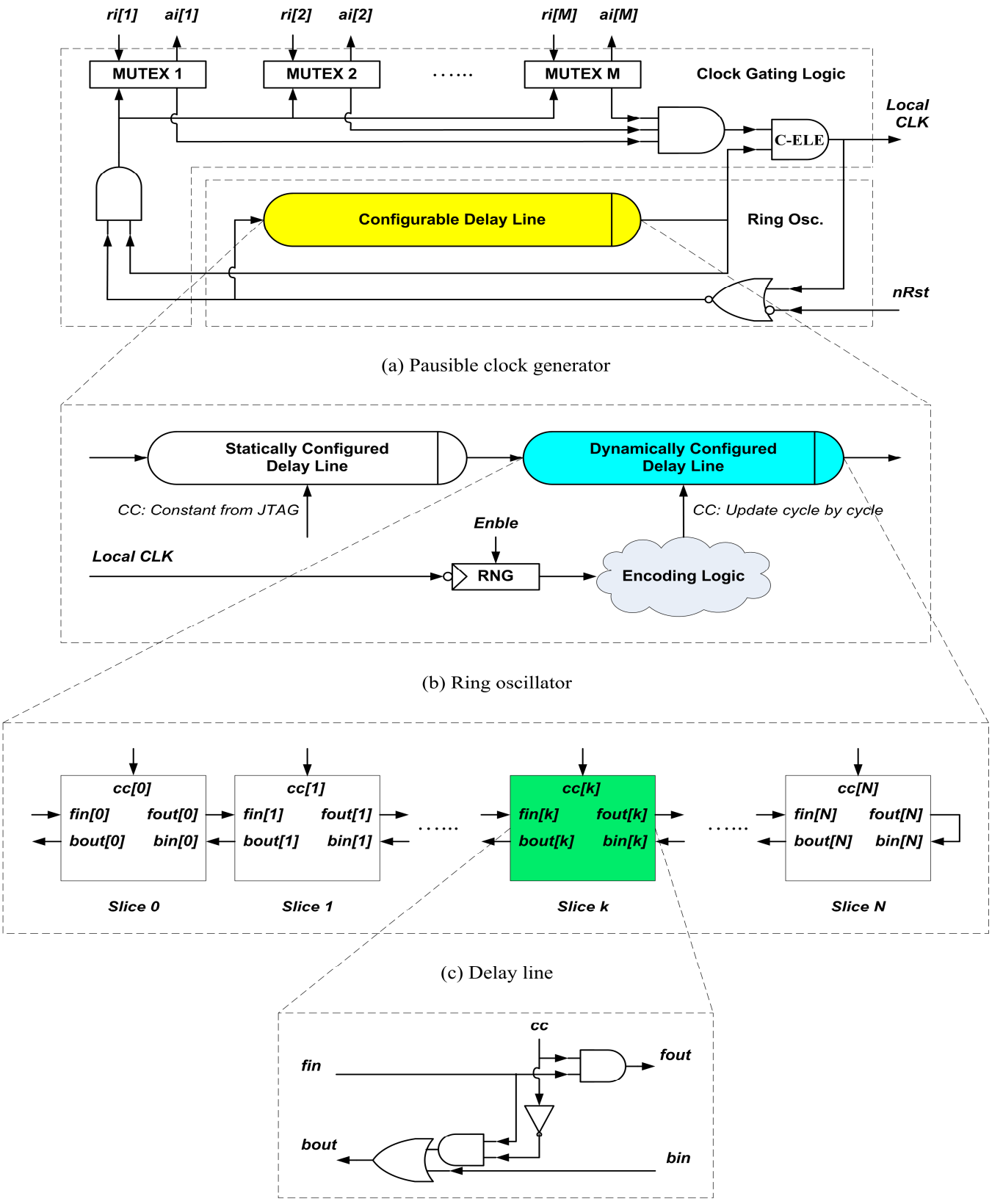
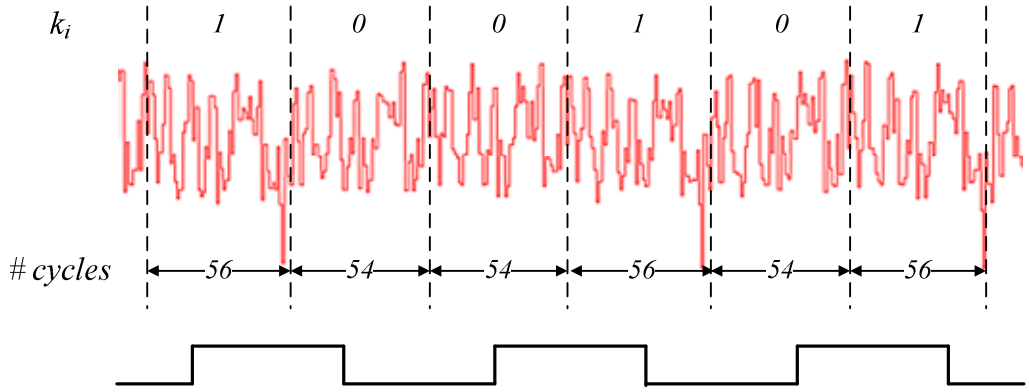
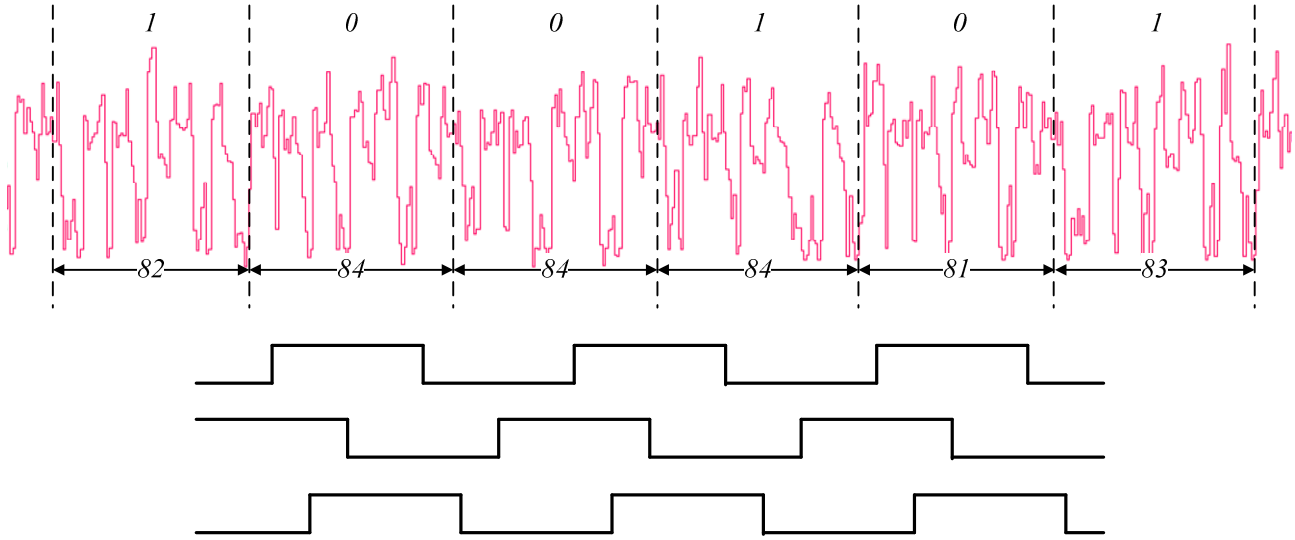


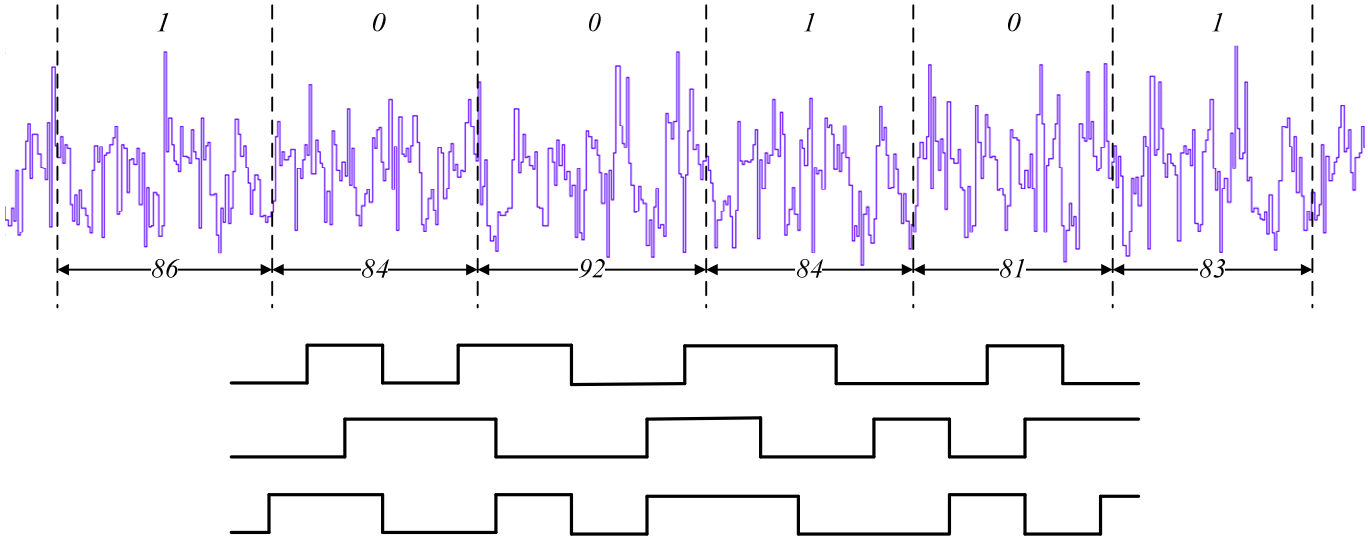
Fig. 3 Hierarchical view of local clock generator supporting dynamic frequency hopping



(a) Synchronous ECC @ T_{CLK} = 30 ns



(b) GALS ECC with plesiochronous clocking @ $T_{CLK} \approx 30$ ns



(c) GALS ECC with random hopping of clock frequency @ 20 ns $\leq T_{CLK} \leq$ 40 ns

Fig. 4 Comparison in power profiles of synchronous and GALS ECC designs (six inner-loop iterations)

Two special working modes have been considered in terms of the GALS design of ECC. Fig. 4(b) illustrates the dynamic power induced in case of the plesiochronous clocking. That is, all the local clocks are programmed to be almost the same in frequency, only having a tiny mismatch with each other. The phase drifting between local clocks is incurred consequently. Synchronization is thus necessary for transferring data reliably across clock domains. Due to the non-deterministic latency of synchronization, however, delay uncertainty is imposed on the crypto-operations of ECC. A variant time of execution therefore can be taken by each of the inner-loop iterations. As disclosed in Fig. 4(b), this diminishes the leakage of k_i by making the respective power patterns less characterized in comparison with the synchronous design.

The random frequency hopping is further evaluated for our GALS design of ECC. The linear feedback shift register was used as a simple example to get the (pseudo) random numbers. Each local clock is programmable independently of the others, with a frequency tuning cycle by cycle. Fig. 4(c) sketches the waveforms of the generated clocks for reference. The random hopping of clock frequencies allows to further de-synchronize the ECC crypto-operations than the plesiochronous clocking. The timing uncertainty that is introduced on each of the inner-loop iterations is magnified. It turns out to be rather infeasible to trace the values of k_i by characterizing and distinguishing the power patterns.

B. Processing Efficiency

The benefit of GALS design against SCAs comes at a cost of the processing efficiency. As manifested in Fig. 4, additional clock cycles are expended on each of the inner-loop iterations. Tab. II reports the running time of an ECPM in the three cases. The GALS ECC was set to work at a frequency on average the same as in the synchronous ECC. The plesiochronous clocking and the RFH clocking are both ~50% over the synchronous one in terms of the execution time.

Tab. II Comparison in processing efficiency of ECC

	Synchronous *@ 33 MHz*	GALS with Plesio. Clocking	GALS with RFH
Proc time per ECPM (μs)	396	590	608
Performance drop	----	49%	52%

C. Hardware Costs

The GALS design also gives rise to the hardware overhead comparing with the synchronous design. As shown in Tab. III, it leads to an increase of 34% in power consumption, and takes 2.41 times of silicon area. Introducing RFH on GALS ECC is found to be negligible in hardware costs. However, to some extent this is due to the generation of pseudo random number by linear feedback shift register.

Tab. III Power and area of GALS ECC design

	GALS BLOCK 1	GALS BLOCK 2	GALS BLOCK 3	TOTAL
Power (mW)	2.79 (34%)	0.99 (12%)	4.42 (54%)	8.20
Area (mm^2)	0.31 (55%)	0.06 (10%)	0.21 (35%)	0.58

D. Discussions

Our GALS design solution based on pausible clocking presents an alternative of low hardware cost for cryptography. As a contrast, the pipelined GALS design suggested in [5] relies on the unrolling of inner-loop iterations of crypto-operations. Essentially it suffers from an expense of power and area which is proportional to the number of GALS blocks in use. For the desynchronization of three clocks, a silicon area of five times over the synchronous design can be induced [5].

The GALS design of ECC challenges the DPA attacks as well. Note that DPA has to be performed on the same crypto-operation over massive power traces. The alignment of power traces is crucial due to this reason. However, the variations on operation time imposed by the GALS design, as exhibited in Fig. 4, significantly misalign the power profiles. This therefore complicates the practical applications of DPA. Further work on the GALS ECC against DPA is in progress.

V. Conclusions

This is the first work reported in the literature on the GALS design of ECC. The pausible clocking scheme, with random frequency hopping of GALS clocks, is employed. It hampers SCAs by de-synchronizing the cryptographic operations over time, and advances the GALS design of hardware efficiency for embedded applications.

References

[1] D. V. Bailey *et. al.*, "Breaking ECC2K-130," in *Cryptology ePrint Archive, Report 2009/541,* 2009.

[2] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems," in *Proc of Advances in Cryptology* (CRYPTO), 1996.

[3] S. Mangard, E. Oswald, T. Popp, "Power analysis attacks: revealing the secrets of smart cards," ADIS book series, Springer, 2007.

[4] F. K. Gurkaynak, "GALS system design: side channel attack secure cryptographic accelerators," Ph.D thesis, ETH Zurich, 2006.

[5] R.I. Soares, N.L.V. Calazans, F.G. Moraes, P. Maurine and L. Torres, "A robust architectural approach for cryptographic algorithms using GALS pipelines," in *IEEE Design & Test of Computers*, 2011.

[6] S. Peter, P. Langendoerfer and K. Piotrowski, "Flexible hardware reduction for elliptic curve cryptography in GF(2m)," in *Proc of Conf. on Design, Automation and Test in Europe* (DATE), 2007.

[7] X. Fan, M. Krstic, and E. Grass, "Analysis and optimization of pausible clocking based GALS design," in *Proc of 26th IEEE Intl. Conf. on Computer Design* (ICCD), 2009.

[8] X. Fan, M. Krstic, and E. Grass, "Performance analysis of GALS datalink based on pausible clocking," in *Proc of 18th IEEE Intl. Symp. on Asynchronous Circuits and Systems* (ASYNC), 2012.

[9] A. K. Lenstra and E. R. Verheul, "Selecting cryptographic key sizes," in *Journal of Cryptology: the Journal of the International Association for Cryptologic Research*, 2001.

[10] J. Lopez, and R. Dahab, "Fast multiplication on elliptic curves over *GF(2^m)* without preconputation," in Proc. of *Intl. Workshop on Cryptographic Hardware and Embedded Systems* (CHES), 1999.

[11] S. Moore, G. Taylor, R. Mullins, and R. Robinson, "Point to piont GALS interconnect," in *Proc. of 8th Intl. Symp. on Asynchronous Circuits and Systems* (ASYNC), 2002.