Evolving Hardware by Direct Bitstream Manipulation of a Modern FPGA

Clemens Fritzsch, Jörn Hoffmann and Martin Bogdan
Neuromorphic Information Processing, Leipzig University, Leipzig, Germany
Email: {fritzsch, jhoffmann, bogdan}@informatik.uni-leipzig.de

***Abstract*—Field-programmable gate arrays (FPGAs) have recently received renewed attention in the context of Evolvable Hardware (EHW). The most fine grained approach to changing their internal structure, direct manipulation of the bitstream, has largely been abandoned. The undocumented bitstream formats of modern FPGAs made it complicated and error-prone. This situation has fundamentally changed with the advent of open-source FPGA toolchains. Previous attempts to exploit this opportunity were promising, but only manged to solve very basic tasks. We present in this paper an evolved tone discriminator circuit. It was evolved by replicating the most famous experiment in this field, but with modern hardware. For that we map the originally used Xilinx XC6200 FPGA to a modern Lattice iCE40 FPGA. We show how to set up the experiment and optimize the evolution environment. Our approach allows over 130 times more reconfigurations per second than previous approaches. Additionally, we discuss reasons for the abandonment of direct bitstream manipulation for EHW in context of the new possibilities created by open-source FPGA toolchains. We show which challenges have been solved and which steps need to be taken next.**

I. Introduction

The basic concept of Evolvable Hardware (EHW) is the application of Evolutionary Algorithms (EAs) to hardware. The idea was first proposed by de Garis [1]. The term itself was later coined in a work together with Higuchi et al. [2] evolving a 4-to-1 multiplexer.

The 4-to-1 multiplexer is an example of an evolutionary hardware design where Evolutionary Algorithms are applied during the design phase. No evolutionary changes are made during the application. In contrast, adaptive hardware embeds the EA in the application system and allows evolution during the deployment. The goal is to enable the system to adapt to changes in the environment, the task itself, or defects in the underlying hardware substrate [3].

Furthermore, the 4-to-1 multiplexer [2] was evolved with extrinsic evaluation, i.e. the fitness of possible hardware designs was evaluated in a simulation. The opposite approach is called intrinsic evaluation, where possible hardware designs are implemented in the target technology to determine their fitness. The latter has various advantages as it can exploit unique physical properties of the technology [3] that are difficult to simulate. This enables them to find designs not possible with more abstract approaches.

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".

Field-programmable gate arrays (FPGAs) are a suitable substrate for intrinsic EHW. They are readily available and easy to reconfigure. The first FPGA-based experiment is the tone discriminator by Thompson [4]. He evolved a circuit, that was able to distinguish between a 1 kHz and a 10 kHz square wave signal without a reference clock. It generates a logical high output for one signal and a logical low for the other. Thompson used a Xilinx XC6200 FPGA for the experiment. It was suitable for intrinsic EHW on account of its well documented bitstream format. Due to the use of multiplexers for routing, all combinations of configuration bits were valid, without the possibility for unsafe values. This allowed the direct and unrestricted manipulation of the bitstream by the Evolutionary Algorithm. The XC6200 family was discontinued without a comparably suitable replacement. Their modern counterparts in contrast are much more flexible, but use switch-matrices for their routing. This leads to the existence of invalid values for the configuration bits, which can even destroy the FPGA itself. In addition, the bitstream formats of modern FPGAs are kept secret by the vendors. The combination of these two factors made the direct bitstream manipulation unfeasible.

Virtual Reconfigurable Circuits (VRCs) are a possible alternative to intrinsic EHW. Sekanina [5] introduced VRC as an additional reconfiguration layer above the FPGA hardware. Reconfiguration of the evolvable region is completely facilitated by the VRCs without the need to reconfigure the FPGA itself. This enables fast (virtual) reconfigurations, as they can be performed within a few clock cycles. Dobai and Sekanina were able to reach 8700 evaluations per second [6]. The additional abstraction layer also allows for a higher portability of an evolved design. Alas, the granularity of the evolution is coarser in regard to the underlying FPGA technology. This makes it difficult to exploit unique physical properties. Another disadvantage is the high resource requirement for implementing the abstraction layer.

In this context, there were some attempts to document bitstream formats to allow direct manipulation for EHW again. Hollingworth et al. [7], for example, created a Java interface for bitstream manipulation of older Xilinx Virtex devices. They avoided invalid configurations by restricting the resources to a subset, that resembled the simplified XC6200 basic cell as used by Thompson [4]. Cancare et al. [8] documented the configuration bits for the lookup tables (LUTs) of Xilinx Virtex4 FPGAs and successfully evolved circuits for small

2022 IEEE Congress on Evolutionary Computation (CEC) | 978-1-6654-6708-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/CEC55065.2022.9870297

tasks like parity generators. Unfortunately, the documentation of other resources proved to be too difficult due to the risk of permanent damage to the FPGA.

Additionally, open-source FPGA toolchains have made strides in recent years. They include at least partial documentation of the bitstream formats of modern FPGAs. This provides a perspective for EHW with direct bitstream manipulation.

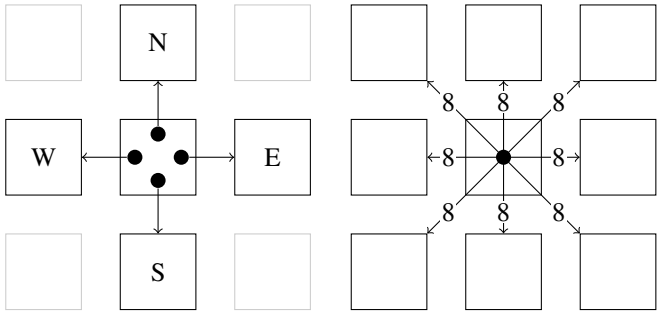
The most advanced open-source toolchain exists for the Lattice iCE40 FPGA family [9]. It is able to create a bitstream from Verilog code without requiring any closed-source tools, like the vendor toolchain iCEcube2. The bitstream format and the configuration bits of the iCE40 family are well documented in Project IceStorm [10]. In addition, iCE40 FPGAs are affordable and breakout boards to simplify the development are available. A disadvantage, however, is that iCE40 FPGAs do not support partial reconfiguration. Nevertheless, they can be used for EHW.

Whitley et al. [11] demonstrated the capability of the iCE40 architecture for evolving hardware by direct bitstream manipulation. They evolved circuits for the simple tasks of amplitude maximization and pulse oscillation. However, they were unsuccessful in reproducing the more difficult tone discriminator by Thompson [4]. Furthermore, they report the reconfiguration time to be around 3.5 s, which makes difficult tasks impracticable. The tone discriminator experiment, for example, required 250 000 reconfigurations, which would amount to over 10 days for reconfigurations alone. There are two reasons for this slow reconfiguration speed. First, the used hardware only allows for indirect reconfiguration of the FPGA by writing the bitstream to a flash memory beside the FPGA. This way, the configuration must be transferred twice: from the workstation to the flash and from the flash to the FPGA. Secondly, the software created by Whitley et al. [12] calls tools from Project IceStorm as subprocesses. The external calls create an overhead. Furthermore, resources like the connection to the FPGA are acquired and released repeatedly in quick succession.

In this paper, we address the gaps in previous research. We present a faster, more feasible approach for direct bitstream manipulation on modern iCE40 FPGAs. This is leveraged to successfully reproduce the tone discriminator experiment by Thompson [4]. The used software, called CoBEA, is open-source and available at https://github.com/nmi-leipzig/cobea.

Our main contributions in this paper are:

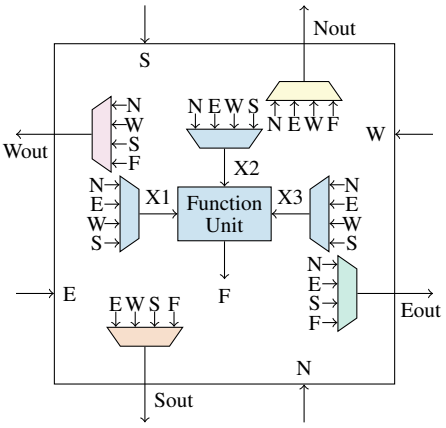
- Replication of the tone discriminator experiment on an iCE40 FPGA with direct bitstream manipulation
- Mapping of the XC6200 resources originally used for the tone discriminator experiment to the modern iCE40 FPGA architecture
- Recommendations on how to setup the evolution environment for direct bitstream manipulation to reduce the reconfiguration time by more than 99 %
- Analysis of the current state of the direct bitstream manipulation approach for Evolvable Hardware, with focus on the impact of open-source FPGA toolchains



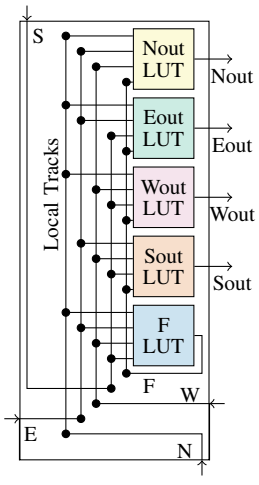
(a) XC6200 cells form a von Neumann neighborhood. The outputs can be configured independently.

(b) iCE40 tiles form a Moore neighborhood. Each output consists of all eight LUT outputs. They are the same in every direction.

Fig. 1. Outgoing connections between neighbors. The incoming connections are correspondent.



(a) Simplified XC6200 basic cell.



(b) Used resources of iCE40 logic tile.

Fig. 2. Overview of the used architectures. Equivalent parts are filled with the same color.

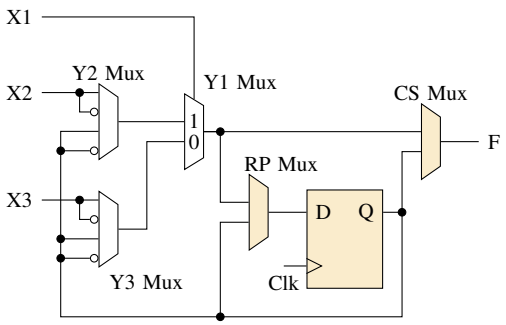


Fig. 3. XC6200 function unit. The flip-flop and two multiplexers are not changed by the Genetic Algorithm.

II. XC6200 on iCE40

A. XC6200 Architecture

Thompson ran his experiment [4] on an FPGA of the Xilinx XC6200 family [13]. He included only basic cells and used only a subset of their resources for the Genetic Algorithm.

Fig.2a shows a single basic cell with all used resources. The cells are connected in a von Neumann neighborhood (see Fig.1a). The four neighbors of a cell are designated as North, East, West, and South (NEWS). Each cell contains a function unit (see Fig.3) that combines three inputs to a single output. Every input of the function unit is independently sourced from the respective outputs of the NEWS neighbors. The outputs of a cell are independently selected from the function unit output and the three outputs of the neighbors not facing the respective output. Regard the North face for example: It receives the South output *S* of the North neighbor and its output *Nout* can be selected from the North output of the South neighbor *N*, the East output of the West neighbor *E*, the West output of the East neighbor *W*, or the output of the function unit *F*.

It is important to note, that all routing is based on multiplexers. Due to that fact, there is no invalid or unsafe routing configuration. This intrinsic property made the XC6200 family highly suitable for Evolutionary Algorithms.

The function unit consists of a D flip-flop and several multiplexers. The circuit is never provided with a clock signal during the experiment, therefore the flip-flop always has a constant output. Since both, the flip-flop output and its negation are provided to *Y1 Mux* and *Y2 Mux*, the actual value of constant flip-flop output does not change the possible outputs of the function unit. Therefore, the D flip-flop and the *RP Mux* can be disregarded. This is in turn also valid for *CS Mux*, which always selects the output of *Y1 Mux*.

B. iCE40 Architecture

The architecture of the iCE40 family is typical for modern FPGAs. It consists of an array of tiles, which are connected in a Moore neighborhood (see Fig.1b). Most of these tiles are logic tiles, which consist of eight lookup tables (LUTs) to accomplish their function. The inputs of the LUTs are routed by a two stage mechanism. First, the output of a neighbor or of another LUT in the same tile is routed to a local track. After that, the local track is routed to a LUT input. In principle, each of the eight outputs of the neighbors can be routed to each LUT, however not to an arbitrary input. Other resources, like the carry chain, flip-flops, or span wires, are not used for the mapping and thus are not discussed.

C. Mapping

The mapping of a (simplified) XC6200 basic cell to an iCE40 logic tile can be seen in Fig.2b. The approach is to assign parts of the basic cell to single LUTs and statically route the necessary inputs to them. Afterwards, the possible truth tables are selected to implement the functionality. Each LUT is represented by one gene, whose alleles are the possible truth tables. The advantage of this approach is its high portability to other architectures. It is even feasible for architectures where only the truth table of the LUTs can be manipulated directly in the bitstream (e.g. [8]).

Each of the four output multiplexers of the basic cell has a corresponding LUT. There are four possible truth tables which simply pass through one of the inputs.

The fifth LUT implements the function unit including its three input multiplexers. While these elements are represented by ten bits[1] in the original bitstream, there are only 166 respective truth tables. This is due to the many logically equivalent combinations. There are, for example, 100 combinations that basically implement a constant zero output.

It should be noted, that logical equivalence does not imply functional equivalence, since the circuit is not clocked and allows effects not observable in time and value discrete systems. One of the 100 combinations for constant zero may produce short peaks during input changes, caused by different signal propagation times, while others do not. Such effects, however, cannot be expected to be replicated on a conceptually different technology such as the iCE40. Nevertheless, the logical equivalence and the structural similarity of the mapping facilitate the comparison of experimental results

III. Reducing Reconfiguration Time

Evaluating the fitness of an individual is one of the most time-consuming parts of running an EA. This is an even more important factor for intrinsic evaluations, because they happen inside an FPGA and cannot be simply accelerated by using more computing power. Here, the reconfiguration time to create the phenotype is one of the most influential variable of the overall running time. It is more remunerative than, for example, to speed up individual measurements, because a reduction in reconfiguration time applies to all experiments on the FPGA.

Table I shows the reconfiguration times of an iCE40 HX8K FPGA, presented in [14]. The FPGA is the same model used in the experiment in Section IV. The bitstream used for the measurements was taken directly from the experiment results.

[1]Three input multiplexers in addition to *Y1 Mux* and *Y2 Mux*, each two bits.

TABLE I
RECONFIGURATION TIMES OF AN ICE40 HX8K FPGA FOR VARIOUS COMBINATIONS OF DIRECT CONFIGURATION OR VIA FLASH, EXTERNAL TOOLS OR INTEGRATED FUNCTIONS, AND COMPACTED OR NOT.

Direct	External Tools	Compacted	Reconfiguration Time (s)		
			Min	Mean	Max
No	Yes	No	9.529	9.671	9.988
No	No	No	4.330	4.358	4.489
No	No	Yes	3.053	3.062	3.078
Yes	Yes	No	0.528	0.605	0.820
Yes	No	No	0.218	0.222	0.225
Yes	No	Yes	0.070	0.074	0.077

The largest difference occurs between direct configuration of the FPGA and writing the bitstream to flash memory. This difference is trivial, since the FPGA always has to additionally load the bitstream from the flash. Nevertheless, many iCE40 based FPGA boards require the indirection via flash memory, e.g. the board used in [11]. Furthermore, the possible write cycles of flash may be less than required for a large EA experiment. The tone discriminator experiment, for example, requires more than 250 000 reconfigurations. Therefore, the ability to be directly reconfigured is an important factor when choosing the board for EHW setup.

The easiest way to evolve hardware with direct bitstream manipulation is calling external tools provided by an open-source FPGA toolchain [12]. The tools provide functionality for transforming bitstreams to different representations and for reconfiguration of the FPGA. Nevertheless, such tools are not optimized for the requirements of EHW, since their repeated invocation generates immense overhead. Resource like the connection to the FPGA are acquired and released for each reconfiguration. This is unnecessary and can be avoided by integrating the functionality of the external tools into the software running the EA. As can be seen in Table I, the integrated approach reduces the time for direct reconfigurations by more than 60 %.

The use of specialized tools allows further optimizations like compaction of the bitstream. It is a viable and often the only way to reduce the bitstream size of low-cost devices without compression capabilities like the iCE40 FPGA. While compaction is not provided by the vendor, the bitstream documentation allows to leave out redundant data. This is most effective when only few resources of the iCE40 FPGA are used. Fortunately, this condition is often met in EHW. The experiment in Section IV uses only 101 of 960 logic tiles. Consequently, the bitstream size and the reconfiguration time can be further reduced. The reconfiguration with compaction takes only one third of the time without it. Overall the reconfiguration can be accelerated more than 130 times.

IV. TONE DISCRIMINATOR EXPERIMENT

The task in the experiment is to evolve a circuit that can discriminate between a 1 kHz and a 10 kHz signal. The evolvable region consists of an area of ten by ten tiles that are prepared as XC6200 basic cells, as described in Section II. The signal is provided as input to one cell at the left border, while the output of one cell at the top is taken as output of the whole circuit.

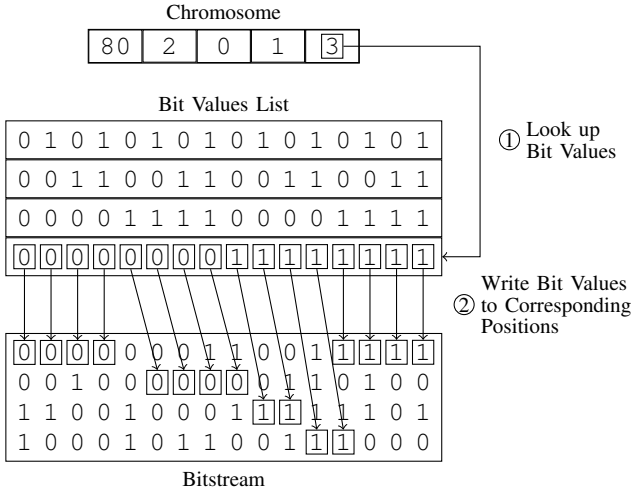


Fig. 4. Decoding of a gene to bits in the bitstream.

The Genetic Algorithm runs not directly on the bitstream bits, but on integers. Since not every value combination of bitstream bits is valid, interdependent bits are grouped together into a single gene. All desired value combinations for these bits are collected in a fixed list of bit values. The chromosome is a list of integers, each describing a position in the list of bit values corresponding to the gene. For example, a *Nout LUT* (see Fig.2b) is described in a single gene containing the 16 bits of the truth table. It has four alleles, one each for passing through *N*, *E*, *W*, or *F* respectively. Fig.4 shows the two decoding steps to modify the bitstream according to a allele.

The initial population of 50 individuals is generated randomly. Further generations are created by first transferring the individual with highest fitness value (elitism). The parents for the remaining 49 individuals are chosen by linear ranked based selection with a selective pressure of 2. A one point crossover takes place with a probability of 0.7. Finally a uniform mutation is applied with a probability of 0.001756 per gene. That kind of mutation was chosen as the alleles have an unordered categorical nature. Values like population size and mutation probability were taken from [4].

A. Setup

The whole evolutionary process is controlled by the EHW software CoBEA. It runs the Genetic Algorithm, drives the other components, and aggregates the data in a single HDF5 file. Furthermore, CoBEA implements bitstream compaction and direct configuration, which enables the fastest way to reconfigure the FPGA as shown in Section III.

Fig.5 shows the components of the setup. To evaluate an individual, the EHW software sets the bitstream bits according to the selected alleles and directly configures the FPGA on an iCE40 HX8K breakout board (ICE40HX8K-B-EVN). Afterwards, it determines a random order of five 1 kHz bursts

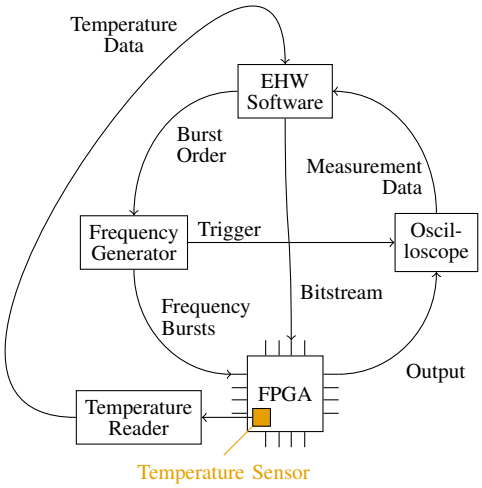


Fig. 5. Setup for the tone discriminator experiment. The Temperature measurement runs continuously during the experiment.

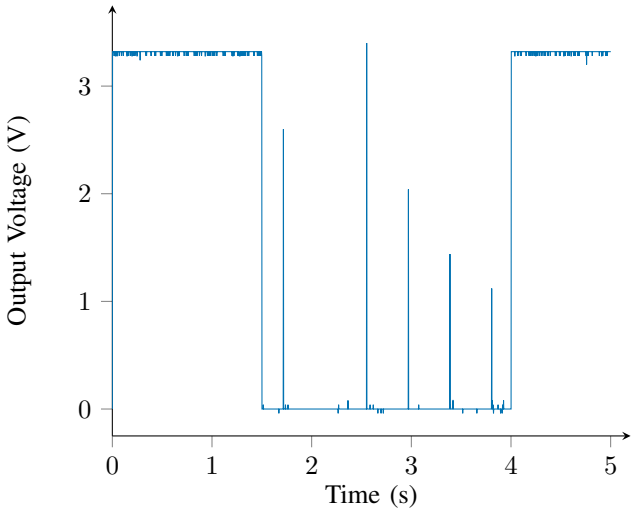


Fig. 6. Output of the final evolved circuit as a function of time. The output is 3.3 V during 10 kHz bursts and 0 V during 1 kHz bursts. A few narrow spikes occur during the 1 kHz bursts.

and five 10 kHz bursts and sends this order to the frequency generator, which is implemented on a second breakout board. The frequency generator produces the requested frequencies in bursts of 500 ms. The output of the frequency generator is wired to the input of the evolvable area. Additionally, a trigger signal is active during the frequency generation. It starts the data recording of an oscilloscope (Rigol DS1102E), which is connected to the output of the evolvable area. The recorded data is sent to the EHW software, which computes the fitness.

The temperature is measured during the whole evolution. For this purpose, a temperature sensor (Maxim DS18B20) with thermal paste is placed directly on the FPGA. The sensor value is read-out every 0.6 s by the temperature reader and sent to the EHW software. The temperature reader is implemented by an Arduino Uno.

The fitness is computed from the voltage measurements during the ten frequency bursts. First the area under the voltage curve a_b is computed for each burst $b \in [1, 10]$. This simulates the analog integrator used by Thompson. The bursts are grouped by their frequency in S_1 for 1 kHz and S_{10} for 10 kHz. Finally the fitness is calculated by (1).

$$fitness = \frac{1}{10}\left|\left(k_1 \sum_{b\in S_1} a_b\right) - \left(k_2 \sum_{b\in S_{10}} a_b\right)\right| \qquad (1)$$

$$where\ k_1 = \frac{1}{30730.746}\ \ k_2 = \frac{1}{30527.973}$$

The fitness function including the constants k_1 and k_2 are taken from [4].

B. Results

After 5355 generation the Genetic Algorithm was manually stopped as there were no further improvements of the fitness value. This took short of 4.5 weeks. The best circuit[2] from the last generation was further analyzed. Fig.6 shows its output. The different values for the two frequencies are recognizable, albeit some noise and several spikes are visible.

The spikes are very narrow, as each time only one of the measured values is substantially higher than 0 V. Consequently, their area under the curve is very small. Their negative contribution to the fitness value is five orders of magnitude below the fitness value itself.

1) Connections: Fig.7 shows the cells of the final circuit and their connections. Surprisingly, the output is not directly connected to the input at all. The Genetic Algorithm found a design that relies heavily on local effects like electromagnetic coupling to transfer information from the input to the output.

2) Clamping: To investigate which cell contribute dynamically to the output, they were set to a fixed output value and the fitness was reevaluated. This process called clamping was executed iteratively by first choosing a random cell. The output of the function unit of this cell was randomly set to constant 0 or constant 1. Afterwards the fitness was evaluated. If the fitness value dropped more than 1 % compared to the fitness of the original circuit, the function unit was reset to its original state. If the fitness value remained close to the original, the constant output was kept for the function unit. Finally these steps were repeated for the remaining, not yet investigated cells. The bound of 1 % was used to reduce the influence of noise on the clamping results.

In Fig.7, the successfully clamped cells are drawn in a lighter color. The majority of the cells do not dynamically contribute to the output of the circuit at all or only in a negligible way. The cells that do dynamically contribute, however, are not clustered together. but distributed throughout the evolvable region.

3) Temperature Dependence: The output of the final circuit is influenced by the temperature. Fig.8 shows the average output as a function of the period of the input signal for three

[2]Identification 283907

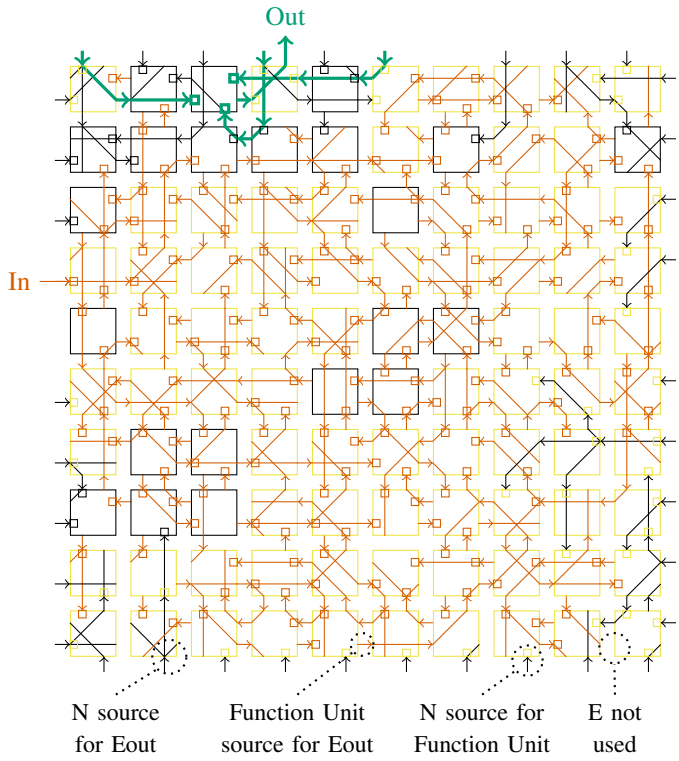


Fig. 7. The final evolved circuit. The function unit of most cells was clamped to a constant output. Elements directly connect to the output and elements directly connected to the input are accentuated.

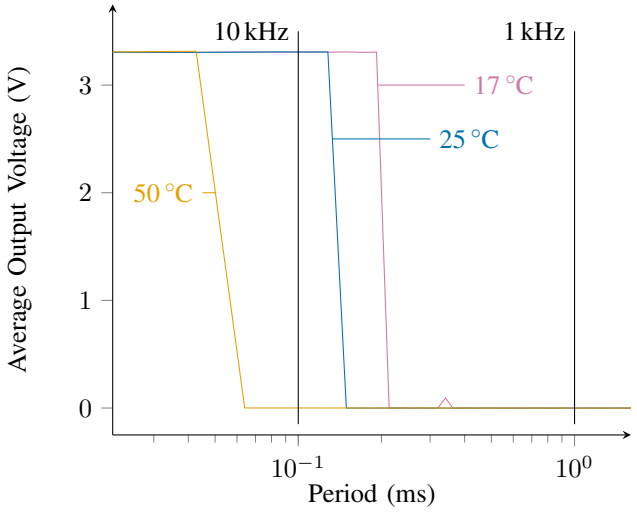


Fig. 8. Output of the final evolved circuit as a function of the period of the input and for three temperatures. The circuit can distinguish 1 kHz and 10 kHz signals at 17 °C and at 25 °C, but not at 50 °C.

different temperatures. The output voltage was averaged over 5 s for each measured period. The shape of all three functions is similar: 3.3 V for high frequencies and then a sharp drop to 0 V. The period where the drop appears depends on the temperature. The drop off period for 50 °C is 0.064 ms, which corresponds to 15.6 kHz. The circuit cannot distinguish 1 kHz and 10 kHz signals at this temperature, because the output is 0 V for both frequencies.

4) Location Dependence: The evolvable region was moved to a different part of the chip, where the final generation was evaluated again. The fitness of the final circuit dropped by two orders of magnitude. The other individuals had approximately the same fitness value. The evolution was continued at the new position for 200 generations. The fitness of the best individuals at that point was still around 2 % lower than the fitness of the final circuit at the original position. This concurs with the output that showed a large amount of short spikes. These effects impede the portability of the evolved circuit immensely. On the other hand, it shows how well the EA was able to utilize the unique physical attributes of the location on the FPGA.

V. Feasibility of Direct Bitstream Manipulation

Direct bitstream manipulation was widely abandoned for EHW. The previous section, however, shows that it is possible, if a suitable open-source FPGA toolchain is available. This section discusses the feasibility of EHW based on direct bitstream manipulation in view of the recent emergence of open-source toolchains. Besides Project IceStorm [10] for iCE40, the focus will be on Project X-Ray [15] that documents the bitstreams of Xilinx 7-Series FPGAs. On one hand the documentation of the 7-Series is making active progress [16]. On the other hand the FPGAs provide desirable features such as Dynamic Partial Reconfiguration (DPR). Additionally, affordable breakout board are readily available.

The first four issues were identified by Salvador as major obstacles for EHW after the discontinuation of the XC6200 FPGAs [3]. The remaining issues are challenges that concern EHW in general [17].

A. Switch-Matrix Routing

Configuration bits for switch-matrix routing cannot be altered randomly and safely as this may create configurations that destroy the FPGA. The documentation of the bitstream by the open-source FPGA toolchains identifies the valid values for the configuration bits. Even if the documentation is not complete, like in Project X-Ray, it is known which bits are safe to manipulate. After the safe values are identified, invalid values can be avoided. One method is to run the EA on a representation that only allows valid values, as was done in Section IV. Therefore this problem can be considered solved.

B. Lack of Tools for Dynamic Partial Reconfiguration

DPR allows to reconfigure only the evolvable region of the FPGA without restarting it. This reduces the reconfiguration time and facilitates new possibilities like module-based evolution. Unfortunately, DPR is not supported by the iCE40 architecture. While the 7-Series supports DPR, it proved difficult to document the mechanism, hence it is not yet included in the open-source FPGA toolchain. The tools provided by vendors have matured regarding DPR, but they impede direct bitstream manipulation. Therefore, DPR remains an open challenge in this context.

C. Bitstream Size and Unknown Format

The bitstream formats are at least partially documented by the respective open-source projects (see also Section V-A), so this concern is resolved. The bitstream size is a more varied challenge. On one hand, DPR is generally available to avoid writing the whole bitstream, yet it is not available in open-source toolchains (see Section V-B). On the other hand, compression options are often available [18]. Furthermore, well documented bitstream formats facilitate new methods for size reduction, like the compaction for iCE40. Overall, the bitstream size proved to be a manageable challenge.

D. Insufficient Reconfiguration Speed

The sufficiency of the reconfiguration speed highly depends on the actual application. Real-time adaptive hardware needs orders of magnitude more reconfigurations per second than evolutionary hardware design. For the tone discriminator experiment (see Section IV) the reconfiguration time can be considered sufficient, since it contributed less than 0.8 % to its whole duration. Yet, the 13.5 reconfigurations per second would be too slow for a real-time adaptive system. It is also at least two orders of magnitude slower than the 8700 evaluations per second reached by VRCs [6]. The reconfiguration speed is an ongoing concern that depends on the actual use case.

E. Portability and Robustness

The portability to other architectures is expected to be low for direct bitstream manipulation, since the goal is to leverage unique physical attributes. Section IV-B, however, shows that even the relocation inside the same device can reduce the functionality of the evolved circuit. The circuits also lack robustness regarding changes in the environment, like the temperature. These effects are well known [4], as are possible solutions. One approach is to include varying conditions in the evolving system [19]. This can be a wide temperature range or evaluation on multiple devices. Portability and robustness remain important concerns that have to be addressed on application basis.

F. Practical Applications

Evolutionary hardware design based on direct bitstream manipulation is not practical for general applications. Especially the challenges with portability and robustness (see Section V-E) put them at a disadvantage compared to conventional design approaches. There are, however, application niches that can benefit from the properties of EHW by direct bitstream manipulation.

One is the field of fault tolerance and fault recovery. Salvador et al. [20] presented self-healing adaptive hardware based on DPR of Processing Elements. As such systems have to handle unique, potentially permanent hardware faults, the concrete configurations cannot be transferred to other devices. Hence, the portability challenges of direct bitstream manipulation EHW are not relevant. Faults can also be subsumed under unique physical attributes, which can be handled well by direct bitstream manipulation based EHW. The remaining challenges are the reconfiguration speed and the absence of DPR. These have to be addressed based on the concrete use case.

Security applications are a second suitable field. The trigger of a Hardware Trojan hides it during tests or detection attempts and starts the malicious activities only in productive systems [21]. The lack of robustness of evolved hardware designed with direct bitstream manipulation can be exploited to design specific triggers, e.g. ones for certain temperature ranges. The same idea can be used in a defensive way to detect if an FPGA leaves its predefined operational envelope and may be target of an attack. Another promising approach is security by diversity. Collins et al. [22] 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 can take this one step further and bind a specific design to a single device by impeding the portability.

VI. Discussion

The Genetic Algorithm presented by Thompson [4] worked surprisingly well for the iCE40 FPGA. Despite the difference in hardware and representation, the same algorithm with the same parameters was able to create a working tone discriminator. It was, for example, in no way obvious that the calibration constants k_1 and k_2 of the fitness function could directly be used for the new environment. They were originally introduced by Thompson to avoid effects caused by the analog integrator he used.

The results of the tone discriminator experiment are similar to the original results by Thompson. The output of the final circuit depends on the temperature and the location on the chip. But Thompson found only a slight decrease in fitness after relocation, while it dropped by orders of magnitude on the iCE40 FPGA. This indicates that it will be more difficult to evolve portable designs with direct bitstream manipulation on the iCE40. The transfer of information by local interactions instead of direct wire connections exemplifies this drawback.

Nevertheless, the integrated approach showed the feasibility of direct bitstream manipulation in regards to reconfiguration speed. All reconfigurations during the experiment together took less than 6 hours. The approach presented in [11], with indirect configuration and the usage of the external tools would have taken 4 weeks.

EHW based on direct bitstream manipulation is currently feasible for applications that have clearly defined envelope of operations and do not require real-time adaption to changed requirements. The most promising approaches come from the field of security research.

VII. Conclusion

We successfully reproduced the tone discriminator experiment by Thompson and showed the feasibility of direct bitstream manipulation on modern iCE40 FPGAs for non-trivial EHW tasks. This was possible by mapping the originally used hardware to the iCE40 architecture and by using the integrated approach for EHW software.

In future work we will add support for more proficient FPGA architectures to our EHW software. Additionally, we will research more suitable EAs and parameters for EHW on modern FPGAs, like crossover variants that incorporate the locality of the tile structures. In combination, this will allow us to implement security applications such as the security by diversity approach.

Data Availability

All data collected during the tone discriminator experiment and the data used for its evaluation in Fig.6, Fig.7, and Fig.8 are openly available at https://zenodo.org/record/6574545. The measurements that support Table I are openly available at https://zenodo.org/record/6413618.

References

[1] H. de Garis, "Genetic programming artificial nervous systems artificial embryos and embryological electronics," in *International Conference on Parallel Problem Solving from Nature*, Springer. Springer Berlin Heidelberg, 1991, pp. 117–123.

[2] T. Higuchi, T. Niwa, T. Tanaka, H. Iba, H. de Garis, and T. Furuya, "Evolving hardware with genetic learning: A first step towards building a darwin machine," in *From Animals to Animats 2*. The MIT Press, 1993, pp. 417–424.

[3] R. Salvador, "Evolvable hardware in FPGAs: Embedded tutorial," in *2016 International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS)*. IEEE, apr 2016.

[4] A. Thompson, "An evolved circuit, intrinsic in silicon, entwined with physics," in *Evolvable Systems: From Biology to Hardware*. Springer Berlin Heidelberg, 1997, pp. 390–405.

[5] L. Sekanina, "Virtual reconfigurable circuits for real-world applications of evolvable hardware," in *Evolvable Systems: From Biology to Hardware*. Springer Berlin Heidelberg, 2003, pp. 186–197.

[6] R. Dobai and L. Sekanina, "Low-level flexible architecture with hybrid reconfiguration for evolvable hardware," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 8, no. 3, pp. 1–24, may 2015.

[7] G. Hollingworth, S. Smith, and A. Tyrrell, "Safe intrinsic and evolution of virtex devices," in *Proceedings. The Second NASA/DoD Workshop on Evolvable Hardware*. IEEE Comput. Soc, 2000.

[8] F. Cancare, M. D. Santambrogio, and D. Sciuto, "A direct bitstream manipulation approach for Virtex4-based evolvable systems," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. IEEE, may 2010.

[9] D. Shah, E. Hung, C. Wolf, S. Bazanski, D. Gisselquist, and M. Milanovic, "Yosys+nextpnr: An open source framework from Verilog to bitstream for commercial FPGAs," in *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, apr 2019.

[10] C. Wolf and M. Lasser, "Project IceStorm," http://bygone.clairexen.net/icestorm/, 2015.

[11] D. Whitley, J. Yoder, and N. Carpenter, "Resurrecting FPGA intrinsic analog evolvable hardware," in *The 2021 Conference on Artificial Life*. MIT Press, 2021.

[12] D. Whitley, J. Yoder, and N. Carpenter, "BitstreamEvolution," https://github.com/evolvablehardware/BitstreamEvolution, 2021.

[13] Xilinx Technical Staff, *The Programmable Logic Data Book*. Xilinx, 1996, ch. XC6200 Field Programmable Gate Arrays, pp. 4–253 – 4–286.

[14] J. Hoffmann, C. Fritzsch, and M. Bogdan, "CoBEA: Framework for evolving hardware by direct manipulation of FPGA bitstreams," in *Genetic and Evolutionary Computation ConferenceCompanion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA*, 2022, [accepted].

[15] "Project X-Ray." [Online]. Available: https://github.com/SymbiFlow/prjxray.git

[16] M. B. Petersen, S. Nikolić, and M. Stojilović, "NetCracker: A peek into the routing architecture of Xilinx 7-Series FPGAs," in *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, feb 2021.

[17] P. C. Haddow and A. M. Tyrrell, "Evolvable hardware challenges: Past, present and the path to a promising future," in *Inspired by Nature*. Springer International Publishing, oct 2018, pp. 3–37.

[18] Xilinx Technical Staff, *7 Series FPGAs Configuration User Guide (UG470)*, Xilinx, 2018.

[19] A. Thompson, "On the automatic design of robust electronics through artificial evolution," in *Evolvable Systems: From Biology to Hardware*. Springer Berlin Heidelberg, 1998, pp. 13–24.

[20] R. Salvador, A. Otero, J. Mora, E. de la Torre, L. Sekanina, and T. Riesgo, "Fault tolerance analysis and self-healing strategy of autonomous, evolvable hardware systems," in *2011 International Conference on Reconfigurable Computing and FPGAs*, IEEE. IEEE, nov 2011, pp. 164–169.

[21] J. Francq and F. Frick, "Introduction to hardware trojan detection methods," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015*, IEEE. IEEE Conference Publications, 2015, pp. 770–775.

[22] Z. Collins, B. King, R. Jha, D. Kapp, and A. Ralescu, "Evolvable hardware for security through diverse variants," in *2019 IEEE National Aerospace and Electronics Conference (NAECON)*, IEEE. IEEE, jul 2019, pp. 257–261.