2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) | 979-8-3503-1199-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/IPDPSW59300.2023.00031

Application-specific FPGAs: cryptographic agility through customized reconfigurable architectures

Jelle Biesmans*, Francesco Regazzoni†‡ and Nele Mentens*§
* ES&S & imec-COSIC, ESAT, KU Leuven, Diepenbeek, Belgium
† Univeristy of Amsterdam, The Netherlands
‡ Università della Svizzera italiana, Switzerland
§ LIACS, Leiden University, Leiden, The Netherlands

***Abstract*—Cryptography is an important feature in modern devices, especially in Internet-of-Things (IoT) networks. When a deployed cryptographic algorithm is not secure anymore, then devices that use it are immediately vulnerable to attacks and secret information can be exposed or devices themselves could be compromised. Because of the widespread need of cryptography, several application domains would certainly benefit from the use a small embedded Field-Programmable Gate Array (eFPGA) for cryptography. To this end, Mentens et al. proposed an eFPGA tailored towards symmetric-key algorithms and designed to be more resource-efficient than general-purpose Lookup Table (LUT) based FPGA structures for the configuration of symmetric-key algorithms, while showing a similar performance.**

In this paper, we extend that earlier work (that was mostly focused on the architectural aspect of the eFPGA) and we complete it considering also the routing overhead of the proposed solution. We use an open-source design flow to estimate the potential of cryptography-tailored eFPGA structures and compare them with classical LUT-based FPGAs. With our design flow, that we describe in details in the paper, we are able to precisely estimate the overhead due to the routing. Our results confirm that for symmetric ciphers, just like for many other applications, the routing overhead is predominant. The methodology we followed for estimation and the precise estimation of the routing overhead that we provide are a fundamental information for the future exploration of cryptography-tailored eFPGAs.

I. Introduction

A commonly used classification of reconfigurable architectures, divides them based on their granularity. Coarse-grained architectures typically consist of larger configurable blocks, like Arithmetic Logic Units (ALUs), multiply-accumulate blocks, memory blocks and other application-specific components. Fine-grained architectures are typically made of regular structures of smaller configurable gates with segmented interconnect. Application-specific coarse-grained reconfigurable architectures have been explored extensively in the past decades for applications such as security, signal processing, image processing and deep learning [1][2]. Fine-grained architectures are mainly developed and optimized by FPGA vendors for general-purpose computing based on configurable Lookup Tables (LUTs). Performance improvement is achieved by adding larger dedicated components like Digital Signal Processing (DSP) slices and embedded memory blocks to the configurable fine-grained sea of gates.

This work is supported by CyberSecurity Research Flanders with reference number VR20192203.

One area in which customized reconfigurable architectures are of extreme interest, is cryptography. Almost every modern system has dedicated hardware to compute one or more cryptographic algorithms. These accelerators are useful only as long as the algorithm, and the associated parameters, are secure. When a microcontroller is present, a software routine could be used to implement a new algorithm. A better solution would be the use of configurable hardware, which combines the efficiency of hardware with the flexibility of software. This route has already been explored in the past, focusing mainly on coarse-grained, configurable architectures tailored for cryptography [3][4]. These previously proposed architectures come with a specific design flow, which however, is targeted towards the specific architecture to which it is associated, and do not support, at least immediately, a systematic exploration of the other architectures.

To support the development of novel design flows and algorithms for FPGAs and ASICs, a number of open-source tools have been proposed in the past decades. Examples are Yosys [5], Verilog-To-Routing [6], Symbiflow [7], OpenFPGA [8] and FABulous [9]. These tools have been designed having general-purpose FPGAs in mind and, for that purpose, they have been shown to be very useful for the exploration of improved architectures and placement-and-routing algorithms. General-purpose FPGAs might not be the only approach to provide reconfigurability to hardware. With many applications requiring only limited reconfiguration capabilities, it can be envisioned that in the future, a number of application-specific reconfigurable architectures, very likely in the form of embedded FPGAs (eFPGAs), will be needed. These application-specific FPGAs, however, can be practically used only if an appropriate design flow can support engineers during the development phase.

In this paper, we explore application-specific FPGAs tailored towards cryptography. Specifically we explore the routing overhead of the cFA (configurable Full Adder) cell, which was proposed by Mentens et al. in [10] to replace LUTs in FPGAs for cryptographic applications and we discuss how the lesson learned could affect the future design of custom FPGAs. Despite our use case is cryptography, the insights gained are general enough to further explore application specific FPGAs also in other domains. More in details, The contributions of this paper are the following:

979-8-3503-1199-0/23/$31.00 ©2023 IEEE
DOI 10.1109/IPDPSW59300.2023.00031

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on August 07,2023 at 10:43:24 UTC from IEEE Xplore. Restrictions apply.

- We evaluate the routing overhead of the cFA cell for several designs
- Based on the results of the cFA cell, we design a new type of cell that improves the initial one.
- We analyse in depth the cFA structure to have a better understanding on the design of specialised cells.

The paper is organized as follows. Section II gives background information on application-specific FPGA structures. Section III describes our experimental setup including the toolchain, the benchmarked cryptographic algorithms, the method to calculate the cell area, the timing and the bitstream size of the circuits mapped on the custom FPGA, and the architectures of the FPGA structures considered in the experiment. Section IV presents the evaluation results. Finally, Section V analyzes the limitations of the cFA cell. Section VI provides a link to the open-source artifacts and Section VII concludes the paper.

II. Background

Fine-grained customized configurable architectures, consisting of logic blocks, are more limited in functionality than general-purpose LUTs but could be optimized to achieve higher performance, area efficiency and/or power efficiency. Because of this, fine-grain configurable blocks are an appealing starting point for improving the the efficiency of a FPGA architectures designed for specific application domains.

For cryptographic configurable circuits, a new type of cell has been proposed by Mentens et al. in [10] to replace traditional LUTs. The new cell is called a cFA (configurable Full Adder). It can be configured to eight different functions with six inputs and two outputs, as opposed to 2^{64} different functions in a 6-2 LUT. Synthesis results show that a cFA-based FPGA features a reduced logic area and bitstream size with a comparable performance for a number of benchmark lightweight cryptographic circuits. Figure 1 shows the design of the cFA cell of Mentens et al.

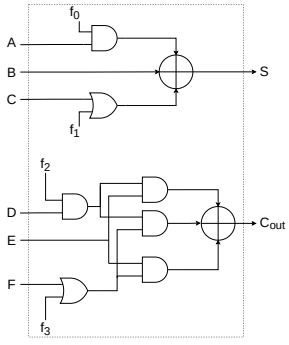


Fig. 1. Configurable Full Adder (cFA) cell with 4 configuration bits (f_0 to f_3)

The cFA cell consists of two separate parts, each configurable in four modes. These modes represent operations which are commonly used in cryptographic algorithms.

III. Experimental Setup

A. Our open-source design flow

Figure 2 shows our tool flow and the generated (intermediate) files. The user design to be mapped on the custom configurable architecture needs to be in Verilog or VHDL. VHDL code gets converted to Verilog with the GHDL as Yosys plugin. We make use of Yosys for synthesis and VPR for placement and routing of the FPGA.

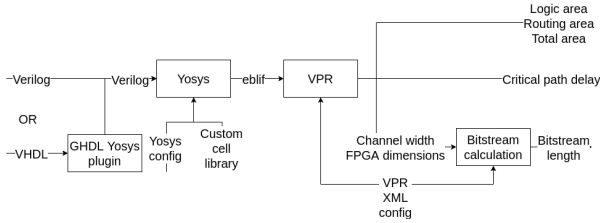


Fig. 2. Design flow

B. Measuring area and timing

To calculate the area and timing of a design synthesized for a custom FPGA architecture, it is required to know the area of the configurable cells and the timing of all input-output paths in the cell. We used Synopsys Design Compiler to synthesise the design of our CLB and report the timing and area. These results are then put in the VPR XML configuration file. This way VPR can calculate the timing and area information for the FPGA loaded with a particular algorithm. The calculation of the area does not take into account the area required to store the bitstream.

C. Determining the bitstream size

The bitstream length is an important criterion when comparing general-purpose and customized FPGA architectures. This is because the configuration bits determine the required configuration memory size and thus the cost of the FPGA. For this research we used a spreadsheet to estimate the bitstream size, because the tools we used don't have support to calculate the bitstream automatically for our architecture.

D. Benchmarked algorithms

We use the following cryptographic algorithms in our evaluation: ACE [11], AES-256 [12], Grain-128AEAD [13], Noekeon [14], Speck [15], Subterranean 2.0 [16], WAGE [17], Keccak/SHA-3 [18] and a basic RISC-V Rocket chip [19]. as an example of non-cryptographic design.

E. Target configurable architectures

1) LUT-based Architecture: The commercial LUT-based FPGA used for comparison is inspired by Xilinx FPGAs. It uses a 6-input, 2-output fracturable LUT. This enables the LUT to be used as a 6-input 1-output LUT or as two 5-input 2-output LUTs, where both 5-input LUTs have the same inputs. Both the outputs of the fracturable LUT can optionally be routed through a flip-flop with the help of a multiplexer. There are 4 slices inside one CLB in all architectures.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on August 07,2023 at 10:43:24 UTC from IEEE Xplore. Restrictions apply.

2) cFA-based Architecture : The cFA cell has the same number of inputs and outputs as a 6-2 fracturable LUT which is 6 inputs and 2 outputs. The cell has 4 configuration bits. Here both outputs of the cFA can optionally be routed through a Flip-flop with the help of a multiplexer.

F. Newly proposed cell

Figure 3 shows the diagram of the cell that is newly proposed in this paper. The most important feature is that both sides are very similar. They both have AND, OR and EXOR gates, with optional inversion.

Only one side has a multiplexer and the select pin is the B1 pin from the other side. When this is used, the B1 pin can still be combined with A1 for the normal gates or the A1 pin can be buffered or inverted. The buffer function is also required to connect to the input of the flip-flop directly from the output of another flip-flop if multiple flip-flops need to be put in series.

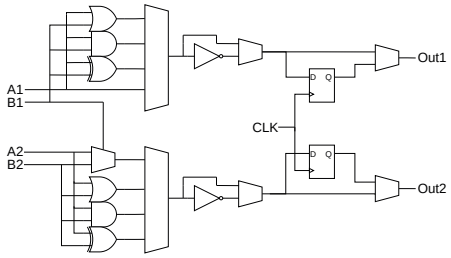


Fig. 3. The newly proposed cell

IV. Evaluation Results

A. Area

Figure 4 compares the baseline, cFA and newly proposed cell architectures. It is clear that the routing area is the biggest contributor in area, especially in the architectures based on the cFA cell and the newly proposed cell.

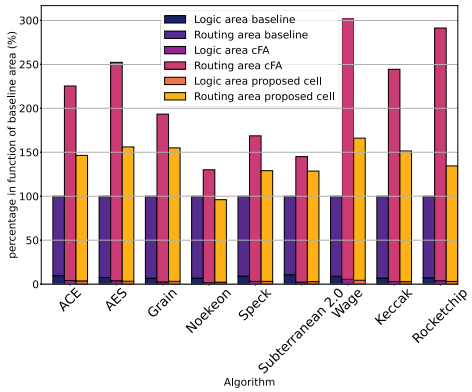


Fig. 4. Relative routing and logic area usage for the baseline architecture and the cFA-based architecture

B. Frequency

Figure 5 shows the relative operating frequency of the benchmark algorithms for the different architectures. Both the cFA-based architecture and the architecture based on the newly proposed cell are slower for all algorithms when compared to the frequency of the baseline architecture. This is due to the additional routing delay, because the signal needs to pass through more switch and configuration boxes in both architectures.

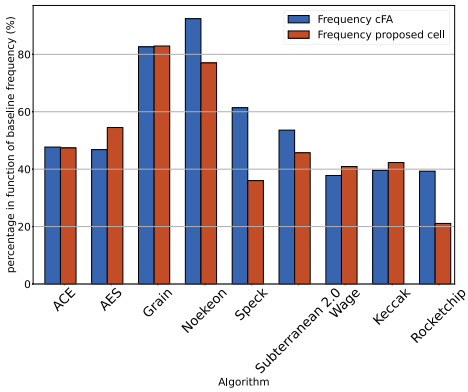


Fig. 5. Relative frequency of the architectures based on the cFA cell and the newly proposed cell compared to the baseline architecture

C. Bitstream

Figure 6 shows the required bitstream size for all architectures. It is clear that the proposed architecture requires less bits than the cFA-based architecture for all benchmarked algorithms. The proposed cell requires an approximately equal amount of configuration bits when compared to the baseline LUT-based architecture. The amount of configuration bits influences the area of a FPGA since more configuration bits require a larger configuration memory.

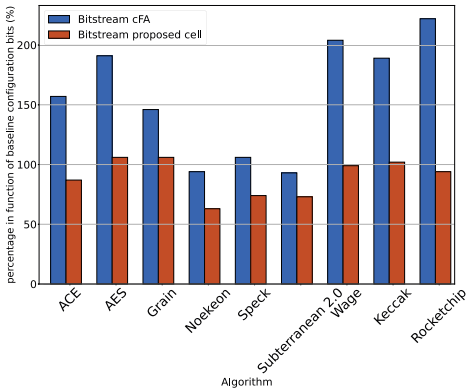


Fig. 6. Relative bitstream size of the architectures based on the cFA cell and the newly proposed cell compared to the baseline LUT-based architecture

V. Analysis of the cFA Architecture

As expected, when taking into account also the routing, the comparison of the cFA architecture is less favorable than the results obtained for only synthesis. One possible explanation could be that the cFA has 2 parts, while Yosys, the synthesis tool, does not use each part of the cFA, leaving a significant amount of cell area and its corresponding routing unused. The largest imbalance occurred in the AES circuit, where only 8% of the gates used were on side 1 of the cFA.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on August 07,2023 at 10:43:24 UTC from IEEE Xplore. Restrictions apply.

The imbalance of logic functions partially explains the results. To evaluate the impact of the imbalance, the two parts of the cFA were merged by sharing their inputs and choosing one of the two outputs with a multiplexer. The design is shown in Figure 7.

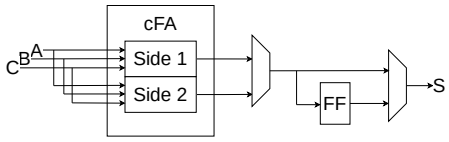


Fig. 7. cFA slice with 3 inputs and 1 output

In Table I, the used routing area of this 3-input 1-output cFA slice is compared with the original 6-input 2-output version. The last column lists the ratio between the routing area of the original cFA and the routing area of the cFA with 3 inputs and 1 output.

TABLE I
ROUTING USAGE COMBINED CFA CELL

Algorithm	Routing area usage cFA	Routing area usage 3-in 1-out cFA	Routing area ratio (%)
Ace	3.67097e+6	2.95844e+6	80.59%
AES	4.45659e+7	3.28332e+7	73.67%
Grain	1.13425e+7	9.48802e+6	83.65%
Noekeon	1.65812e+6	1.56822e+6	94.58%
Speck	6.65534e+5	5.22763e+5	78.55%
Subterranean 2.0	2.45269e+6	2.16206e+6	88.15%
Wage	2.61057e+6	2.09694e+6	80.32%
Keccak	3.40206e+7	2.97984e+7	87.59%

VI. ARTIFACTS

The code to run the toolflow is available at https://gitlab.esat.kuleuven.be/jelle.biesmans/custom-efpga

VII. CONCLUSION

Towards the goal of enabling efficient cryptographic agility, we focused on reconfigurable architectures for cryptography and we compare three different designs of reconfigurable blocks for cryptographic applications. The first architecture is based on a traditional Lookup Table structure. The second uses the configurable Full Adder approach proposed by Mentens et al.. The third is based on a newly proposed configurable cell.

Few considerations can be made from the analysis of the achieved results. The asymmetric nature of the cFA cell lowers the performance in our design flow. This is because current synthesis tool does not optimize towards a balanced occupation of both parts in the cFA cell. To this end, could be interesting to explore tailored synthesis tools.

With the tool chain we used in this work, the newly proposed cell achieves better results than the cFA cell. However, the area occupation for the routing of our designs mostly overshadows the area reduction achieved by the cell itself.

Our results confirm that the routing area plays a significant role in the total area occupation of cryptographic primitives implemented in reconfigurable hardware. Although this trend was expected, for the first time, our work quantifies it in a precise way, providing important insights on how to design a cell for specialized reconfigurable architectures.

REFERENCES

[1] L. Liu, J. Zhu, Z. Li, Y. Lu, Y. Deng, J. Han, S. Yin, and S. Wei, "A survey of coarse-grained reconfigurable architecture and design: Taxonomy, challenges, and applications," *ACM Computing Surveys (CSUR)*, vol. 52, no. 6, pp. 1–39, 2019.

[2] M. Shafique, R. Hafiz, M. U. Javed, S. Abbas, L. Sekanina, Z. Vasicek, and V. Mrazek, "Adaptive and energy-efficient architectures for machine learning: Challenges, opportunities, and research roadmap," in *2017 IEEE Computer society annual symposium on VLSI (ISVLSI)*. IEEE, 2017, pp. 627–632.

[3] S. C. Goldstein, H. Schmit, M. Moe, M. Budiu, S. Cadambi, R. R. Taylor, and R. Laufer, "PipeRench: A Co/Processor for Streaming Multimedia Acceleration," *SIGARCH Comput. Archit. News*, vol. 27, no. 2, p. 28–39, may 1999. [Online]. Available: https://doi.org/10.1145/307338.300982

[4] A. Elbirt and C. Paar, "An instruction-level distributed processor for symmetric-key cryptography," *IEEE transactions on parallel and distributed systems*, vol. 16, no. 5, pp. 468–480, 2005.

[5] J. G. Clifford Wolf, "Yosys - A Free Verilog Synthesis Suite," in *Proceedings of Austrochip*, 2013.

[6] K. E. Murray, O. Petelin, S. Zhong, J. M. Wang, M. ElDafrawy, J.-P. Legault, E. Sha, A. G. Graham, J. Wu, M. J. P. Walker, H. Zeng, P. Patros, J. Luu, K. B. Kent, and V. Betz, "VTR 8: High Performance CAD and Customizable FPGA Architecture Modelling," *ACM Trans. Reconfigurable Technol. Syst.*, 2020.

[7] K. E. Murray, M. A. Elgammal, V. Betz, T. Ansell, K. Rothman, and A. Comodi, "SymbiFlow and VPR: An Open-Source Design Flow for Commercial and Novel FPGAs," *IEEE MICRO*, vol. 40, no. 4, pp. 49–57, 2020.

[8] X. Tang, E. Giacomin, B. Chauviere, A. Alacchi, and P.-E. Gaillardon, "OpenFPGA: An Open-Source Framework for Agile Prototyping Customizable FPGAs," *IEEE MICRO*, vol. 40, no. 4, pp. 41–48, 2020.

[9] D. Koch, N. Dao, B. Healy, J. Yu, and A. Attwood, "FABulous: An Embedded FPGA Framework," in *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 45–56. [Online]. Available: https://doi.org/10.1145/3431920.3439302

[10] N. Mentens, E. Charbon, and F. Regazzoni, "Rethinking Secure FPGAs: Towards a Cryptography-Friendly Configurable Cell Architecture and Its Automated Design Flow," in *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2018, pp. 215–215.

[11] M. Aagaard, R. AlTawy, G. Gong, K. Mandal, and R. Rohit, "ACE: An authenticated encryption and hash algorithm," *Submission to NIST-LWC*, 2019.

[12] J. Daemen and V. Rijmen, *The Design of Rijndael: The Advanced Encryption Standard (AES)*. Berlin, Heidelberg: Springer Berlin / Heidelberg, 2020.

[13] M. Hell, T. Johansson, W. Meier, J. Sönnerup, and H. Yoshida, "Grain-128AEAD-A lightweight AEAD stream cipher," *NIST Lightweight Cryptography, Round*, vol. 1, 2019.

[14] J. Daemen, M. Peeters, G. Van Assche, and V. Rijmen, "Nessie proposal: NOEKEON," in *First Open NESSIE Workshop*, 2000, pp. 213–230.

[15] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The SIMON and SPECK families of lightweight block ciphers," Cryptology ePrint Archive, Report 2013/404, 2013, https://eprint.iacr.org/2013/404.

[16] J. Daemen, P. M. C. Massolino, A. Mehrdad, and Y. Rotella, "The subterranean 2.0 cipher suite," *IACR Trans. Symm. Cryptol.*, vol. 2020, no. S1, pp. 262–294, 2020.

[17] R. AlTawy, G. Gong, K. Mandal, and R. Rohit, "WAGE: An authenticated encryption with a twist," Cryptology ePrint Archive, Report 2020/435, 2020, https://eprint.iacr.org/2020/435.

[18] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Keccak," in *Advances in Cryptology – EUROCRYPT 2013*, ser. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 313–314.

[19] K. Asanovic, R. Avizienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio, H. Cook, D. Dabbelt, J. Hauser, A. Izraelevitz *et al.*, "The rocket chip generator," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17*, vol. 4, 2016.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on August 07,2023 at 10:43:24 UTC from IEEE Xplore. Restrictions apply.