CoBEA: Framework for Evolving Hardware by Direct Manipulation of FPGA Bitstreams

Jörn Hoffmann*
Clemens Fritzsch*
Martin Bogdan
jhoffmann@informatik.uni-leipzig.de
fritzsch@informatik.uni-leipzig.de
bogdan@informatik.uni-leipzig.de
Leipzig University
Leipzig, Germany

ABSTRACT

Evolvable Hardware is a general approach to apply Evolutionary Algorithms to hardware in order to design, improve, or adapt circuits. Approaches that directly manipulate the bitstream of field-programmable gate arrays (FPGAs) had been abandoned due to the lack of well-documented bitstream formats.

Recent advancements in open source FPGA toolchains fundamentally changed the feasibility of direct bitstream manipulation yet again. Unfortunately, contemporary tools are slow and waste valuable time calling external tools.

Therefore, we present an integrated approach that combines bitstream manipulation, low-level communication, and hardware evaluation into a single framework called CoBEA. In addition, the framework allows compaction of the bitstream and direct configuration of the FPGA device without having to program flash memory. Compared to the state of the art, our framework achieves an acceleration of 130 times for FPGA reconfiguration. This allows complex hardware evolution experiments to be performed.

CCS CONCEPTS

• **Hardware → Evolvable hardware**; • **Software and its engineering** → *Abstraction, modeling and modularity*; *Software performance*; • **Computing methodologies → Evolvable hardware**.

KEYWORDS

Bitstream manipulation, Evolvable hardware, Framework

ACM Reference Format:
Jörn Hoffmann, Clemens Fritzsch, and Martin Bogdan. 2022. CoBEA: Framework for Evolving Hardware by Direct Manipulation of FPGA Bitstreams. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3520304.3528821

*Both authors contributed equally to this research.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9268-6/22/07.
https://doi.org/10.1145/3520304.3528821

1 INTRODUCTION

Evolvable Hardware (EHW) generally describes approaches that apply Evolutionary Algorithms (EA) to hardware. The subdomain of evolutionary hardware design is concerned with the creation of electronic circuits for implementation in a target system [4].

Field-programmable gate arrays (FPGAs) are a very suitable target system for evolutionary hardware design. They are readily available and highly reconfigurable. Thompson has shown their capacity for solving non-trivial problems [6]. His approach was to directly manipulate the configuration bits inside the FPGAs bitstream with a Genetic Algorithm and evaluate the fitness intrinsically.

Unfortunately, the FPGAs with well-documented bitstreams (e.g. the Xilinx 6200 family) were discontinued. Due to security concerns, the vendors keep details about the bitstreams of modern FPGAs secret. In conjunction with the use of switch-matrices for routing, the direct manipulation of bitstreams was deemed unfeasible [4]. It was abandoned in favor of Virtual Reconfigurable Circuits (VRCs) [5] that implement an application specific reconfigurable system inside an FPGA. A VRC can be reconfigured without reconfiguration of the underlying FPGA. Another alternative is the use of dynamic partial reconfiguration (DPR) to switch whole blocks of the configuration without having to know the internal structure of the partial bitstream [7]. While these abstractions enable the safe application of EAs, they neglect the huge potential for solutions outside of their abstraction model.

While there were attempts to analyze and document the unknown bitstream formats, the results were limited. Cancare et al. [1] were able to document the configuration bits for lookup tables in Xilinx Virtex-4 FPGAs. The more complex routing resources proved too difficult and their potential remained unused.

Recent advancements in open source FPGA toolchains fundamentally changed the feasibility of direct bitstream manipulation yet again. They led to a profound documentation of Lattice iCE40 FPGAs by Project Icestorm [9]. Whitley et al. leveraged the provided open source toolchain and the included documentation to show the feasibility of safe direct bitstream manipulation [8]. Nevertheless their approach proved to be highly inefficient. Their implementation takes 3.5 seconds for a single reconfiguration of the FPGA. Thompsons experiment, for example, took 5000 generations with 50 individuals each [6]. As a result the necessary reconfigurations take more than 10 days. Our investigation has identified the reason for

this lack of speed: a low degree of integration and the dependence on external programs in combination with ill-suited hardware.

In this paper, we therefore present a highly integrated and modular framework called CoBEA (Configure by Evolutionary Algorithm). It speeds up the reconfiguration more than 130 times and is an all-in-one solution. It contains the possibility to implement experiments, integrate new EAs, allows the direct manipulation of bitstreams, is able to directly program the FPGA and can save the results to CSV or HDF5 format.

Our main contributions in this paper are:

- An integrated approach to directly manipulate the bitstream. We show how it can be used to reduce the reconfiguration time.
- The implementation of the integrated approach in the form of the open-source CoBEA framework.
- An investigation of the speedup of the reconfiguration time by comparing experimental measurements of CoBEA with an implementation based on Project Icestorm.

The rest of the paper is organized as follows. Section 2 describes the peculiarities of evolutionary hardware design by direct bitstream manipulation. In section 3, we briefly present the implementation of our framework and the difference to other approaches. Section 4 covers experimental measurements and section 5 discusses them. Section 6 concludes the paper with a summary and an outlook on further research.

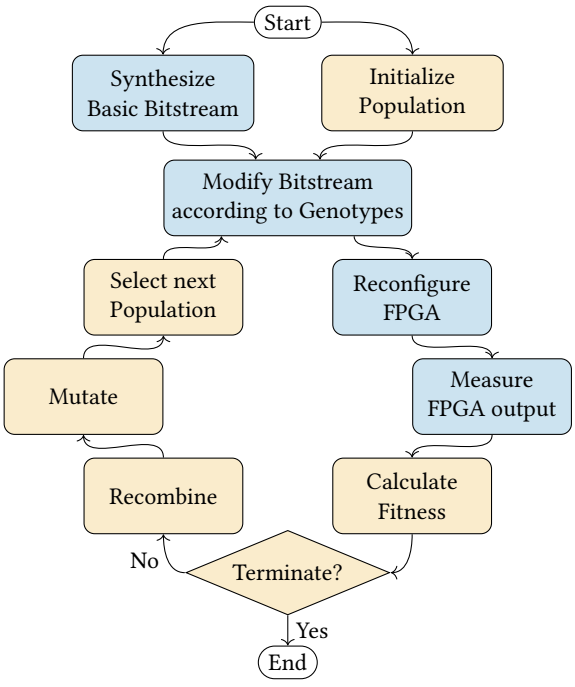


Figure 1: Evolutionary hardware design by direct manipulation of the bitstream. Most steps are common for EAs while others are peculiar to this approach.

2 BASICS

Evolutionary hardware design aims to create electronic circuits for implementation in a target system by employing Evolutionary Algorithms. The use of direct bitstream manipulation implies that the target system is an FPGA and that the fitness value is determined intrinsically. Intrinsic fitness evaluation means that the circuit candidate is implemented in the target FPGA and measured in an application specific way.

A generalized overview of evolutionary hardware design by manipulating bitstreams is shown in Figure1. Most steps are typical for EAs, like population initialization or mutation. Their details depend on the specific employed EA.

Four steps are indispensable when manipulating bitstreams directly. Since a complete FPGA is very complex, only a portion of it is designated as evolvable region. Other parts are kept constant and defined in a basic bitstream. This bitstream has to be synthesized before the evolutionary cycle can be started. Inside the cycle, its evolvable region is modified according to the genotypes to create a matching configuration. Afterwards, the target FPGA is reconfigured with the bitstream and the measurements for the intrinsic fitness evaluation are taken.

The most time-consuming tasks in the evolutionary cycle are the reconfiguration of the FPGA and the measurements. The measurements have to be chosen according to the desired circuit and have to be optimized for each application. The reconfiguration, on the other hand, is independent of the application and can be optimized generically. Therefore, we focused on the acceleration of the reconfiguration in our CoBEA framework.

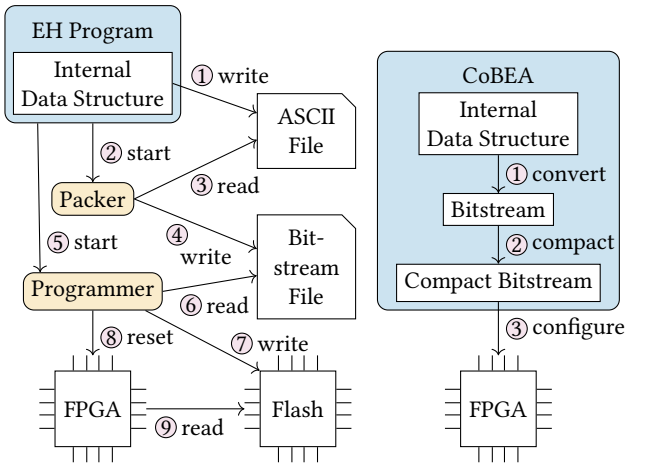
3 IMPLEMENTATION

The most recent approach to EHW with direct bitstream manipulation employ an FPGA toolchain. Such toolchains typically consist of distinct tools for specific tasks, like creating the bitstream from intermediate representations or reconfiguration of the FPGA.

The reconfiguration workflow used by Whitley et al. [8] is depicted in figure Figure2(a). It involves expensive data transformations and the execution of external tools. At first, the internal data structure of the circuits is written to an ASCII representation ①. Then an external packer is invoked ② which reads the file ③ and creates the actual bitstream file ④. After that, a programmer tool is executed ⑤. It reads the bitstream ⑥ and writes it to the flash memory beside the FPGA ⑦. Finally, the FPGA is reset ⑧ to trigger the reconfiguration and the reading of the bitstream from the flash memory ⑨.

As depicted in Figure2(b), our approach reduces this procedure to only three steps. First, it converts the internal representation of the circuitry directly to a bitstream representation ①. Then our tool automatically compacts this bitstream to reduce its size ②. For this purpose, it removes unused configuration data, omits metadata and reorders the commands send to the device. Finally, it triggers the reconfiguration of the fabric and pushes the configuration to the FPGA ③.

It is important to emphasize that the FPGA is configured directly without flash memory. On the one hand, this increases the configuration speed, since the programming cycles require additional time. On the other hand, reliability is increased as the flash



(a) Using an FPGA toolchain and flash.

(b) Using CoBEA and direct reconfiguration.

Figure 2: Approaches for reconfiguring an FPGA for evolutionary hardware design.

memory wears out with each programming cycle. Unfortunately, the availability of direct reconfiguration depends on the system around the FPGA. We strongly advise to select a system that allows for direct reconfiguration, like the iCE40 HX8K breakout board (ICE40HX8K-B-EVN).

In addition to the reconfiguration module for multiple FPGAs, the CoBEA framework currently consists of reusable data structures for EA (e.g. genes), simple Evolutionary Algorithms, and example experiments (e.g. [6]). Furthermore, it is able to read and write comma separated value (CSV) files and supports the Hierarchical Data Format version 5 (HDF5), which is widely used for the exchange of research results.

The framework itself is completely written in Python. The code follows the clean architecture principle [3] and is therefore highly extensible and maintainable. The code is open source and can be obtained from https://github.com/nmi-leipzig/cobea.

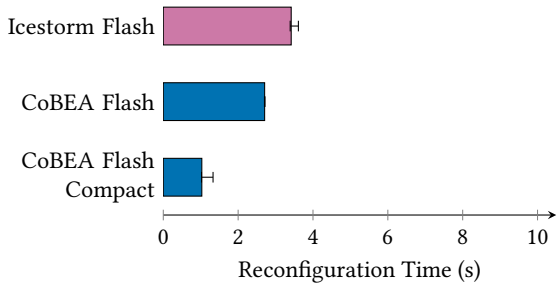
4 EXPERIMENTS

The experiments are focused on the reconfiguration speed. The reconfiguration is the most time-consuming part that can be optimized without regard to the actual circuit design.

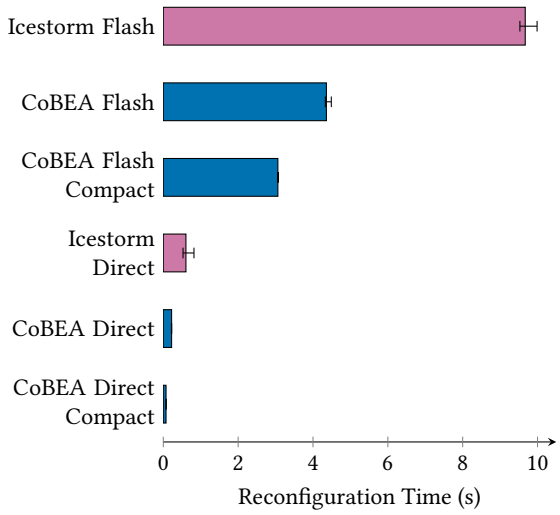
4.1 Setup

The reconfiguration module of our CoBEA framework was isolated and compared with an implementation based on the open source toolchain Project Icestorm [9], which is used in other approaches [8]. The software used to create the measurements is available at https://zenodo.org/record/6418292.

The experiment consists of loading a prepared bitstream onto an FPGA board. The two used bitstreams are intermediate results from actual EHW experiments with the software by Whitley et al. [8] on an HX1K FPGA and with CoBEA on an HX8K FPGA. In order to compensate for fluctuations, the reconfiguration is repeated 100 times back to back and the average time is determined. Two FPGA platforms are used: an iCEstick with an iCE40 HX1K FPGA and an iCE40 HX8K breakout board (ICE40HX8K-B-EVN). In contrast to the iCEstick, the HX8K breakout board allows direct reconfiguration without writing to a flash first. Both approaches were measured for the HX8K breakout board. Additionally, CoBEA was measured with and without bitstream compaction.



(a) For iCEstick with iCE40 HX1K. The iCEsticks design does not allow direct reconfiguration of the FPGA.



(b) For iCE40 HX8K breakout board.

Figure 3: Reconfiguration times using Icestorm or CoBEA. The error bars show the minimum and maximum.

4.2 Results

In the first experiment, the HX1K on the iCEStick was used. The results are depicted in Figure3(a). Our integrated approach is faster than the Icestorm-based implementation even without compaction. With compaction, a speedup of more than $3x$ was achieved. Since the iCEstick does not allow direct reconfiguration of the FPGA without hardware modifications, both tools had to write to the flash and trigger an FPGA reset. Therefore, the speedup of CoBEA is mainly achieved by the compaction of the bitstream.

In the second experiment, the more flexible breakout board with the HX8K was used. Both tools are therefore able to write either to

the flash or reconfigure the FPGA directly. Comparing Figure3(b) to Figure3(a) demonstrates the effect of the larger bitstream size for the HX8K FPGA. The Icestorm-based implementation takes more than double the time for a single reconfiguration. On the other hand, the potential of our integrated approach is also demonstrated as CoBEA with compaction is still faster for the HX8K than the Icestorm based implementation for the HX1K.

The situation dramatically changes when the direct reconfiguration of the FPGA is considered. Here, both tools achieve a significant reduction in reconfiguration time. The Icestorm-based implementation reconfigures the FPGA nearly 16 times faster compered to the usage of flash. CoBEA with compaction in turn is more than 8 times faster than the Icestorm-based implementation.

Overall, CoBEA directly reconfiguring the FPGA achieves an speedup of more than $130x$ compared to the standard case of writing to the flash with the Icestrom toolchain.

5 DISCUSSION

The experiments show a tremendous increase in reconfiguration speed. Thompsons experiment [6] with 50*5 000 evaluation would need less than 3 days for reconfigurations on the iCEstick when using CoBEA compared to more than 9 days using the an Icestorm-based implementation. Using the HX8K breakout board and direct reconfiguration the overall time for reconfiguration reduces to 6 hours.

Three conclusions can be drawn from these results. First, full integrated approaches like CoBEA gain a significant speedup over calling external programs in the evolutionary cycle. Second, even better programming times can be achieved when the bitstream is compacted. And third, for this kind of experiments, the FPGA has to be reconfigured directly.

CoBEA directly manipulates the bitstream of an FPGA and does not rely on repeated synthesis runs. It achieves a significant speedup over current approaches due to the integration of formerly external tools. Because of its clean architecture, our framework is both extensible and maintainable. This makes it comparatively easy to support other FPGAs or add new Evolutionary Algorithms. In contrast to other tools, it enables the experimental setup and results to be exchanged via standardized data formats such as HDF5. However, one disadvantage that can be mentioned is that CoBEA is more complex. But this downside is manageable due to the architectural design.

A promising area of application for CoBEA is security research. In contrast to classical design tasks, many security applications are not impeded by the limited portability of results of EHW by direct bitstream manipulation. They may even benefit from it, like the security by diversity approach. Collins et al. [2], for example, impeded reverse engineering by applying EHW on the level of the hardware description language to create diverse designs for the same function. Direct bitstream manipulation with CoBEA could take this one step further and bind a specific design to a single device.

6 CONCLUSION

In this paper we present an integrated approach to directly manipulate the bitstream of FPGAs for evolutionary hardware design. Our implementation, the CoBEA (Configure by Evolutionary Algorithm) framework, consists of predefined Evolutionary Algorithms, example experiments and reusable data structures.

CoBEA incorporates three approaches. First, the direct manipulation of the bitstream to omit additional synthesis steps for circuit evolution. Second, the integration of the FPGA low level handling, which eliminates time-consuming external program calls. And third, the compaction of the bitstream in combination with the direct reconfiguration of the FPGA. Compared to the state of the art, the results show that an evolutionary experiment on different FPGA devices with CoBEA can achieve a speed increase of more than $130x$.

In further work, it is planned to support more FPGAs like the Xilinx 7-Series. In opposite to Lattice iCE40 family, they support the dynamic partial reconfiguration. This makes it possible to update only parts of the chip to further speed up the reconfiguration process.

Another point is to use CoBEA to conduct experiments with circuits for security purposes. In this way, the plan is to morph or individualize circuits to harden them against hardware-level attacks such as differential power analysis and clock glitching. For example, we consider to extend the work of Collins et al. [2]. On the other hand, we plan to develop hardware around existing circuits to analyze and manipulate their internal processes.

DATA AVAILABILITY

The time measurements and the bitstreams that support Figure3 are openly available at https://zenodo.org/record/6413619.

ACKNOWLEDGMENTS

This work was supported by the German Federal Ministry of Education and Research (BMBF, 01IS18026B) by funding the competence center for Big Data and AI "ScaDS.AI Dresden/Leipzig".

REFERENCES

[1] Fabio Cancare, Marco D. Santambrogio, and Donatella Sciuto. 2010. A direct bitstream manipulation approach for Virtex4-based evolvable systems. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. IEEE. https://doi.org/10.1109/ISCAS.2010.5537429

[2] Zachary Collins, Bayley King, Rashmi Jha, David Kapp, and Anca Ralescu. 2019. Evolvable Hardware for Security through Diverse Variants. In *2019 IEEE National Aerospace and Electronics Conference (NAECON)*. IEEE. https://doi.org/10.1109/NAECON46414.2019.9058062

[3] Robert C. Martin. 2017. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall, Boston, MA. https://www.safaribooksonline.com/library/view/clean-architecture-a/9780134494272/

[4] Ruben Salvador. 2016. Evolvable Hardware in FPGAs: Embedded tutorial. In *2016 International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS)*. IEEE. https://doi.org/10.1109/DTIS.2016.7483877

[5] Lukáš Sekanina and Richard Růžička. 2000. Design of the Special Fast Reconfigurable Chip Using Common FPGA. In *Proc. of Design and Diagnostics of Electronic Circuits and Systems - IEEE DDECS'2000* (Smolenice, SK). 161–168. https://www.fit.vut.cz/research/publication/6394

[6] Adrian Thompson. 1997. An evolved circuit, intrinsic in silicon, entwined with physics. In *Evolvable Systems: From Biology to Hardware*. Springer Berlin Heidelberg, 390–405. https://doi.org/10.1007/3-540-63173-9_61

[7] Jim Torresen, Geir Aarstad Senland, and Kyrre Glette. 2008. Partial Reconfiguration Applied in an On-line Evolvable Pattern Recognition System. In *2008 NORCHIP*. IEEE. https://doi.org/10.1109/NORCHP.2008.4738283

[8] Derek Whitley, Jason Yoder, and Nicklas Carpenter. 2021. Resurrecting FPGA Intrinsic Analog Evolvable Hardware. In *The 2021 Conference on Artificial Life*. MIT Press. https://doi.org/10.1162/isal_a_00448

[9] Claire Wolf and Mathias Lasser. 2015. Project IceStorm. http://bygone.clairexen.net/icestorm/.