A research project on FPGA-accelerated cryptographic computing

(Invited Paper)

Domenico Argenziano
University of Naples Federico II
domenico.argenziano@unina.it

***Abstract*—This invited presentation describes the contents of a research programme aimed at the definition of new components for FPGA-based cryptographic computing. The text describes the state of the art and the context of the proposed research, the methodology, as well as the current preliminary results, with particular emphasis on the ongoing development of an FPGA-based homomorphic encryption accelerator.**

I. Introduction

This invited presentation describes the contents of a research programme currently being carried out by the author aimed at the definition of new components for FPGA-based cryptographic computing. In fact, an increasing number of applications are requiring hardware-accelerated cryptographic operations [1]. Current trends in computing architectures offer a variety of solutions, ranging from general-purpose multi/many-core processors to Graphics Processing Units (GPUs) and Field-Programmable Gate Arrays (FPGAs). In particular, many previous works in the technical literature suggest that FPGAs can provide an ideal mix of high-performance, power-efficiency, and flexibility for several classes of applications. In fact, FPGAs offer to general applications a number of advantages compared to both programmable processors and traditional Application Specific Integrated Circuits (ASICs), embracing disparate areas from high-level design and software-like programming [2], [3], [5], [8] to application-specific testing [11]. However, cryptographic processing is an application domain that appears to be particularly well-suited for dedicated circuit-level acceleration [12] mostly because cryptographic algorithms have peculiar characteristics, like integer computation, bit-level manipulations, special data movement patterns, etc., that can be directly translated to a dedicated hardware architecture far more efficiently than software processing on general-purpose platforms. Consequently, special FPGA-based circuits and complex systems have been demonstrated in a variety of cryptographic applications with the aim of improving time performance with large workloads [4], [6], [7] as well as for cryptanalytic purposes [9], [10], [13], [14]. In both cases, one additional advantage FPGAs provide is the possibility to customize the system based on specific instances of the parameters, e.g. a particular numeric value of a cryptographic key without modifying the system at the physical level.

In particular, as a prominent case study, the project will target a current hot-topic in cryptographic research, i.e. homomorphic encryption (HE). In its full form, HE allows an arbitrary funtion to be computed directly on encrypted data, without disclosing the confidential data being processed. Hence, such computation can be delegated to an untrusted third party without it gaining any knowledge of the plaintext. Clearly, such scenario is particularly desirable in Cloud Computing, where security is a major concern. More specifically, homomorphic encryption can act as an enabling cryptographic tool for a number of different application scenarios, including: *multiparty computation*, where several parties are interested in computing a common, public function on their inputs while keeping their individual inputs private; *financial applications*, where both the data and the function to be computed on the data is private and proprietary, e.g. data about corporations, their stock price or their performance is often relevant to making investment decisions, and functions which do such computations may be proprietary, based on predictive models for stock price performance identified by financial analysts as part of their business; *medical applications*, where highly confidential medical records of patients are encrypted by the healthcare providers before being uploaded to a cloud system doing computation on the encrypted data on behalf of the patient for monitoring, alerting, etc; *electronic voting* provided over the Internet, where voters can participate from any location letting the voting authorities to calculate the tally without decrypting any of the individual ballots.

The first fully homomorphic encryption (FHE) scheme was presented in 2009 by the breakthrough work of Craig Gentry. Since the introduction of Gentry's scheme, based on a few properties of ideal lattices, various alternative solutions have been proposed, the most relevant being the van Dijk, Gentry, Halevi and Vaikuntanathan's (DGHV) scheme over the integers, and the Brakerski and Vaikuntanathan's scheme [15] based on the Learning with Errors (LWE) and Ring Learning with Errors (RLWE) problems [16].

Despite its great potential, homomorphic encryption suffers from a high computational cost, both in terms of time and memory occupancy, which currently prevents its practical use in real-world settings. The main concerns are the very large size of public keys and ciphertexts, the impressively high computing time for encrypting a single bit, as well as the fact that after every few operations on the ciphertext, a re-encryption operation is needed in order to lower the noise to acceptable levels.

The project described in this presentation aims at mitigating these limitations by introducing new solutions for hardware-accelerated homomorphic encryption. The text describes the

Invited presentation for the 3rd International Workshop on Cloud and Distributed System Applications (CADSA-15) state of the art and the context of proposed research in Section II, the methodology in Section III, as well as current state and preliminary results in Section IV, with particular emphasis on the ongoing development of an FPGA-based FFT accelerator specifically conceived for homomorphic encryption.

II. State of the art

Recent trends show that FPGA devices may potentially provide a key role for datacenters and high-performance computing [17], [18], matching the raise of cloud computing, now emerged as an important paradigm shift for disparate applications [19], [20], [24]. On one hand, this calls for new methods to allow general developers to access the potential of hardware acceleration through software-like programming approaches [2], [5], [17], [8], [25]. On the other hand, this may give FPGAs a key role for accelerating cloud processing for compute-intensive tasks like security-related operations. In a future scenario, this potential may perfectly fit the computing demand of homomorphic encryption, which is currently limited by its prohibitive requirements in practical settings.

While several traditional cryptographic primitives support non-complete sets of homomorphic operations, only in 2009 a work by Gentry [21] showed a fully homomorphic encryption scheme. A first implementation of a variant of Gentry's scheme was proposed by Gentry and Halevi [22], while more recent software implementations include [23] and [26]. As of writing this paper, there are also a number of open-source software implementations including *hcrypt* [27] and *HElib* [28]. The acceleration of the homomorphic primitives has been already explored since its introduction. For example, [29] presents a GPU implementation based on an NVIDIA GTX 690. [30] compares a solution based on an Altera Stratix V FPGA with an NVIDIA Tesla C2050 GPU, and shows that the FPGA reaches around a 2X performance improvement with lower power comsumption. [33] proposes an ASIC implementation of long-operand multiplication. A hardware implementation of the cryptographic primitives of the Gentry-Halevi FHE scheme is presented by the same authors [35]. The design includes optimizations previously introduced in [29] to reduce the number of FFT computations.

Most existing implementations focus on optimizing the most time consuming operations used by the various encryption schemes: multiplication and modular reduction. Such operations are performed on very large operands, in the order of millions of bits, and can benefit from better asymptotic algorithms. For example, the work in [36], based on the FHE scheme presented in [38], [39] and implemented on a Xilinx Virtex-7 FPGA, proposes an FFT-based long-integer multiplier along with a Barrett reduction module, reaching a considerable speedup compared to existing software solutions.

III. Methodology

The objective of the project is to define the architecture of an FPGA-based accelerator building on previous research results in the area of cryptography-related operations [32], [31] and computer arithmetic [34] and retargeting them to the specific area of reconfigurable hardware technologies. The starting point of the project is the efficient implementation of the Schönhage-Strassen algorithm (SSA) for the multiplication of large integers, exploiting the properties of the Discrete Fourier Transform over integers, which pays off for operands of at least $100,000$ bits. The most time consuming operation in the SSA is the computation of the FFT (and IFFT). By using the *divide et conquer* approach of the Cooley-Tukey scheme, it can be executed in an $O(N \cdot \log N)$ time, where N is the number of points on which the transform is computed. Instead of a classic binary recursive splitting approach relying on a radix-2 transform, the project uses higher order transforms as the basic block for FFT, namely Radix-16 and Radix-64 FFT, since they can be efficiently computed (choosing an adequate finite field) by using only shifts and additions/subtractions.

Each FFT based multiplier will contain one Radix-64 FFT and one Radix-16 FFT. A 64K-point FFT will be computed in three subsequent stages; the first two requiring 1024 Radix-64 FFTs each, the last using 4096 Radix-64 FFTs. Each stage can be efficiently parallelized, according to the available Processing Elements (PEs) that will be implemented on the FPGA part of the system. The resulting execution time for an FFT is $T_{FFT} = (1024/M) \cdot T_{FFT64} + (1024/M) \cdot T_{FFT64} + (4096/M) \cdot T_{FFT16}$, where M is the number of PEs, T_{FFT64} and T_{FFT16} are the times required by, respectively, Radix-64 and Radix-16 FFTs. The ultra-large result of the multiplication operation (~ 2 Mbits) then needs to be reduced by a modulus, which depends on the cryptographic primitive and key. We rely on Barrett modular reduction, since it ensures less overhead for single multiplications compared to other approaches like Montgomery reduction.

The overall activity leading to the long-term objectives of the project is structured in three phases:

- **Phase 1** aims to develop a dedicated HE accelerator based on next-generation heterogeneous platforms. The activity will rely on the experimental FPGA-based heterogeneous platform, enabling innovative architecture-level approaches such as direct inter-accelerator communication and shared virtual memory. At a higher level, the solution will rely on a hybrid approach, exploiting the heterogeneous resources available on the platform by distributing the computation workload between general-purpose cores and application specific Processing Elements (PEs) implemented on the FPGA as well as relying on dedicated solutions for communication between heterogeneous elements. The array of PEs implemented on the FPGA part will provide a complete set of functional units tuned for the HE primitive, and will be replicated in a highly scalable fashion to maximize performance under a given hardware resource budget. Figure 1 shows the high-level structure of the implemented system. The Data Switch is used mainly by the FFT multiplier for reading operands from memory using a butterfly pattern. A scratchpad memory, made up of FPGA SRAM blocks, is used to buffer at least a full single operand (~ 1 Mbit). The SRAM-based solution allows the fastest memory accesses for our application not only because it resides on chip, but also because the FFT has a non-sequential access pattern (butterfly accesses) which does not match the optimized sequential accesses supported by DDR DRAM.

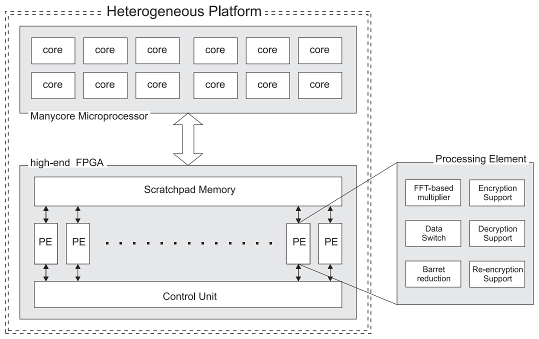

Fig. 1. Overview of the hardware-accelerated platform

The remaining components (Encryption, Decryption, Re-encryption Support) contain functional units like ALUs and coefficient tables, specific to the particular primitive.

- **Phase 2** will build on the outcome of Phase 1 and will define the architecture of a programmable cryptographic acceleration. This "GPU-like" heterogeneous accelerator will be based on innovative mechanisms like direct inter-accelerator communication and virtual shared memory. It will be specifically targeted to the acceleration of cryptographic processing through a suitable mix of dedicated resources and firmware-programmable compute units. The integer-based Homomorphic Encryption will provide a compelling case-study for our flexible cryptographic accelerator, although the implementation of other standard primitives will also be demonstrated.
- **Phase 3** will demonstrate the approach in a real setting. This phase will first develop low-level drivers and software wrappers to access the functions of the accelerator from software, which will be then used to implement a crypto-device component embedded in the open-source OpenSSL toolkit. The project will also consider the integration with existing open-source hypervisor solutions, based on the above low-level library, possibly relying on the accelerator as a sort of hardware-assisted virtualization support for the execution of homomorphically encrypted applications.

IV. Current state

This section briefly summarizes the current state of the activity. As of writing this contribution, we have implemented a first release of the FPGA-based accelerator focused on the FFT computation. While we plan to demonstrate the system on a high-end Altera Stratix V device, currently we are equipped with entry-level Cyclone V boards. Because of this constraint we took a distributed, i.e. multi-board, approach for the implementation of the 64K-point FFT accelerator, based on the use of several processing elements connected in a hypercube topology. In fact, the distributed approach is well-suited for FFT computing, because of the peculiar data exchange pattern. Our current solution is similar to [30], although they use a shared memory approach with local buffers, while we adopted a distributed approach and introduced a few additional optimizations not described in this contribution.

As we explained earlier, instead of a classic binary recursive splitting approach (i.e. a radix-2 transform), the project uses higher order transforms as the basic block for FFT. The core computing element is the Radix-64/16 FFT unit, computing the basic FFT blocks. We make an extensive use of double buffering, since in our distributed scheme we need often to concurrently compute and communicate. The associated memory infrastructure allows 8 read/write operations in parallel with different access patterns for read and write in order to efficiently support the FFT butterfly access pattern. Additionally, we also need a bank of modular multipliers, for twiddle factor multiplications, needed between two consecutive FFT computation stages.

Currently most components of the above processing element have been finalized, particularly the Radix-64 unit, the banked memory, and the modular multiplicators. By synthesizing the design for an Altera's Stratix V, namely a 5SGSMD8N3F45I4 device, we can pack four PEs in a single device and we obtain an operating frequency of 180 MHz, corresponding to an FFT execution time of $T_{FFT} \approx 34.4\mu s$. The processing element is able to perform 8 multiplications simultaneously (with the twiddle factors) and can be reused to perform the component-wise multiplications, taking $T_{MUL} = T_C \cdot 65536/(8 \cdot P) \approx 11.5\mu s$. The resulting time for a complete SSA multiplication is around $150\mu s$.

We highlight that the developed FPGA accelerator is not intended for a specific scheme, as it may support any variant requiring operations on very large operands.

V. Developments and conclusions

This presentation reviewed the main objectives and current results of a research project currently being carried out by the author aimed at the implementation of an FPGA-based cryptographic accelerator to be used for supporting homomorphic encryption. Current results focus on an efficient FPGA-based implementation of large-operand FFT, showing encouraging results in terms of execution time and resource efficiency. On the long term, the work will contribute to developing a high-end dedicated hardware accelerator relieving Homomorphic Encryption form its main limitation, i.e. very prohibitive requirements in terms of computing power.

Acknowledgments

This presentation has been supported by the European Commission in the framework of the H2020-FETHPC-2014 project n. 671668 - *MANGO: exploring Manycore Architectures for Next-GeneratiOn HPC systems*.

Furthermore, the first-release design described above is currently under evaluation for the 2015 Altera Innovate Europe designer contest.

References

[1] Çetin K. Koç, *Cryptographic Engineering*. Springer Science & Business Medi, 2008.

[2] P. Coussy and A. Morawiec, *High-Level Synthesis from Algorithm to Digital Circuit*. Springer, 2008.

[3] A. Cilardo and L. Gallo, "Improving multibank memory access parallelism with lattice-based partitioning," *ACM Transactions on Architecture and Code Optimization*, vol. 11, no. 4, pp. 45:1–45:25, Jan. 2015.

[4] D. Suzuki, “How to maximize the potential of FPGA resources for modular exponentiation,” in *Proceedings of the 9th international workshop on Cryptographic Hardware and Embedded Systems*, ser. LNCS, vol. 4727. Springer, 2007, pp. 272–288.

[5] A. Cilardo, L. Gallo, and N. Mazzocca, “Design space exploration for high-level synthesis of multi-threaded applications,” *Journal of Systems Architecture*, vol. 59, no. 10, pp. 1171–1183, 2013.

[6] G. de Meulenaer, F. Gosset, M. M. de Dormale, and J.-J. Quisquater, “Integer factorization based on elliptic curve method: Towards better exploitation of reconfigurable hardware,” in *Proceedings of the 15th Annual Symposium on Field-Programmable Custom Computing Machines (FCCM) 2007*. IEEE, 2007, pp. 197–206.

[7] S. Ghosh, D. Mukhopadhyay, and D. Roychowdhury, “High speed flexible pairing cryptoprocessor on FPGA platform,” in *Proceedings of the 4th international conference on Pairing-based cryptography*. Springer-Verlag, 2010, pp. 450–466.

[8] A. Cilardo, L. Gallo, A. Mazzeo, and N. Mazzocca, “Efficient and scalable OpenMP-based system-level design,” in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, March 2013, pp. 988–991.

[9] S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, and M. Schimmler, “Breaking ciphers with COPACOBANA - a cost-optimized parallel code breaker,” in *Proceedings of the 8th international conference on Cryptographic Hardware and Embedded Systems (CHES)*. Springer-Verlag, 2006, pp. 101–118.

[10] T. Güneysu, T. Kasper, M. Novotný, C. Paar, and A. Rupp, “Cryptanalysis with COPACOBANA,” *IEEE Transactions on Computers*, vol. 57, no. 11, pp. 1498–1513, 2008.

[11] A. Cilardo, “New techniques and tools for application-dependent testing of FPGA-based components,” *Industrial Informatics, IEEE Transactions on*, vol. 11, no. 1, pp. 94–103, Feb 2015.

[12] ——, “Exploring the potential of threshold logic for cryptography-related operations,” *Computers, IEEE Transactions on*, vol. 60, no. 4, pp. 452–462, April 2011.

[13] (2011, Jan.) Rivyera S3-5000. [Online]. Available: http://www.sciengines.com/

[14] A. Cilardo and N. Mazzocca, “Exploiting vulnerabilities in cryptographic hash functions based on reconfigurable hardware,” *Information Forensics and Security, IEEE Transactions on*, vol. 8, no. 5, pp. 810–820, May 2013.

[15] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, “Fully homomorphic encryption over the integers,” *EUROCRYPT*, pp. 24–43, 2010.

[16] Z. Brakerski and V. Vaikuntanathan, “Fully homomorphic encryption from ring-lwe and security for key dependent messages,” *CRYPTO*, pp. 505–524, 2011.

[17] K. Paranjape, S. Hebert, and B. Masson, “Heterogeneous computing in the cloud: Crunching big data and democratizing HPC access for the life sciences,” 2010.

[18] A. Putnam and other, “A reconfigurable fabric for accelerating large-scale datacenter services,” in *41st Annual International Symposium on Computer Architecture (ISCA)*, June 2014. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=212001

[19] Y. K. Sinjilawi, M. Q. AL-Nabhan, and E. A. Abu-Shanab, “Addressing security and privacy issues in cloud computing,” *Journal of Emerging Technologies in Web Intelligence*, vol. 5, no. 2, pp. 192–199, 2014.

[20] F. Amato, A. Mazzeo, V. Moscato, and A. Picariello, “A framework for semantic interoperability over the cloud,” in *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*. IEEE, 2013, pp. 1259–1264.

[21] C. Gentry, “A fully homomorphic encryption scheme,” Ph.D. dissertation, Stanford University, 2009.

[22] C. Gentry and S. Halevi, “Implementing gentry’s fully-homomorphic encryption scheme,” *Proc. Advances in Cryptology-EUROCRYPT 2011*, pp. 129–148, 2011.

[23] C. Gentry, S. Halevi, and N. P. Smart, “Homomorphic evaluation of the aes circuit,” *IACR Cryptology ePrint Archive*, vol. 2012, p. 99, 2012.

[24] F. Amato, A. Chianese, V. Moscato, A. Picariello, and G. Sperli, “Snops: a smart environment for cultural heritage applications,” in *Proceedings of the twelfth international workshop on Web information and data management*. ACM, 2012, pp. 49–56.

[25] A. Cilardo, D. Socci, and N. Mazzocca, “ASP-based optimized mapping in a Simulink-to-MPSoC design flow,” *Journal of Systems Architecture*, vol. 60, no. 1, pp. 108 – 118, 2014.

[26] J. Coron, T. Lepoint, and M. Tibouchi, “Batch fully homomorphic encryption over the integers,” *IACR Cryptology ePrint Archive*, vol. 2013, p. 36, 2013.

[27] H. Perl, M. Brenner, and M. Smith. (2011) hcrypt. [Online]. Available: http://www.hcrypt.com/scarab-library/

[28] S. Halevi and V. Shoup. (2012) Helib, homomorphic encryption library. [Online]. Available: https://github.com/shaih/HElib

[29] W. Wang, Y. Hu, L. Chen, X. Huang, and B. Sunar, “Exploring the feasibility of fully homomorphic encryption,” vol. 99, p. 1, 2013.

[30] W. Wang and X. Huang, “Fpga implementation of a large-number multiplier for fully homomorphic encryption,” *ISCAS*, pp. 2589–2592, 2013.

[31] A. Cilardo, “Modular inversion based on digit-level speculative addition,” *Electronics Letters*, vol. 49, no. 25, pp. 1609–1610, December 2013.

[32] ——, “Efficient bit-parallel GF(2^m) multiplier for a large class of irreducible pentanomials,” *Computers, IEEE Transactions on*, vol. 58, no. 7, pp. 1001–1008, July 2009.

[33] Y. Doröz, E. Öztürk, and B. Sunar, “Evaluating the hardware performance of a million-bit multiplier,” *Digital System Design (DSD),16th Euromicro Conference on*, 2013.

[34] A. Cilardo, D. De Caro, N. Petra, F. Caserta, N. Mazzocca, E. Napoli, and A. Strollo, “High speed speculative multipliers based on speculative carry-save tree,” *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 61, no. 12, pp. 3426–3435, Dec 2014.

[35] Y. Doröz, E. öztürk, and B. Sunar, “Accelerating fully homomorphic encryption in hardware,” *draft, Under Review*, 2013.

[36] X. Cao, C. Moore, M. O’Neill, N. Hanley, and E. O’Sullivan, “High speed fully homomorphic encryption over the integers,” *Workshop on Applied Homomorphic Cryptography, to appear*, 2014.

[37] A. Cilardo, E. Fusella, L. Gallo, and A. Mazzeo, “Exploiting concurrency for the automated synthesis of MPSoC interconnects,” *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 14, no. 3, p. 57, 2015.

[38] J. Coron, A. Mandal, D. Naccache, and M. Tibouchi, “Fully homomorphic encryption over the integers with shorter public keys,” *CRYPTO*, pp. 487–504, 2011.

[39] J. Coron, D. Naccache, and M. Tibouchi, “Public key compression and modulus switching for fully homomorphic encryption over the integers,” *EUROCRYPT*, pp. 446–464, 2012.