2016 11th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)

Evolvable Hardware in FPGAs: Embedded tutorial

Ruben Salvador
Research Center on Software Technologies and Multimedia Systems for Sustainability - CITSEM
Universidad Politecnica de Madrid (UPM)
Madrid, Spain
Email: ruben.salvador@upm.es

***Abstract*—The roots of adaptive and bio-inspired computing systems date back to the early days of computers. Evolvable Hardware is one of the various approaches that emerged in the last two decades with this objective. By bringing together Reconfigurable Computing (RC) and Evolutionary Computing (EC), i.e., a self-reconfigurable computing substrate and an algorithm working as an adaptation engine, a new species of hardware arose that should be able to evolve and achieve self-adaptation through its operational life. This paper introduces the field of EHW using FPGAs, classifies the different approaches and surveys the main works reported. Emphasis is put to highlight the works that have addressed Dynamic Partial Reconfiguration (DPR) in EHW trying to overcome the limitations imposed by the state of the technology in order to keep advancing the field.**

I. Introduction to EHW

Behind the suggestive name of ***Evolvable Hardware*** (EHW) lies a rather attractive idea: the design of self-adaptive hardware able to autonomously and dynamically perform self-reconfiguration to modify its computational behaviour to adapt to changing functional requirements, to overcome from faults and degradation and to recover from unexpected operating conditions and disturbances from the environment [1]. In order to achieve such a general objective, two main elements are needed: a self-reconfigurable computing device able to be dynamically changed at runtime and an embedded adaptation engine. If the latter is an Evolutionary Algorithm (EA) [2] a new *species* of hardware that is *evolvable* emerges.

EHW is a field at the confluence of reconfigurable devices, autonomous systems, artificial intelligence and automatic design [1]. It was proposed [3] as the automatic design of hardware using EAs, i.e., as a new way of circuit design by just providing a black-box behavioural description of the desired input/output relationship to the EA. But its real projection promotes it as a third generation of hardware in terms of flexibility and fault tolerance [1], [4] given its capabilities of self-reconfiguration and evolvability, i.e., of self-adaptation, to turn hardware adaptable through reconfiguration into a self-evolvable entity driven by an EA.

EAs are inspired from Darwin's principles of natural evolution and comprise a set of population-based algorithms considered as universal problem solvers. Each individual of the population encodes a candidate solution to the problem into a structure known as *chromosome*, which is formed by so-called *genes*; this is the *genotype* or representation space of the EA. Hence, genotypes encode the physical manifestations that will result from a transformation process into the *phenotype* or solution space. This way, chromosomes, i.e., the representations of candidate solutions, are transformed through a mapping process or *growth function* into actual solutions in order to evaluate the whole population by using a function that assigns a *fitness* measure to each individual to assess how well each solves the problem. This process is repeated a number of times called *generations* looking for incremental fitness improvements by combining the fittest individuals to produce a new (hopefully fitter) offspring population.

Within EHW, chromosomes contain representations of circuits and genes a set of components and their interconnections, typically encoded using bit strings, integer lists or parse-tree representations. Fig. 1 shows a graphical representation of EHW using a reconfigurable device. The individuals of the population feature reconfigurable circuit descriptions that are evaluated by reconfiguring the device with each of them and computing the fitness function. Those individuals with a higher fitness have a higher chance to be selected as parents for the next generation of circuits. These parents undergo the genetic operators known as *crossover* and *mutation* (see Fig. 2) to create the new offspring in the expectation that the inherited traits from each parent and the random mutations improve the population fitness. This process is repeated until a stop criterion is met, eventually obtaining a circuit that satisfies the required specification, or evolution might also be kept active for continuous system adaptation.

This paper introduces and surveys the field of EHW using FPGAs. Section II contains a classification framework while Section III features a brief historical perspective to introduce EHW limitations posed by FPGA DPR technology. A survey of the state of the art of FPGA-based evolvable systems with remarks on the use of DPR is included in section IV before final concluding remarks are addressed in Section V.

II. A Classification Framework for EHW

There are two approaches to EHW that yield an initial classification into two sub-fields: ***evolutionary hardware design*** (or evolutionary electronics or evolutionary circuit design), as a design methodology based on EAs by which a circuit description fulfilling the required specifications (sometimes featuring innovative solutions) is obtained to be afterwards implemented in the target technology; and ***adaptive hardware***, as a way to provide a means for self-adaptation of the underlying computational hardware (substrate) of reconfigurable systems. Three different categories can be considered for

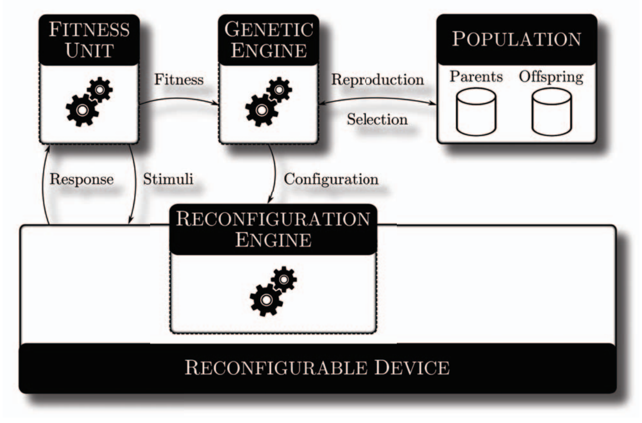
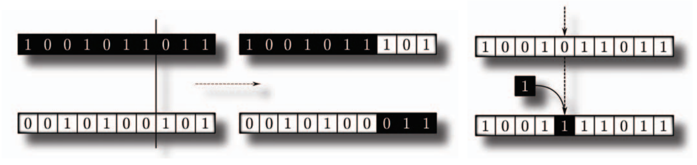


Fig. 1. Evolvable hardware. Adapted from Sekanina [5].



(a) (b)

Fig. 2. Genetic operators for binary chromosome representations. Crossover (a) combines characteristics from selected parents by exchanging their genes to generate two new offspring. Mutation (b) usually assigns a small mutation probability to each gene to randomly modify (bit-flipping for binary genes) a few offspring genes.

further classification purposes: (i) evaluation medium; (ii) evolution scope; and (iii) granularity of evolution.

1) Evaluation medium: according to where the evaluation of candidate circuits is done, *extrinsic* and *intrinsic* EHW can be distinguished:

- ***Extrinsic EHW***: framed in the sub-field of evolutionary hardware design, it is also known as *offline hardware evolution*. Evaluation takes place in a computing system other than the final device, for example a PC, using models and simulators of the target technology.
- ***Intrinsic EHW***: also known as *online hardware evolution*, the evaluation of candidate solutions takes place in the target device With this method, as opposed to simulation, the evolutionary search process performed by the EA can mainly benefit: from the speedup derived by the evaluation taking place in hardware; and from the exploitation of the unique physical characteristics of a specific piece of device achieved by evolution dealing directly with the reconfigurable substrate.

2) Evolution scope: if the long-term scope of evolution is considered, ***evolved hardware*** and ***adaptive hardware*** can be distinguished:

- ***Evolved hardware***: merges all the approaches to *evolutionary hardware design*, either by extrinsic or intrinsic evolution, and is probably the approach actually followed by most researchers. The EA can run in a PC, but in case of intrinsic evolution in reconfigurable devices with embedded processors, it might also be running in the target device for convenience and to help speeding up reconfiguration and hence evaluation. The final solution (usually the fittest circuit) can be used to configure other devices in case the behaviour of the evolved solution is perfectly reproducible; this is, in general, only true for digital, synchronous systems implemented in FPGAs.
- ***Adaptive hardware***: the EA is embedded in the final system and, along with certain *self/environment-awareness*, autonomous adaptation capabilities are provided through a kind of continuous evolution mechanism that maintains the expected performance. If no human intervention is needed for adaptation, which involves the device having self-reconfiguration capabilities too, ***self-adaptive hardware*** emerges. Ideally, it should be able to adapt to changes in the input signal conditions, the environment, the specifications, and even to defects and failures (temporary and permanent) in its very own substrate.

3) Granularity of evolution: refers to what are genes in chromosomes actually representing. It is related to an important issue in EHW, the ***scalability of representation*** [6], which basically considers how longer chromosomes to represent more complex circuits affect search efficiency due to the solution space becoming larger. In practice, this issue has greatly limited the complexity of evolved circuits. A classification [5] from this perspective is:

- ***Transistor/Component-level evolution*** [7]: uses basic analogue blocks (transistors, resistors, wires, etc.) and operational amplifiers to evolve analogue circuits and innovative topologies for logic gates.
- ***Gate-level evolution*** [7]: exploited to evolve digital circuits using basic logic gates and wires.
- ***Bitstream manipulation***: used with FPGAs to directly manipulate the configuration bitstream, was mostly abandoned when these modifications turned unsafe for the device integrity as introduced in the following section.
- ***Function-level evolution*** [8]: introduces domain knowledge into the chromosomes by defining application-specific functional blocks at higher granularity levels (adders, shifters, comparators, etc.) and ad-hoc datapath bitwidths.

In order to overcome the scalability problem, four main proposals can be found in the literature: (i) ***function-level evolution***; (ii) ***incremental evolution*** [9], [10], as a *divide-and-conquer* approach in which the circuit to be evolved is decomposed in smaller sub-circuits that are evolved separately; (iii) ***development*** [11], [12], which proposes an indirect mapping process called *morphogenesis* to construct a circuit from its genotype by encoding a set of rules into it, and hence enabling the construction of more complex circuits than using direct, proportional mappings; and (iv) ***modularisation*** [12], [13], which inspired in SW sub-routines, proposes the use of certain EAs that allow the dynamic creation, evolution and destroy of reusable modules, making evolution of larger circuits easier than from scratch.

III. EHW and FPGA Reconfiguration: Brief Historical Survey

Two are the works considered as the trigger of the research in EHW. The first of them [3] is an example of extrinsic evolution using a simulator to evolve a 6-to-1 multiplexer from just a behavioural description of its input/output relationship; the evolved circuit was implemented in a GAL16Z8 device, where final, real tests were performed. The second [14], considered to be the first attempt with FPGAs, is an example of intrinsic evolution that was accomplished with the discontinued Xilinx XC6216. Working over a subset of the available resources, unconstrained evolution, understood as the direct and unrestricted manipulation of the configuration bitstream, was performed by the EA. The work in particular designed an EA that run in a PC connected to the FPGA to reconfigure it and evaluate each of the candidate circuits in order to evolve a tone discriminator.

EHW in FPGAs have been closely tied to the state of reconfiguration technology and to the possibility of whether *internal reconfiguration* was available to achieve self-reconfiguration, or support for *external reconfiguration* from other device or PC was needed. Xilinx XC6200 FPGA family, although missing the important internal reconfiguration feature, was once considered the best FPGA-based evolvable platform mainly because: (i) the configuration bitstream was publicly available allowing the creation of efficient representations; and (ii) its routing resources were based on multiplexers, enabling random safe modifications of the configuration bitstream.

However, the state of reconfiguration technology changed after this family was discontinued and replaced by Virtex families in the late 90s, posing several restrictions to embrace DPR as a standard practice for self-reconfigurable systems. This turned into a major obstacle for advancements in EHW due to: (i) the MUX-based routing approach being abandoned in favour of a switch-matrix, impeding random, safe bitstream modifications; (ii) the lack of adequate DPR tools and design flows at those dates, including the impossibility to relocate modules (partial bitstreams) to different positions of the FPGA; (iii) the increasing bitstream sizes and unknown formats; and (iv) the insufficient reconfiguration speed to achieve near real-time self-adaptation.

In order to overcome these limitations, a proposal in the first 2000s allowed to keep investigating in this field while DPR technology kept maturing, the ***Virtual Reconfigurable Circuit*** (VRC) [15]. Initially proposed for array-like processing structures, VRCs are derived from ***Cartesian Genetic Programming*** (CGP) [16], [17], thus being VRCs the specific form CGP usually takes in FPGA implementations. CGP was invented as a new form of Genetic Programming from the previous work of the authors on the evolution of digital circuits, so it has been widely used by EHW practitioners for many different applications [17]. It describes a digital circuit as a directed graph, with a simple integer genotype that describes the functionality and connections of each node of the tree. This genotype is mapped on a kind of grid of computing nodes, with *Cartesian* referring to the coordinates that locate each node within the grid. It often uses a simple Evolution Strategy (ES) [2] with small populations of 1 parent and between 4 and 10 children. Mutation at low rates is the only evolutionary operator since recombination does not seem to affect the search in an effective manner.

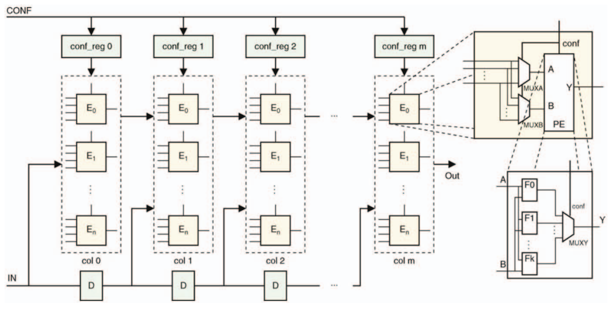


Fig. 3. Virtual Reconfigurable Circuit architecture [5].

A VRC in an FPGA (Fig. 3) is a virtual layer that defines an application specific reconfigurable circuit on top of an FPGA fabric to reduce the complexity of the reconfiguration process and increase its speed (compared to native reconfiguration). It is specified using standard HDL descriptions and contains: (i) an array of Configurable Functional Blocks (CFBs) acting as the nodes of the array such that each CFB contains all the required functions inside a Processing Element (PE)[1]; (ii) a set of functionality multiplexers (one per CFB) to select the required function from the PE; (iii) a set of routing multiplexers that provide inter-node connectivity; and (iv) a large register as the configuration memory. VRC reconfiguration is very fast given it only involves writing this register, which drives the selection signals of all the multiplexers (MUXs). However, large overheads are introduced by this virtual layer: (i) an area overhead due to the simultaneous implementation of every function in every node of the array, plus the MUXs; and (ii) a delay overhead due to these MUXs, which also reduces maximum frequency.

During the first 2000s Xilinx introduced the Internal Configuration Access Port (ICAP) [18] which enabled ***native, self-reconfiguration*** by providing a way to trigger reconfiguration from inside the FPGA. This port, missing in previous MUX-based devices, was a desired step-ahead in order to achieve on-chip, structural circuit evolution. Sadly, reconfiguration speed was yet too slow for EHW applications and, besides, the rest of limiting factors mentioned previously was still present. Altogether, made the use of native reconfiguration in FPGAs being held back, so DPR usage would greatly remained in the background within EHW until technology improved. Since Xilinx Zynq devices appeared in the last years, the new Processor Configuration Access Port (PCAP) for the embedded ARM processors was introduced to address these current heterogeneous devices that FPGAs are.

[1]PEs might be defined, for instance, at logic gate level (and, or, xor, etc.) to evolve combinational circuits or at the function level (adders, subtracters, shifters, etc.) for other applications such as image/signal processing

IV. FPGA-Based Evolvable Systems

This section features a survey of works and a discussion on architectural issues regarding evolvable systems in FPGAs using internal reconfiguration, with a focus on native reconfiguration (within the literature, deeper surveys [5], [7], [12], [19]–[22] that also consider external reconfiguration in FPGAs, evolution on other devices and extrinsic EHW, can be found). Main works are collected in Table I, which covers both EHW categories dealing with intrinsic evolution in FPGAs introduced in Section II: evolutionary hardware design and adaptive HW. Whichever the objective is, the following components can always be identified: (i) a reconfigurable area featured by a set of reconfigurable elements determined by the problem domain; (ii) the EA in charge of driving the evolutionary process; (iii) a HW evaluation module for the fitness computation unit (or k multiple instances for further speed-ups, indicated by '$k\times$HW'); and (iv) a reconfiguration controller. Columns 3, 4 and 5 in Table I compile the information related to (i), (ii) and (iii), respectively.

Typical architectures for FPGA-based evolvable systems closely follows general IP-based designs for FPGAs using a central processor, which holds the EA, and different hard-IP cores, among which an evolvable component is included. Complete hardware evolution experiments in which the EA is also implemented in HW have also been conducted [23]–[25], [29], [32], [33], but given fitness computation is the most time consuming task, no significant speed-ups are obtained with a HW implementation of the EA for most applications reported.

Regarding VRCs, different applications have been reported: image filters [24], [27], [32]; classifiers [28], [31], [32]; cellular automata [26]; combinational logic circuits [29]; hash functions for packet classifiers [25]; and Multiple Constant Multipliers [33], among many others. It is worth to mention too the work in HW acceleration of CGP [30], which can be further increased by replicating the fitness unit [29], [35].

To address typical VRC-CGP overheads, an architecture based on traditional systolic arrays (SA) together with a DPR engine with relocation capabilities has been demonstrated for image filters [40]. Area is reduced by only having one PE per node at the same time in the FPGA while delay is improved by restricting PE interconnection to the four closest neighbours and thus eliminating routing multiplexers, which at the same time favours local, short interconnections that further reduce propagation delay and improve dynamic scalability [44], [45]. Other works [39], [42] propose hybrid VRC-DPR methods for CGP structures that keep virtual reconfiguration using multiplexers for interconnections and different forms of PE reconfiguration as introduced in the following analysis on the use of DPR in EHW.

*1) **FPGA native reconfiguration in EHW**:* given some of the issues with DPR being progressively solved by FPGA manufacturers, some works began to appear in the last decade. These can be classified according to the following categories (all using Xilinx FPGAs):

- ***Library of pre-sinthesized partial bitstreams***: (i) [46] DPR through ICAP of Category Detection modules (CDM) with different number of functional units (FU) per CDM for previous VRC-based classifiers [28]; (ii) [36] DPR through ICAP to change the functionality of nodes (or columns of nodes) of VRC-CGP-like structures with static routing for small combinational circuits and [37] for more complex circuits and classifiers; (iii) [40] DPR of the PEs of an evolvable SA using an optimised ICAP controller with relocation capabilities and a self-healing mechanism for image filters.
- ***LUT reconfiguration through shift register functionality (SRLUTs)***, changes LUT functions by directly shifting configuration bits into them[2]: (i) [34] online incremental evolution of FUs in the CDMs to reduce area and reconfiguration time of evolvable classifiers (of previous ICAP [46] and VRC [28] versions); (ii) [38] low-level evolution of nodes from [36].
- ***DPR-based LUT reconfiguration through direct bitstream manipulation***: (i) [26] evolvable Random Boolean Network with arbitrary connectionism by using LUTs as multiplexers; (ii) [41] fine-grained LUT reconfiguration in evolvable classifiers (updates [34]); (iii) [36] low-level evolution of node functionality and routing of VRC-CGP-like structures; (iv) [43] optimised PE implementation in evolvable SAs, improving area cost and maximum frequency from [40].
- ***Hybrid VRC-DPR methods*** for CGP-based image filters using PCAP. Virtual reconfiguration using multiplexers (but adding flip-flops at the outputs to reduce delay impact) is used for PEs interconnection, while PE reconfiguration is done through: (i) [39] DPR using pre-sinthesized partial bitstreams; (ii) [42] DPR-based LUT reconfiguration for approximate PEs by direct bitstream manipulation.

Up to date, two ([42], [43]) can be considered the most advanced implementations of EHW systems using DPR of FPGAs dealing with image filters for noise removal (using PEs at the function level, such as adders, subtracters, etc.). Regarding area: the SA [43] requires 16 LUTs and 8 flip-flops for each PE of the evolvable SA from [40] (100% of the available reserved logic resources for each PE is used so no further reduction is possible) in Virtex 5 devices; the hybrid VRC-DPR approach [42] requires 8 LUTs and 8 flip-flops per PE for a set of *approximate* PEs in a Zynq-7000 device. Regarding speed: the SA [43] works at 500 MHz, the fastest work reported so far to the best of this author's knowledge, achieving 20 000 circuit reconfigurations and evaluations per second on a system with a single array or up to 80 000 with 8 arrays in parallel[3]; the hybrid VRC-DPR approach [42] works

[2]A serious drawback of this reconfiguration strategy is the decreasing number of LUTs with this operating mode: 100% in Virtex-II Pro and around 25% in Virtex 5.

[3]Since Virtex 5 devices feature around 60% smaller reconfiguration frames (which is the minimum reconfigurable unit) compared to Zynq-7000, an implementation in this platform would probably increase the reconfiguration time of the SA. This means the 20 000 evaluations reported [43] would be fewer in a Zynq device, but in any case probably well over the 8 700 figure.

TABLE I
FPGA IMPLEMENTATIONS OF EVOLVABLE DIGITAL SYSTEMS.

Reference	Application	Platform	EA	Fitness
[23]	FIR filters	Register values	HW	HW
[24]	Image filters	VRC@XC2V3000	HW	HW
[25]	Hash functions	VRC@XC4VFX20	HW	HW
[26]	RBN[c]/Cellular automaton	Virtex II CLB & ICAP	MicroBlaze	MicroBlaze
[27]	Image filters	VRC@XC2VP50	PowerPC	HW
[28]	Sonar spectrum classifier	VRC@XC2VP30	PowerPC	HW
[29]	Arith. circuits	VRC@XCV2000E	HW	2×HW
[30]	CGP accelerator	VRC@XC2VP50	PowerPC	HW
[31]	Face image classif.	VRC@XC2VP30	MicroBlaze	HW
[32]	Image filters, classif.	VRC@XCV2000E	HW	HW
[33]	Multiple Constant Coefficient Multipliers[e]	VRC@XC2VP50	HW	HW
[34]	Face classifiers	SRLUTs @XC2VP	PowerPC/MicroBlaze	HW
[35]	CGP accelerator	VRC@XC5VFX100T	PowerPC	4×HW
[36]	Small comb. circuits	Virtex 4 logic & ICAP	PowerPC	HW
[37]	Classifiers and comb. circuits	Virtex 4 logic & ICAP	PowerPC	HW
[38]	Same as [37]	SRLUTs-CFGLUT5[i]@Virtex 5	MicroBlaze	HW
[39]	Image filters	VRC&Module-based-DPR[j] @Zynq-7000	ARM Cortex-A9	HW
[40]	Image filters	Virtex 5 logic	MicroBlaze	HW
[41]	Face/sonar classif.	LUT-based-DPR[k](ICAP)@Virtex	MicroBlaze	HW
[42]	Image filters	VRC&LUT-based-DPR (PCAP)@Zynq-7000	ARM Cortex-A9	HW
[43]	Image filters	LUT-based-DPR (ICAP)@Virtex 5	MicroBlaze	HW

[c] Random Boolean Networks [e] Multiple Constant Coefficient Multipliers
[j] DPR using partial bitstream describing functional modules; requires Xilinx module-based methodology.
[k] Lookup table DPR through changing LUT configuration data by direct bitstream manipulation.

at 300 MHz, which results in 8 700 evaluations per second or up to 30 000 on 6 arrays.

V. CONCLUDING REMARKS

This paper has surveyed the field of EWH and offered a general classification framework. The focus has been set in the implementation of FPGA-based evolvable systems, for which a historical perspective has been introduced analysing the deep dependence with the state of DPR technology.

Along with the survey, some of the limitations that prevent reaching the promise of self-adaptation have been analysed too. However, there are other issues that also need to be solved [4] which have not yet been mentioned in this paper, among which the most important are: (i) unpredictability of finding suitable configurations (and do it on real-time) due to the stochastic nature of EAs; (ii) lack of formal validation methods for online, safety critical evolvable systems; and (iii) specification of safe, dynamic fitness computation in its interaction with the environment.

All these limitations contribute to a big open issue in the field: finding real-world applications of EHW with industrial acceptance and market penetration. In this regard, opportunities might be found if EHW is used to provide new or improve traditional solutions to complex challenges like fault and defect tolerance and dynamic energy management [4] rather than trying to overcome complex applications. Highly unattended systems and space systems (both manned and unmanned) might benefit from this. As an example, fault tolerance has to be considered, given the inherent capability shown by EHW [47], [48]. This promotes evolutionary fault recovery in FPGAs as a good niche of application. Although most works have been addressed using simulators or external reconfiguration, some experiments were reported addressing autonomous self-healing and graceful degradation from cumulative faults in FPGAs [45], [49]–[51].

In any case, considering the challenges that self-adaptation poses (whose analysis is out of the scope of this paper), the quest for fully autonomous systems seems quite demanding to be tackled by a single discipline, so the path to achieve it will surely need contributions from other fields besides EHW.

REFERENCES

[1] A. Stoica, “Evolvable hardware for autonomous systems,” Tutorial at *IEEE Congr. Evol. Comput.* (CEC), 2004.

[2] A. Eiben and J. Smith, *Introduction to Evolutionary Computing*, ser. Nat. Comp. Ser. Springer Berlin Heidelberg, 2003.

[3] 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: Proc. 2nd Int. Conf. on Simulation of Adaptive Behavior*, MIT Press, 1993, pp. 417–424.

[4] P. C. Haddow and A. M. Tyrrell, “Challenges of evolvable hardware: Past, present and the path to a promising future,” *Genet. Program. Evol. Mach.*, vol. 12, no. 3, pp. 183–215, 2011.

[5] L. Sekanina, “Evolvable hardware,” in *Handbook of Natural Computing*, G. Rozenberg, T. Bäck, and J. N. Kok, Eds., Springer Berlin Heidelberg, 2012, pp. 1657–1705.

[6] T. G. W. Gordon, “Exploiting Development to Enhance the Scalability of Hardware Evolution,” PhD thesis, Dept. Comp. Sc., University College London, 2005. https://www.bcs.org/upload/pdf/tgordon.pdf.

[7] R. S. Zebulum, M. A. C. Pacheco, and M. M. B. R. Vellasco, *Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms*, 1st ed., ser. Int. Ser. Comput. Intell. 22. CRC Press, Inc., 2001, 304 pp.

[8] M. Murakawa, S. Yoshizawa, I. Kajitani, T. Furuya, M. Iwata, and T. Higuchi, “Hardware evolution at function level,” in *Parallel Problem Solving from Nature – PPSN IV*, ser. Lect. Notes Comput. Sc. Vol. 1141, Springer Berlin Heidelberg, 1996, pp. 62–71.

[9] J. Torresen, “A Divide-and-Conquer Approach to Evolvable Hardware,” in *Evolvable Systems: From Biology to Hardware, Proc. 2nd Int. Conf. (ICES)*, ser. Lect. Notes Comput. Sc. Vol. 1478, Springer Berlin Heidelberg, 1998, pp. 57–65.

[10] ——, "A Scalable Approach to Evolvable Hardware," *Genet. Program. Evol. Mach.*, vol. 3, no. 3, pp. 259–282, 2002.

[11] H. Kitano, "Morphogenesis for evolvable systems," in *Towards Evolvable Hardware*, ser. Lect. Notes Comput. Sc. Vol. 1062, Springer Berlin Heidelberg, 1996, pp. 99–117.

[12] J. R. Koza, F. H. Bennett III, D. Andre, and M. A. Keane, *Genetic Programming III, Darwinian Invention and Problem Solving*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, 1154 pp.

[13] J. A. Walker and J. F. Miller, "The Automatic Acquisition, Evolution and Reuse of Modules in Cartesian Genetic Programming," *IEEE Trans. Evol. Comput.*, vol. 12, no. 4, pp. 397–417, Aug. 2008.

[14] A. Thompson, "Silicon Evolution," in *Proc. 1st Annu. Conf. Genet. Prog.*, MIT Press, 1996, pp. 444–452.

[15] L. Sekanina, "Virtual Reconfigurable Circuits for Real-World Applications of Evolvable Hardware," in *Evolvable Systems: From Biology to Hardware, Proc. 5th Int. Conf. (ICES)*, ser. Lect. Notes Comput. Sc. Vol. 2606, Springer Berlin Heidelberg, Mar. 17–20, 2003, pp. 186–197.

[16] J. F. Miller and P. Thomson, "Cartesian Genetic Programming," in *Genetic Programming, Proc. European Conf. Genet. Prog., (EuroGP)*, ser. Lect. Notes Comput. Sc. Vol. 1802, Springer Berlin Heidelberg, 2000, pp. 121–132.

[17] J. F. Miller, Ed., *Cartesian Genetic Programming*, ser. Nat. Comp. Ser. Springer Berlin Heidelberg, 2011.

[18] B. Blodget, P. James-Roxby, E. Keller, S. McMillan, and P. Sundararajan, "A Self-reconfiguring Platform," in *Field Programmable Logic and Applications, Proc. 13th Int. Conf., FPL*, ser. Lect. Notes Comput. Sc. Vol. 2778, Springer Berlin Heidelberg, 2003, pp. 565–574.

[19] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, and J. Y. Guido, *Genetic Programming IV, Routine Human-Competitive Machine Intelligence*, ser. Genet. Progr. Ser. Springer US, 2005, vol. 5, 624 pp.

[20] G. W. Greenwood and A. M. Tyrrell, *Introduction to Evolvable Hardware*, D. B. Fogel, Ed., ser. IEEE Press Ser. Comput. Intell. Wiley-IEEE Press, 2006, 208 pp.

[21] T. Higuchi, Y. Liu, and X. Yao, Eds., *Evolvable Hardware*, ser. Gen. Evol. Comp. Ser. Springer US, 2006.

[22] R. Salvador, "Parametric and Structural Self-Adaptation of Embedded Systems using Evolvable Hardware," PhD thesis, Universidad Politecnica de Madrid, Madrid, 2015. http://oa.upm.es/39354/.

[23] G. Tufte and P. C. Haddow, "Evolving an Adaptive Digital Filter," in *Proc. 2nd NASA/DoD Workshop on Evolvable Hardware*, IEEE, 2000, pp. 143–150.

[24] T. Martinek and L. Sekanina, "An Evolvable Image Filter: Experimental Evaluation of a Complete Hardware Implementation in FPGA," in *Evolvable Systems: From Biology to Hardware, Proc. 6th Int. Conf. (ICES)*, ser. Lect. Notes Comput. Sc. Vol. 3637, Springer Berlin Heidelberg, Sep. 12–14, 2005, pp. 76–85.

[25] R. Salomon, H. Widiger, and A. Tockhorn, "Rapid Evolution of Time-Efficient Packet Classifiers," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, IEEE, 2006, pp. 2793–2799.

[26] A. Upegui and E. Sanchez, "Evolving Hardware with Self-reconfigurable connectivity in Xilinx FPGAs," in *Proc. 1st NASA/ESA Conf. Adaptive Hardware and Systems (AHS)*, IEEE, 2006, pp. 153–162.

[27] Z. Vasicek and L. Sekanina, "An evolvable hardware system in Xilinx Virtex II Pro FPGA," *Int. J. Innovative Comput. Appl. (IJICA)*, vol. 1, no. 1, pp. 63–73, 2007.

[28] K. Glette, J. Torresen, and M. Yasunaga, "An Online EHW Pattern Recognition System Applied to Sonar Spectrum Classification," in *Evolvable Systems: From Biology to Hardware, Proc. 7th Int. Conf. (ICES)*, ser. Lect. Notes Comput. Sc. Vol. 4684, Springer Berlin Heidelberg, Sep. 21–23, 2007, pp. 1–12.

[29] J. Wang, C. H. Piao, and C. H. Lee, "Implementing Multi-VRC Cores to Evolve Combinational Logic Circuits in Parallel," in *Evolvable Systems: From Biology to Hardware, Proc. 7th Int. Conf. (ICES)*, ser. Lect. Notes Comput. Sc. Vol. 4684, Springer Berlin Heidelberg, Sep. 21–23, 2007, pp. 23–34.

[30] Z. Vasicek and L. Sekanina, "Hardware Accelerators for Cartesian Genetic Programming," in *Genetic Programming, Proc. 11th European Conf., EuroGP*, ser. Lect. Notes Comput. Sc. Vol. 4971, Springer Berlin Heidelberg, Mar. 26–28, 2008, pp. 230–241.

[31] K. Glette, "Design and Implementation of Scalable Online Evolvable Hardware Pattern Recognition Systems," PhD thesis, Faculty of Mathematics and Natural Sciences, University of Oslo, 2008. http://urn.nb.no/URN:NBN:no-20883.

[32] J. Wang, Q. Chen, and C. Lee, "Design and implementation of a virtual reconfigurable architecture for different applications of intrinsic evolvable hardware," *IET Comput. Digit. Tech.*, vol. 2, no. 5, pp. 386–400, 2008.

[33] Z. Vasicek, M. Zadnik, L. Sekanina, and J. Tobola, "On Evolutionary Synthesis of Linear Transforms in FPGA," in *Evolvable Systems: From Biology to Hardware, Proc. 8th Int. Conf. (ICES)*, ser. Lect. Notes Comput. Sc. Vol. 5216, Springer Berlin Heidelberg, Sep. 21–24, 2008, pp. 141–152.

[34] K. Glette, J. Torresen, and M. Hovin, "Intermediate Level FPGA Reconfiguration for an Online EHW Pattern Recognition System," in *Proc. NASA/ESA Conf. Adaptive Hardware and Systems (AHS)*, IEEE, 2009, pp. 19–26.

[35] Z. Vasicek and L. Sekanina, "Hardware Accelerator of Cartesian Genetic Programming with Multiple Fitness Units," *Comput. Inform.*, vol. 29, no. 6, pp. 1359–1371, 2010.

[36] F. Cancare, M. D. Santambrogio, and D. Sciuto, "A Direct Bitstream Manipulation Approach for Virtex4-based Evolvable Systems," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, IEEE, 2010, pp. 853–856.

[37] F. Cancare, D. B. Bartolini, M. Carminati, D. Sciuto, and M. D. Santambrogio, "On the Evolution of Hardware Circuits via Reconfigurable Architectures," *ACM T. Reconfigurable Technol. Syst.*, vol. 5, no. 4, 22:1–22:22, Dec. 2012.

[38] D. B. Bartolini, M. Carminati, F. Cancare, M. D. Santambrogio, and D. Sciuto, "Hera project's holistic evolutionary framework," in *IEEE 27th Int. Parallel and Distributed Processing Symp. Workshops (IPDPSW)*, IEEE, May 2013, pp. 231–238.

[39] R. Dobai and L. Sekanina, "Image Filter Evolution on the Xilinx Zynq Platform," in *Proc. NASA/ESA Conf. Adaptive Hardware and Systems (AHS)*, IEEE, Jun. 2013, pp. 164–171.

[40] R. Salvador, A. Otero, J. Mora, E. de la Torre, T. Riesgo, and L. Sekanina, "Self-Reconfigurable Evolvable Hardware System for Adaptive Image Processing," *IEEE Trans. Comput.*, vol. 62, no. 8, pp. 1481–1493, Aug. 2013.

[41] K. Glette and P. Kaufmann, "Lookup Table Partial Reconfiguration for an Evolvable Hardware Classifier System," in *IEEE Congr. Evol. Comput. (CEC)*, IEEE, Jul. 2014, pp. 1706–1713.

[42] R. Dobai and L. Sekanina, "Low-Level Flexible Architecture with Hybrid Reconfiguration for Evolvable Hardware," *ACM T. Reconfigurable Technol. Syst.*, vol. 8, no. 3, 20:1–20:24, May 2015.

[43] J. Mora, A. Otero, E. de la Torre, and T. Riesgo, "Fast and compact evolvable systolic arrays on dynamically reconfigurable FPGAs," in *10th Int. Symp. on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, IEEE, Jun. 2015, pp. 1–7.

[44] A. Gallego, J. Mora, A. Otero, R. Salvador, E. de la Torre, and T. Riesgo, "A Novel FPGA-based Evolvable Hardware System Based on Multiple Processing Arrays," in *IEEE 27th Int. Parallel and Distributed Processing Symp. Workshops (IPDPSW)*, IEEE, May 2013, pp. 182–191.

[45] A. Gallego, J. Mora, A. Otero, E. de la Torre, and T. Riesgo, "A Scalable Evolvable Hardware Processing Array," in *Int. Conf. Reconfigurable Computing and FPGAs (ReConFig)*, IEEE, Dec. 2013, pp. 1–7.

[46] J. Torresen, G. A. Senland, and K. Glette, "Partial Reconfiguration Applied in an On-line Evolvable Pattern Recognition System," in *NORCHIP*, IEEE, Nov. 2008, pp. 61–64.

[47] A. Thompson, "Evolving fault tolerant systems," in *1st Int. Conf. Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA)*, IET, Sep. 1995, pp. 524–529.

[48] A. M. Tyrrell, G. Hollingworth, and S. L. Smith, "Evolutionary Strategies and Intrinsic Fault Tolerance," in *Proc. 3rd NASA/DoD Workshop on Evolvable Hardware*, IEEE, Jul. 2001, pp. 98–106.

[49] 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 *Int. Conf. Reconfigurable Computing and FPGAs (ReConFig)*, IEEE, Nov. 2011, pp. 164–169.

[50] R. F. DeMara, K. Zhang, and C. A. Sharma, "Autonomic fault-handling and refurbishment using throughput-driven assessment," *Applied Soft Computing*, vol. 11, no. 2, pp. 1588–1599, 2011.

[51] R. A. Ashraf and R. F. DeMara, "Scalable FPGA Refurbishment Using Netlist-Driven Evolutionary Algorithms," *IEEE Trans. Comput.*, vol. 62, no. 8, pp. 1526–1541, Aug. 2013.