Parallel VLSI Architectures for Cryptographic Systems

Fabio Ancona, Alessandro De Gloria, and Rodolfo Zunino

DIBE - University of Genoa - Via all'Opera Pia 11a - 16145 Genova - Italy
e-mail(s): {ancona, degloria, zunino}@dibe.unige.it — Phone: +39- 10 - 3532268, Fax: +39- 10 - 3532175

Abstract

This paper describes a parallel VLSI implementation of a private-key cryptographic system based on Peano-Hilbert curves. The basic unit of the VLSI architecture is the Crypto Processor, that is an SIMD composed of a grid of 256x256 processing units performing elementary operations of encoding process. The key lenght of the system, measured as number of free parameters, depends linearly on hardware complexity: the cryptographic system is modular and its implementation is very cheap. The CP has been implemented as a single chip with a 1-micron CMOS technology and shows a working frequency of 30 MHz. The chip can be used in consumer applications as well as add-on whenever a certain degree of safety in communication is required.

1. Introduction

Until recently, cryptography [1] has been of interest both to the military and diplomatic communities and to the commercial organizations.

Public-key and private-key approaches represent basic paradigms for cryptographic systems. The operational baseline of public-key cryptography is that no algorithm is known yet to factor products of large prime integers in a polynomial time. This feature motivates both the success and the controversies associated with public-key cryptology.

Private-key methods can benefit from a wider variety of structural approaches; although building a really safe cryptographic schema is very difficult, novel, well-designed schemata may result in additional complexity that will hamper a cryptanalyst's task [2, p.68]. The Data Encryption Standard (DES) [3] represents a reference model for this class of approaches. On the other hand, critical drawbacks of such methods are the required security in key distribution and possible hidden implementation flaws: an algorithm may appear quite robust in theory and prove insecure when implemented in practice [4].

Computational complexity is a key property of a cryptographic system, since it gives a measure of the effort that a cryptanalist must spend to break a cryptosystem. In ideal cryptographic system, cryptography complexity is measured by the key lenght, for example by the bit number representing free parameters. This paper presents a fractal approach that makes cryptography complexity grow linearly with the hardware implementation. The fractal nature of Peano-Hilbert curves [5-9] is exploited to draw the architecture VLSI layout, which integrates several modular processing units (PUs) (Fig. 5) performing elementary operations. The overall cryptographic function stems from the collective behaviour of all the PUs. The basic unit of the VLSI architecture is the Crypto Processor (CP) (Fig. 4), which can be used alone or with other CPs when it is necessary to extend the hardware implementation.

The CP is an SIMD composed of a grid of 256x256 PUs and a control unit. The PU interconnection topology can be programmed at run time through connection switches. This makes it possible to implement two different 2-D Peano-Hilbert curves (Peano1, and Peano2). Each 2-D curve defines a pipeline along which the data shift while the cryptographic process is being performed. In addition, each unit supports elementary pseudorandom functions at the character- and bit-wise levels. The CP has been implemented as a single chip with a 1-micron CMOS technology and shows a working frequency of 30 MHz. This implementation is very cheap, as it does not stress the technology but shows a good performance for the exploitation of the parallelism intrinsic in the algorithm.

Section 2 describes the basic cryptographic schema and its supporting architecture; in addition, some results from elementary tests are shown. Section 3 shows a VLSI implementation of the cryptographic system. Some concluding remarks are made in Section 4.

1066-1395/97 $10.00

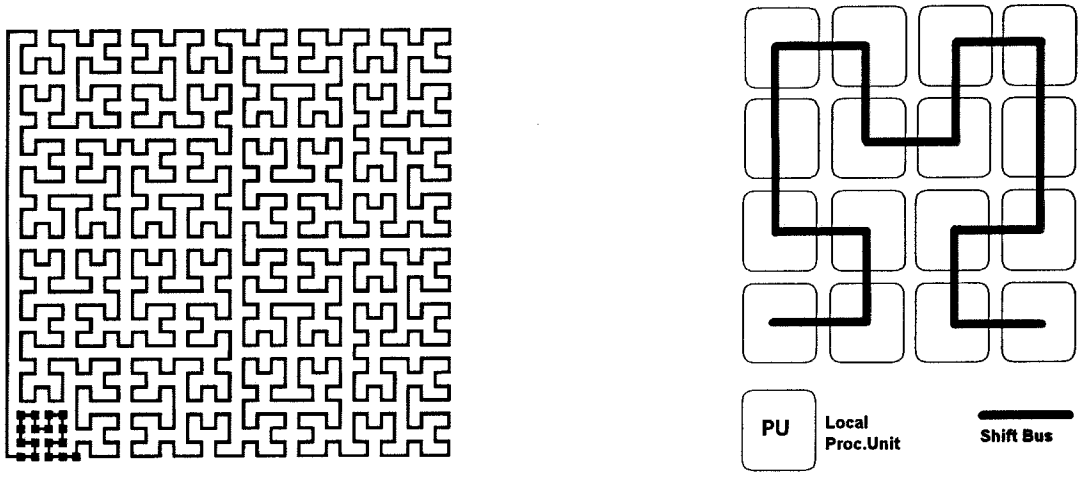

Fig.1 - Fractal layout of PUs.
a) A 32x32 Peano-Hilbert curve. **b) Elementary struct. w/ PU cells.**

2. Theory

2.1 Basic cryptographic architecture

The basic idea is to lay out PUs along one closed two-dimensional path operating as a circular shift register (bus). The (plain)text is initially loaded into the PU grid, and encryption proceeds in a synchronous way: in each cycle, a PU performs two processes: *local encoding* and *bus shift*. Decryption is accomplished by reversing these processes.

The importance of the PU layout lies in its effectiveness in rearranging units to define different sequences; this is far beyond the (trivial) transposition involved by circular shifting. In this sense, any interconnected grid might be satisfactory for character scrambling. In fact, the crucial requirements for a schema to be effective can be stated as follows: 1) at run time, the probability of a character's position must ideally be uniformly distributed among all possible units; 2) the actual layout must be scale-invariant to inhibit bounds to the size of a practical implementation. For example, a classical "mesh" structure can prove effective after a sufficient number of pseudorandom row-wise and column-wise cyclic shifts, but does not satisfy the second constraint, as the number of "cut wires" to enlarge a layout grows with a system's size.

The solution adopted in this paper involves the use of 2-D fractal Peano-Hilbert curves [5] to support the PU grid. Figure 1 shows an example of such a layout, in which the marked points correspond to PUs. The choice of this specific family of curves is justified by various reasons. From a structural point of view, a curve follows a two-dimensional layout but is characterized by a one-dimensional structure. From a cryptographic perspective, according to fractal theory, if the curve density (i.e., the number of points per coordinate unit) tends to infinity, the curve fills up the entire 2-D domain. As a result, the eventual position of a shifted element becomes unpredictable (a few shifts are sufficient to destroy spatial correlation); this implies that increasing the number of PU leads to the desired uniform probabilistic distribution of character positions. This feature, joined with the iteration of local unit processing, yields the pseudorandom behaviour of the system. Finally, from a design viewpoint, the nature of the two-dimensional layout allows a curve to be half-cut, but each portion maintains the same properties as the original curve. In particular, suitable dichotomies of Peano-Hilbert curves always leave two "open wires", independently of the two partitions' sizes. This feature proves extremely useful for practical implementations, as it provides the designer with a straigthforward method to partition a curve and to allocate the segments to available platforms (Fig.2). For instance, in a VLSI implementation, this implies that only one couple of I/O ports is always required.

The use of a single curve might grant a cryptanalyst the advantage of the one-dimensional nature of the structure, which still preserves some intrinsic correlation. This flaw can be removed without affecting the method's basic principles by superimposing a second, rotated fractal curve upon the first one. At run time, the encoding

process switches between the two alternative layouts. It is easy to see that this extension preserves the previously described advantages, that is, structural simplicity, randomizing ability, and flexible implementation. Figure 3 presents the schema of the overall, double-layout system architecture.

In the following, one summarizes the encoding process in the form of an algorithm.

Local Encryption

N = number of PUs; L = number of cryptographic cycles,
Δ_i = i-th shift width;
Set the interconnection so as to organize the PUs as a grid;
For i=1 to N
    Load character C_i into unit PU_i
    Load Seed$_i$ into unit PU_i
        (*Seed* = initialization number supplied to the pseudorandom number generator (PNR) of the unit PU_i; the PNR's output is $Mask_i$)
end for;
Change interconnection for implementing the Peano-Hilbert curve (there are two pseudorandom choices: either 0° or 90° rotated curve);
For i=1 to L
    For each PU_i $i \in \{1,N\}$
        $x_i = C_i$ XOR $Mask_i$;
        $x_i = Scramble\ (f_i(x_i))$
        (the function f_i uses a lookup table for rearranging the bits of the char x_i)
        If the unit PU_i is *not* in the first position
            $x_i = x_i$ XOR x_{i-1} ;
        For d=1 to Δ_i
            $x_i = x_{i+1}$;
        Switch connections according to the alternative Peano-Hilbert curve
        end for;
end for;
Download resulting cyphertext in the type-mode used to load plaintext.

The distributed algorithm and the specifc layout properties of the fractal curves greatly facilitate any HW implementation of the method on parallel VLSI architectures.

2.2 Results from Elementary Tests

The method was tested on two square Peano-Hilbert grids of varying sizes; the number of elements ranged from a simple 8x8 grid to a larger 64x64 curve. The use of a few units aimed to stress the method's effectiveness, as much larger plaintexts were adopted as input samples. Due to the preliminary nature of the presented results, quite simple criteria were used to evaluate the quality of the encryption process; more sophisticated tests are currently being developed, including, among others, chaotic tests.

A frequency-based test working out a cyphertext FFT made it possible to inspect possible cyphertext-inherent structures. Applying this criterion with 1) many different key values to the same plaintext, and 2) many different

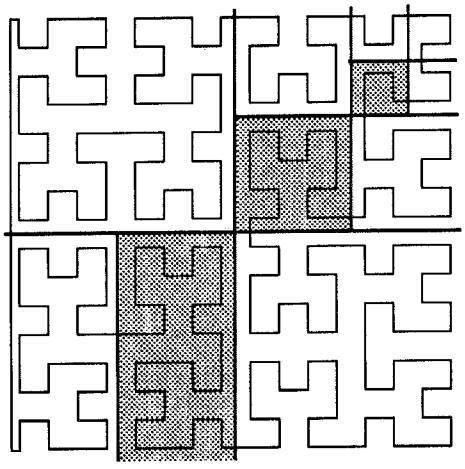

Fig. 2 - Dichotomies in Peano-Hilbert curves.

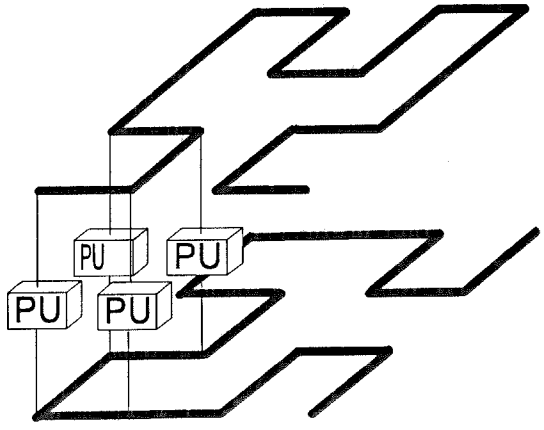

Fig.3 - Use of two rotated curves.

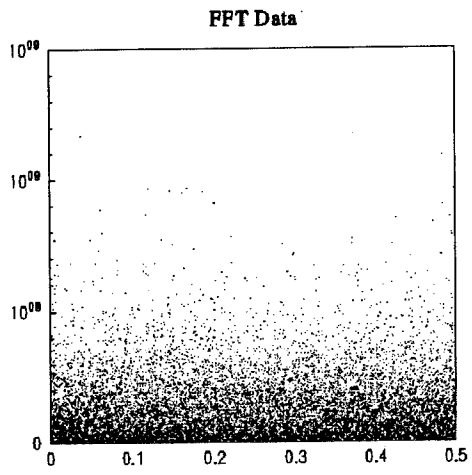

Fig. 4 - Sample power spectrum from frequency cyphertext analysis (x-coordinate: normalized frequency; y axis: corresponding power)

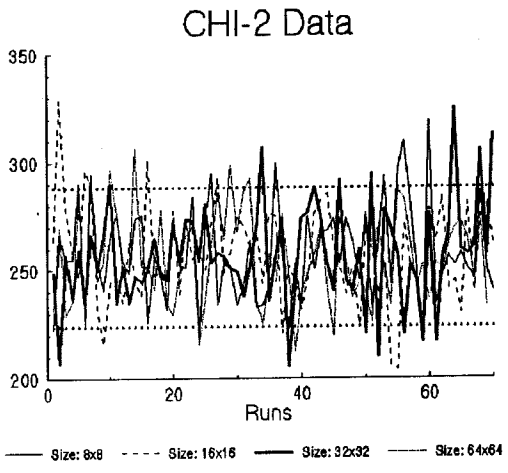

Fig.5 - Sample chi-square results from independent runs

input texts, involved massive experiments that yielded a huge amount of data.

Averaging over cross-correlation tests among different FFTs did not reveal any significant relationship or hidden structure. Figure 4 presents a sample power spectrum of such an FFT; showing a quasi-uniform distribution of energy among cyphertext frequencies.

The chi-square test provided an assessment of the "randomness" of the cyphertext. Again, the test considered different keys to a common plaintext and many different plaintexts with varying statistics. The experiments included runs encoding a common (20,480 byte long) plaintext with 1-bit- distant keys (to evaluate sensitivity to key variations), and varying grid sizes, ranging from 8x8 up to 64x64. The graph presented in Fig.5 shows that the statistical indexes are satisfactorily within the tolerance range (256±32); a significant result of these experiments is the confirmation of the method's effectiveness in terms of mask-scale invariance. The same tests using DES yielded chi-square values never smaller than 3,000.

These results just represent a confirmation of the model's practical effectiveness, as may suggest partial fulfilment of a necessary condition (i.e., statistical random-likeness). This experimental evidence, anyway, cannot be directly related to safety features: the mere fact that the method passed a few statistical tests does not imply anything about a priori safety, whose proof is now being developed theoretically.

3. Implementation

The basic unit of the VLSI architecture is the Crypto Processor (CP), which can be used alone or with other CPs, when it is necessary to increase cryptography complexity (Fig. 6). The CP is an SIMD composed of a grid of 256x256 PUs, that perform elementary operations of the encoding process, and a control unit (CU) managing the CP functioning. The overall system is modular and it is arranged as a pipeline of CPs: the first CP is the *master* processor, the other are *slave* processors.

The first PU of a CP differs slightly from the others: a control logic bypasses the second *xor* whenever the CP works in master mode. As a matter of fact, the first PU plays a crucial role in the encoding process, as it remains unaffected by the parallel process.

Each PU (Fig. 7) is equipped with a pseudorandom number generator [6] (PRN) (Fig. 8), a scrambler (SF), and glue logic. The PRN generates a sequence of mutually uncorrelated masks that alter the statistics of the

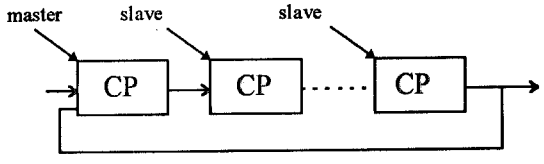

Fig. 6 - The organization of the system with multiple CPs.

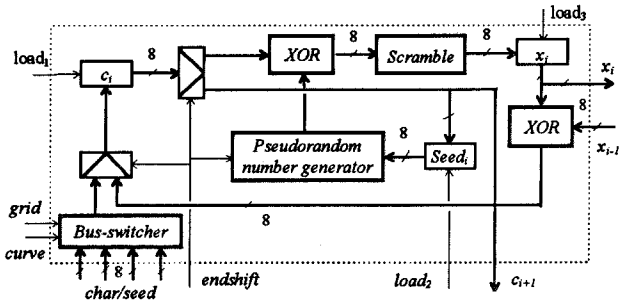

Fig. 7 - A single processing unit (PU).

encoded text. Figure 6 shows the schematic diagram of the pseudorandom number generator; the circuit includes a shift register and a counter. By an 8-bit seed and a suitable initialization of the counter device, one can obtain a valid pseudorandom sequence with a 256x511 period. The SF is used to make the sequence of masking operations not trivially invertible and non-linear.

At the reset, the PUs are connected as a linear vector through the grid switch; in this configuration, the seeds for the PRN are stored in the PUs as well as the chars to encode. Then the Fig. 7 topology switches to one of the possible Peano-Hilbert curves and the encoding process begins.

The encoding is accomplished in four steps that are repeated L times. First, an *xor* with the mask is done, the result bits are scrambled locally, and then an *xor* between the scrambled bits in the PU_i and those scrambled in the PU_{i-1} is performed. These three steps are completed within three clock cycles. The fourth step performs a shift of the character sequence. Each PU transmits the locally encoded char to the following PU in the Peano-Hilbert curve. The duration of this step depends on the amount of shiftings required; usually, ten shifts are enough so that a single encryption pass lasts 12 cycles.

The encryption operations are scheduled by the control unit (CU) (Fig. 9), whose actions are controlled by a simple FSM. The CU implements a pseudorandom generator (Fig. 8) for initializing the PRN contained in the PUs. The CU also contains two counters used to implement the indexes of the external loop and of the sequence shifting.

The CU has been designed for a possible incremental use of the CP. When multiple CPs are used, only one CU is active, whereas the others are disconnected. This operating mode is controlled by the master/slave signal. The registers and the counter that depend on the length of the sequence have been designed with a length of 32 bits. Their effective length is programmable via software through the control register. This enables the user to set the number of cryptographic cycles, L, the number of characters to encode, N, and the number of shifts, Δ, at each cryptographic cycle.

The CU has also been designed to interface a master processor in a synchronous or asynchronous way. This can also be chosen by software through the control register. In the synchronous operating mode, the master

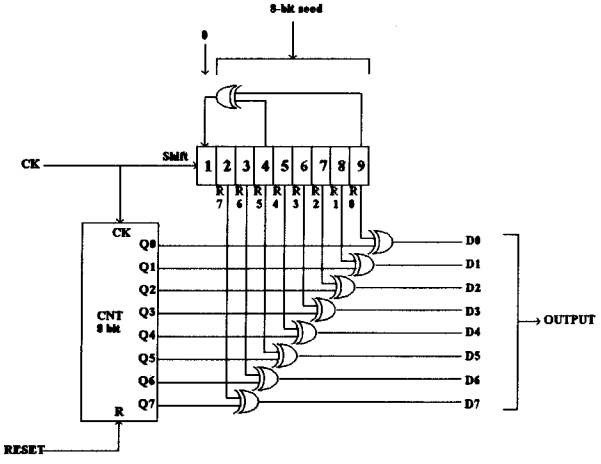

Fig. 8 - The pseudorandom number generator.

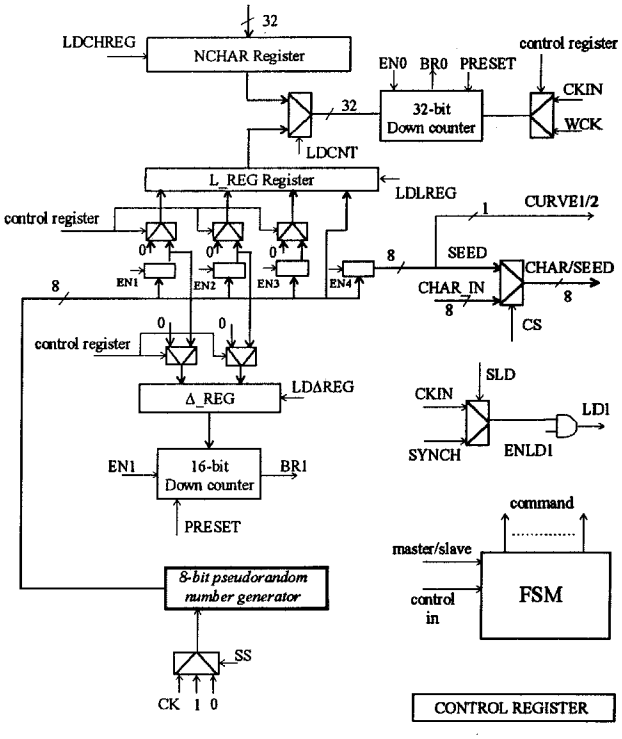

Fig. 9 - The Control Unit (CU).

processor sends a start command and then the CP loads one character at each clock cycle; a handshaking mechanism rules the asynchronous mode.

The CP has been implemented as a single chip with a 1-micron CMOS technology and shows a working frequency of 30 MHz. This implementation is very cheap, as it does not stress the technology but shows a good performance for the exploitation of the parallelism intrinsic in the algorithm. The computed speed-up versus a sequential implementation is

$$sp = \frac{(sN + sh)L}{(s + sh)L} = \frac{sN + sh}{s + sh} \approx N$$

where L is the length of the external loop, s is the number of steps for performing the mask and the scramble, sh is the number of shifts after each iteration, and N is the length of the sequence. As we can obtain good results by a value of sh in the range [3,10] and as N is usually equal to about 4000, we have an approximate speed-up of N and the approximation becomes better with greater values of N. This demonstrates the linear dependence of cryptography complexity on hardware complexity.

4. Concluding remarks

We have described the architecture of a VLSI implementation of a cryptographic system. The system is scalable and its complexity grows linearly with the complexity of the encryption. In fact, thanks to the distributed and modular nature of the system, the cryptographic complexity can be easly controlled at design time just by increasing the system size, either at the chip or at the board level without any specific reconfiguration. The chip is inexpensive and can be used in consumer applications as well as add-on whenever a certain degree of safety in communication is required.

References

[1] Diffie W., and Hellman M.E., "Privacy and Authentication: An Introduction to Cryptography", *Proceedings of the IEEE*, vol.67, No. 3, March 1979, pp. 379-427.

[2] Simmons GJ (Ed.) *Contemporary Cryptology*, IEEE Press, 1992.

[3] "Data Encryption Standard (DES)", *FBS (US), FIPS* Publ. 46, Nat.Tech.Info.Serv., Apr. 1977.

[4] Zeng K.Z., et. al., "Pseudorandom bit generators in stream-cypher cryptography", *IEEE Computer*, Feb. 1991, pp. 8-17.

[5] Peano G: Sur une courbe, que remplit toute une aire plaine. *Math Annalen*, 1890, vol. 36, pp.157-160.

[6] Hilbert D: Ueber die stetige abbildung einer linie auf ein flächenstück. *Math Annalen*, March 1891, vol. 38, pp. 459-460.

[7] Abend K, Harley TJ, Kanal LN: Classification of binary random patterns. *IEEE Trans. Inform. Theory*, Oct. 1965, vol. IT-11, pp. 538-544.

[8] Butz AR: Convergence with Hilbert's space filling curve. *J. Comput. Sys. Sci.*, May 1969, vol.3, pp. 128-146.

[9] Butz A.R.: "Alternative algorithm for Hilbert's space-filling curve", *IEEE Trans. Comput.*, April 1971, pp. 424-426.

[10] Anguita D., Rovetta S., and Zunino R.: "Compact, digital pseudo-random number generator", *Electronics Letters*, vol. 31, n. 12, pp. 956-958.