Constructing Cluster of Simple FPGA boards for Cryptologic Computations

Yarkin Doröz and Erkay Savaş

Sabanci University
Istanbul, Turkey
{yarkin, erkays}@sabanciuniv.edu

Abstract. In this paper, we propose an FPGA cluster infrastructure, which can be utilized in implementing cryptanalytic attacks and accelerating cryptographic operations. The cluster can be formed using simple and inexpensive, off-the-shelf FPGA boards featuring an FPGA device, local storage, CPLD, and network connection. Forming the cluster is simple and no effort for the hardware development is needed except for the hardware design for the actual computation. Using a softcore processor on FPGA, we are able to configure FPGA devices dynamically and change their configuration on the fly from a remote computer. The softcore on FPGA can execute relatively complicated programs for mundane tasks unworthy of FPGA resources. Finally, we propose and implement a fast and efficient dynamic *configuration switch technique* that is shown to be useful especially in cryptanalytic applications. Our infrastructure provides a cost-effective alternative for formerly proposed cryptanalytic engines based on FPGA devices.

1 Introduction

Cryptographic operations usually contain high degree of parallelism, which favors repetitive instantiation of the same basic block for the cryptographic primitives. Thus, hardware-based cryptographic accelerators, harnessing the aforementioned parallelism, have become the focus of both industrial and academical interests in the last two decades.

Cryptanalytic studies aim to discover the strength of cryptographic algorithms against certain attack techniques, efficiency of which is determined by, to a large extent, amount of computational power available at affordable costs. As it is possible to make relatively accurate predictions (at least so far) for the increase in computational power and decrease in their associate costs in future (e.g. Moore's Law), we can provide some predictions for the future strength of certain cryptographic algorithms and their key lengths. Moreover, since increase in raw computational power does not necessarily lead to the same level of increase in our capacity for breaking ciphers, it is important to work on new architectures that will make an efficient use of the new *computing capabilities*.

Recent developments in FPGA technology, in terms of increased resources and declining costs, emphasize the configurable logic devices as the economic alternative for both cryptographic acceleration and cryptanalytic computations. Grasping this great potential, previous works in literature propose FPGA-based designs and architectures for both cryptographic acceleration [1, 2] and cryptanalytic purposes [3, 4, 5].

Nowadays, many FPGAs can be configured to implement microprocessor cores that can handle mundane tasks, which are not performance bound and unworthy of valuable

FPGA resources (e.g. TCP/IP communication). MicroBlaze, which is a soft processor core (*softcore* henceforth) by Xilinx and can be implemented even on inexpensive FPGAs using the reconfigurable logic of FPGAs [6], can be utilized in this context.

In addition, FPGAs can be *dynamically* configured to implement multiple hardware designs. Relatively fast dynamical switching between configurations provides agility as well as flexibility to meet computational diversity of cryptologic applications. Moreover, the configuration files for multiple designs can be sent over a network.

This work uses Spartan-3E Starter Kit to form a so-called *FPGA cluster* for cryptologic purposes. Our approach differs from similar and the closest works in [3, 4, 5] in the sense that a super computer does from a server cluster. Our FPGA cluster can be formed using a host-PC acting as the *cluster head* and any off-the-shelf FPGA board featuring an FPGA, a network interface, local storage, and a simple CPLD.

The proposed cluster can be efficiently used for cryptanalytic purposes (e.g. exhaustive search). For certain cases, it can also be used as an accelerator to speedup the cryptographic applications. By supporting a fast, dynamic configuration switch, each FPGA board can combine the versatility of general-purpose computer with the parallel computing capability of hardware designs, even for FPGAs in the low-end of the cost spectrum. The software components we develop and denote as proxies running both in the cluster head and the softcore enables a transparent programing experience similar to the one provided by middleware for parallel programing and remote procedure call.

Outline of the paper is as follows. In Section 2, we introduce the architecture of the proposed FPGA cluster, employed FPGA boards and its components. We explain operational steps adopted for FPGA device and dynamic configuration switch technique in Section 3. Section 4 provides the details of the usage of the proposed cluster for cryptographic acceleration and cryptanalysis. Implementation details and experimental results are provided in Section 5. Finally, Section 6 concludes the paper.

2 Proposed Architecture for FPGA cluster

The architectural overview of the FPGA cluster we use in our work is depicted in Figure 1. Since our architecture uses TCP/IP for communication, any FPGA board connected to the Internet can be a part of our cluster and individually accessed from anywhere in the network.

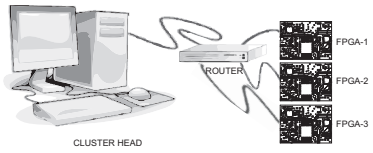

Fig. 1. General overview of the FPGA cluster

As pointed out earlier, our goal is to harness especially the computation power of inexpensive FPGA boards for the use in cryptologic applications. Therefore, we use Spartan-3E Starter Kit board, which is one of the basic equipment used in logic design courses. A Spartan-3E Starter Kit [7] is a board that consists of the following hardware components: i) volatile programmable unit (FPGA - XC3S500E), ii) a nonvolatile programmable unit (CPLD - XC2C64A), iii) 128 Mbit parallel flash memory, iv) 64 MB

DDR SDRAM (MT46V32M16), v) Standard Microsystems LAN83C185 10/100 Ethernet physical layer (PHY) interface and vi) a RJ-45 connector. In Figure 2, we show how we utilize those components in our cluster.

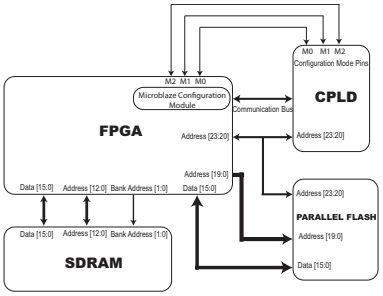

Fig. 2. Components of the FPGA board

For flexibity and transparency purposes, the proposed infrastructure is designed as a self-configuring system, which becomes ready for remote configuration once it is connected to the Internet. In addition, *runtime reconfigurability* allows to switch between 16 different configurations, which is controlled by the CPLD, dynamically.

3 Our Scheme and Its Operational Steps

Our scheme can be understood better if the following four key steps of its operation are explained in detail as follows:

Softcore Configuration: A state machine implemented in the non-volatile CPLD configures FPGA automatically using the configuration bit stream of the softcore stored in the parallel flash, when the device is turned on. Following the softcore configuration, a special program called *boot-loader* is executed by the softcore.

Execution of Boot-loader: The boot-loader is a small piece of code, which comes as a part of configuration file of the softcore and stored in internal Block RAM (BRAM) of the FPGA. It is responsible of moving the proxy code[1] from the parallel flash to the SDRAM since the latter is too large to fit in the internal BRAM.

Execution of Proxy for Implementing Client/Server Communication Model: To assign tasks, the cluster head communicates with FPGA boards using reliable TCP/IP protocols which are implemented by the proxy code on the softcore side. In our communication model, the cluster head and the softcore plays the roles of server and clients interchangeably.

Automatic, Remote Configuration of FPGA Device and Configuration Switch: The actual computations for specific tasks are performed by hardware implementations, optimized for the FPGA devices. Once the configuration file for a hardware implementation is available, the cluster head can send it through network to the FPGA devices. The proxy code running on the softcore is responsible of receiving the hardware configuration file and storing it in the parallel flash.

The configuration switch necessitates the execution of the following steps in this order: **i) Communication 1:** Input data is sent to the softcore and stored in parallel

[1] The code of the main software application which is used to communicate with the cluster head.

flash, **ii) Configuration switch 1:** Softcore removes itself from the FPGA and loads the hardware, **iii) Computation:** The hardware works on the task, and writes the results in the parallel flash, **iv) Configuration switch 2:** The hardware is removed and the softcore is loaded in the FPGA, **v) Communication 2:** The softcore reads the results from the parallel flash, and sends them to the cluster head.

Alternatively, especially for applications and FPGAs where the area overhead of the softcore is not important, the hardware design and the softcore can run simultaneously in FPGAs, which eliminates the need for configuration switch as described above. In subsequent sections, we give applications that benefit from the configuration switch.

4 Using FPGA Cluster for Acceleration of Cryptographic Computations and Cryptanalysis

A simple and inexpensive FPGA device such as Spartan-3 running at a low clock frequency of 119 MHz can perform an RSA exponentiation operation in about 8 ms using 1553 slices and 10 hardwired multipliers [1][2]. Similarly, the same FPGA device can achieve an encryption rate of 429 Mbps for the AES standard block cipher algorithm using only 103 slices at 161 MHz [8]. Since the FPGA device can realize more than one block of AES encryption engine, it is possible to reach much higher throughput values for encryption operation either using multi-message encryption techniques or a suitable working mode (e.g. counter mode). Therefore, using simple, inexpensive FPGA clusters can be cost-effective alternatives for accelerating cryptographic operations. However, cryptographic acceleration through a simple FPGA board may not be feasible if configuration switching is needed[3]. In Section 5, we provide a scenario where cryptographic acceleration may be possible even in the case of configuration switch. But, block cipher acceleration is always possible since the FPGA device can be shared between the softcore and the hardware.

Most cryptanalytic algorithms can be adjusted to alleviate the time overhead incurred in inter-process communication between the cluster head and the FPGA boards. Both designs in [4] for exhaustive key search and [5] for solving discrete logarithm problem (DLP) rely on a massively parallel computer of inexpensive FPGA devices as the computational work horse. Also, as stated in [5], certain computations in the Pollard's Rho method [9, 10] for solving elliptic curve DLP (ECDLP) can be so adjusted to meet any bandwidth restriction between the cluster head and the FPGA boards.

For instance, in an exhaustive search for a AES key using the implementation in [8], one AES block (103 slices) can try approximately 3.3 million key candidates in one second. In a single computation task submitted to an FPGA, which takes about one minute, one additional AES block implemented on the FPGA resources gained

[2] Note that RSA timings for one signature operation vary between 0.15 ms and 8 ms on a PC depending on the processor (cf. http://bench.cr.yp.to/results-sign.html). In order to obtain acceleration over common PCs, a larger FPGA device that can accommodate more than one instance of crypto unit should be used. Otherwise, many FPGA boards will be needed to outperform PC implementations.

[3] Because of large resource consumptions some hardware designs cannot co-exist with the softcore (e.g. RSA).

by removing the softcore can try out an extra 200 million key candidates. This value commensurates with the number of AES instances that can fit in the space saved through removing the softcore. Since the communication between the cluster head and FPGA device is not intense (in fact only the key interval is needed to be communicated to the FPGA), overlapping communication and computation would not help. Therefore, in such cases it is always beneficial to apply the configuration switch.

5 Implementation and Experimental Results

In this section, we provide some implementation details and experimental results to evaluate the true potential of the proposed FPGA cluster for cryptologic computations. We start with the resources needed to implement the softcore in Spartan-3E (XC3S500E) device, which consumes 4,270 out of 9,312 (45%) 4-input LUTs and occupies 3,526 out of 4,656 (75%) slices. The utilization percentage for such a small FPGA device is relatively high and leaving limited configurable FPGA resources for hardware unit that will perform the actual computation. This is, in fact, one of the primary motivation for the scheme that will allow an efficient configuration switch between the softcore and the hardware unit.

In our experiments, we used a Linux-based PC (cluster head) and ASUS RT-N13U router in addition to three Spartan-3E Starter Kit boards. We used Verilog for all hardware designs and C/C++ language for software components on the softcore and the cluster head. The first experiment is intended to find out the efficiency of using the FPGA boards mainly for cryptographic acceleration if the hardware unit and the softcore cannot co-exist in the FPGA board and therefore, configuration switch is necessary. The experiment is performed as follows: the cluster head sends input values (e.g. messages to be encrypted or signed) to the FPGA device which are written in the parallel flash thereafter. After the transmission is completed, a configuration switch command is sent to the FPGA. Since we are interested only in the overhead the whole process creates, the hardware unit reads the data from the parallel flash first and then writes it back to it. Without doing anything else, it switches immediately back to the softcore configuration and the softcore sends the written data back to the cluster head.

The data exchanged between the cluster head and the FPGA are sent in different packet sizes (i.e. sizes of send/receive buffers in both sides). The timing values obtained through averaging for the first experiment are enumerated in rows 2-5 of Table 1. Timing overhead for handling 1 MB of data is about 27.31 s, on average. 1 MB of data, for example, means 8192 RSA operation (e.g. signature), where the modulus is 1024 bit. This results in an overhead of 3.33 ms per 1024-bit RSA operation. Considering that the state-of-the-art implementation of RSA for Spartan-3E in [1] executes the same operation in about 8 ms, this increases the effective time per RSA operation roughly by 37% percent, on average. Note that this overhead would be about 0.87 s for 1 KB data size, which is definitely not an acceptable performance for a cryptographic accelerator. In summary, our FPGA cluster may be useful in case the configuration switch is necessary only when we are able to group the input data in large chunks.

Packet Size (B)	Storage Device	*1 KB*	*10 KB*	*100 KB*	*1 MB*
256	Parallel Flash	15.74	17.38	32.70	216.81
512	Parallel Flash	14.40	15.50	21.11	102.90
1024	Parallel Flash	13.65	14.33	16.77	72.73
1024(opt.[4])	Parallel Flash	6.96	7.44	8.79	27.31
1024(opt.[4])	SDRAM	0.27	0.35	1.06	7.81

Table 1. Timing overhead (in seconds) for different data and packet sizes

In the second experiment, we measured the time to send and receive data of different sizes when configuration switch is not needed for the scenario where the cryptographic unit and the softcore fit in the FPGA device. The SDRAM can be used to store the data since there is no configuration switch that causes the SDRAM to miss refreshment cycles. The timing values for the second experiment are enumerated in the last row of Table 1 only for buffer size of 1024 B. As can be observed from the table, using the SDRAM rather than the parallel flash, decreases the total time by 19.50 s for 1 MB of data. However, the timing values in the last row should not really be considered as actual overhead since operations for sending/receiving data and writing/reading to/from the SDRAM can be overlapped with the actual cryptographic computation.

In the third experiment, we tried to measure the total overhead time when the data transfer is not intense and configuration switch can be used, which is typical mostly for cryptanalytic purposes. The cluster head sends a message of 32 B to the softcore, which contains the task description as well as the input parameters. After the task description and input parameters are received, two configuration switch operations occur to configure FPGA first with the hardware than with the softcore. We performed all steps except for the actual computation time of the task to determine the overhead in time. The timing results for one, two, and three FPGA boards are measured as 7.06 s, 7.09 s, and 7.14 s, respectively. These results show that we can multiply our computational power with minor overheads in time.

In the next experiment, we performed an exhaustive key search for PRESENT algorithm [11], which is a lightweight block cipher intended for embedded applications. The results for a single FPGA board are listed in Table 2. The maximum number of encryption engines that will fit in Spartan-3E is only 13.With configuration switch and communication costs included, we are able to test about 928 million keys in 61.76 s. The last row enumerates the experimental results when there is no configuration switch and the softcore and seven encryption engines run concurrently. This experiment demonstrate the advantage of configuration switch for exhaustive key search applications. Note that speed optimized, single-threaded C implementation of the PRESENT algorithm (cf. http://www.lightweightcrypto.org/present/) on a PC with an AMD 3.2 GHz quad-core processor and 4 GB RAM can try roughly 106 million keys in 62 s, which also demonstrates that acceleration of PRESENT algorithm is possible.

Finally, we implemented Pollard's Rho method [9, 10] to compute discrete logarithms in elliptic curves over prime fields of odd characteristics. For elliptic curve arithmetic we used Huff model to take advantage of the fast explicit formulae for point ad-

[4] We optimized the execution times by adding cache to MicroBlaze and performing improvements in the software.

Conf.	Area LUT + Slice	Max/Usable Freq. (MHz)	no of keys tried in $\approx 60s$
Single PRESENT	$3\% + 3\%$	187.37/NA	NA
13 PRESENT	$74\% + 99\%$	65.23/50	928,628,190 in 61.76 s
7 PRESENT + Softcore	$83\% + 99\%$	53.709/50	510,656,511 in 60.10 s

Table 2. Experimental results for exhaustive key search

no of FPGA boards	no of runs	avg.	med.	max.	min.	stdev
3 (Spartan-3E500)[5]	36	927	849	2368	232	531
7 (Spartan-3E500)[5]	27	478	442	1074	223	216
5 (Spartan-3E500)+2(Spartan-3E1600)[6]	30	302	236	687	192	123

Table 3. Timing statistics (in seconds) for Pollard Rho's alg. on different number of FPGA boards

ditions on Huff curves [12]. The FPGA boards are used to find the distinguished points, which constitutes the most time-consuming part of the computation in Pollard's Rho method. Since a similar approach to the one in [4] is adopted, we only implemented point addition. The hardware implementation of the circuit to find the distinguished points (i.e. distinguished points-generating engine) consumes 50% of the total LUTs, 19% of slice flip-flops, and 5% of the block RAMs. In the experiments, we used an elliptic curve over a prime field where the prime is a 160-bit integer. The base point order is chosen as a 50-bit integer to demonstrate that the discrete logarithm can be computed within a reasonable amount of time using several FPGA boards. A single-threaded PC implementation on an AMD 3.2 GHz quad-core processor with 4 GB RAM completes the same task in about 6 minutes, on average, using NTL package [13].

In order to demonstrate that the time performance of the attack improves linearly with the number of FPGA boards (and the total number of distinguished point-generating engines), we conducted several experiments. Firstly, we optimized the distinguished point-generating engine to fit two instances of it in one FPGA device. Secondly, we employed different number of FPGA boards in our experiments. Using the same curve and the base point mentioned above, we solved different number of elliptic curve discrete logarithm problems (cf. the second column of Table 3) and enumerated the timing statistics in Table 3. As can be observed from the table, we can solve one elliptic curve discrete logarithm problem in about 5.03 minutes using seven FPGA boards (i.e. 22 instances of distinguished point-generating engine), on average and also outperform the PC based implementation[7].

6 Conclusion

The experiments demonstrate that the proposed FPGA cluster can be useful for both cryptographic acceleration and implementing cryptanalytic attacks. Dynamic configuration switch between the soft processor core and the hardware unit, proposed as among the foremost contributions of this work, proves to be useful especially in exhaustive search applications in cryptanalysis, where the need for interprocess communication is

[5] In these experiments, we performed the attack using three and seven boards of Spartan-3E500, each of which can fit two instances of distinguished points-generating engine.

[6] In this experiment, we performed the attack using two Spartan-3E1600, which can fit six instances of distinguished points-generating engine, along with five boards of Spartan-3E500.

[7] We used Asus GIGAX1008B 8 Port 10/100 Layer2 Switch to connect seven FGPA boards.

very limited (if not absent). Dynamic configuration switch can be useful even for more powerful FPGA devices since FPGA resources salvaged from the softcore can be put into effective use.

The proposed FPGA cluster offers advantages over PC-based implementations, when a single FPGA device can accommodate as many instances of the main computation unit as possible to take advantage of the parallelism the hardware implementations offer. While exhaustive search for simple algorithms such as PRESENT can be substantially accelerated, relatively heavy-weight algorithms such as RSA does not benefit from the cluster if only one instance of RSA circuit is implemented in one FPGA device. For acceleration of heavy-weight algorithms, either more advanced FPGA devices or a multitude of simple FPGA devices should be used. Naturally, price performance analysis of the FPGA cluster must be performed on the basis of the specific operation we are trying to accelerate.

References

1. Öksüzoglu, E., Savas, E.: Parametric, secure and compact implementation of rsa on fpga. In: Proceedings of the 2008 International Conference on Reconfigurable Computing and FPGAs, Washington, DC, USA, IEEE Computer Society (2008) 391–396
2. Le Masle, A., Luk, W., Eldredge, J., Carver, K.: Parametric encryption hardware design. In Sirisuk, P., Morgan, F., El-Ghazawi, T., Amano, H., eds.: Reconfigurable Computing: Architectures, Tools and Applications. Volume 5992 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2010) 68–79 10.1007/978-3-642-12133-39.
3. Kumar, S., Paar, C., Pelzl, J., Pfeiffer, G., Schimmler, M.: Copacobana a cost-optimized special-purpose hardware for code-breaking. In: FCCM, IEEE Computer Society (2006) 311–312
4. Güneysu, T., Paar, C., Pelzl, J.: Special-purpose hardware for solving the elliptic curve discrete logarithm problem. TRETS **1** (2008)
5. Güneysu, T., Paar, C., Pfeiffer, G., Schimmler, M.: Enhancing copacobana for advanced applications in cryptography and cryptanalysis. In: FPL, IEEE (2008) 675–678
6. Xilinx: MicroBlaze Soft Processor Core. (2011) `http://www.xilinx.com/tools/microblaze.htm`.
7. Xilinx: Spartan-3E Starter Kit. (2011) `http://www.xilinx.com/products/devkits/HW-SPAR3E-SK-US-G.htm`.
8. Helion: High Performance AES (Rijndael) cores for Xilinx FPGA. (2011) `http://www.heliontech.com/aes.htm`.
9. Pollard, J.M.: Monte carlo methods for index computation $(\mathrm{mod}\ p)$. Mathematics of Computation **32** (1978) pp. 918–924
10. Oorschot, P.C.V., Wiener, M.J.: Parallel collision search with cryptanalytic applications. Journal of Cryptology **12** (1996) 1–28
11. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In Paillier, P., Verbauwhede, I., eds.: CHES. Volume 4727 of Lecture Notes in Computer Science., Springer (2007) 450–466
12. Joye, M., Tibouchi, M., Vergnaud, D.: Huff's model for elliptic curves. Cryptology ePrint Archive, Report 2010/383 (2010) `http://eprint.iacr.org/`.
13. Shoup, V.: NTL: a library for doing number theory. Online: last accessed (2011) `http://www.shoup.net/ntl/`.

This article was processed using the LaTeX macro package with LLNCS style