Genet Program Evolvable Mach (2011) 12:183–215
DOI 10.1007/s10710-011-9141-6

Challenges of evolvable hardware: past, present and the path to a promising future

Pauline C. Haddow · Andy M. Tyrrell

Received: 23 September 2010 / Published online: 19 June 2011

Abstract Nature is phenomenal. The achievements in, for example, evolution are everywhere to be seen: complexity, resilience, inventive solutions and beauty. Evolvable Hardware (EH) is a field of evolutionary computation (EC) that focuses on the embodiment of evolution in a physical media. If EH could achieve even a small step in natural evolution's achievements, it would be a significant step for hardware designers. Before the field of EH began, EC had already shown artificial evolution to be a highly competitive problem solver. EH thus started off as a new and exciting field with much promise. It seemed only a matter of time before researchers would find ways to convert such techniques into hardware problem solvers and further refine the techniques to achieve systems that were competitive with or better than human designs. However, 15 years on—it appears that problems solved by EH are only of the size and complexity of that achievable in EC 15 years ago and seldom compete with traditional designs. A critical review of the field is presented. Whilst highlighting some of the successes, it also considers why the field is far from reaching these goals. The paper further redefines the field and speculates where the field should go in the next 10 years.

Keywords Evolvable hardware · Future technology · Scalability · Computation medium · Review

P. C. Haddow (✉)
CRAB Lab, Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway
e-mail: pauline@idi.ntnu.no

A. M. Tyrrell
Intelligent Systems Group, Department of Electronics, University of York, York, UK
e-mail: amt@ohm.york.ac.uk

1 Introduction

Yao and Higuchi published a paper in 1999 entitled “Promises and Challenges of Evolvable Hardware” which reviewed the progress and possible future direction of what was then a new field of research, Evolvable Hardware [1]. A little more than 10 years on, this paper considers the progress of this research field, both in terms of the successes achieved to date and also the failure to fulfil the promises of the field highlighted in the 1999 paper. Through a critical review of the field, the authors’ intention is to provide a realistic status of the field today and highlight the fact that the challenges remain, redefine the field (what should be considered as Evolvable Hardware) and propose a revised future path.

In the mid 1990s, researchers began applying Evolutionary Algorithms (EAs) to a computer chip that could dynamically alter the hardware functionality and physical connections of its circuits [2–10]. This combination of EAs with programmable electronics, e.g. Field Programmable Gate Arrays (FPGAs) & Field Programmable Analogue Arrays (FPAAs), spawned a new field of EC called Evolvable Hardware.

The EH field has since expanded beyond the use of EAs on simple electronic devices to encompass many different combinations of EAs and biologically inspired algorithms (BIAs) with various physical devices or simulations thereof. Further, the challenges inherent in EH have led researchers to explore new BIAs that may be more suitable as techniques for EH.

In this paper we will define the field of EH and split the field into the two related, but different sub-fields: Evolvable Hardware Design (EHD) and Adaptive Hardware (AH).

Evolvable Hardware, as the name suggests, should have a connection to embodiment in a real device. However, in a number of EH papers, results produced are interpreted as hardware components e.g. logic gates, illustrated as a circuit diagram and justified as a hardware implementation despite the lack of a realistic hardware simulator. In this paper such work is not considered as Evolvable Hardware. The lack of grounding in real hardware, either physical or through realistic simulators, defies their inclusion in the field of EH. In other cases, authors attempt to speed up their EC process by implementing part or all of the BIA in hardware. In other words, hardware accelerators and certainly not, as defined in this paper, Evolvable Hardware.

In this paper we define the field of Evolvable Hardware (EH) as the design or application of EAs and BIAs for the specific purpose of *creating*[1] physical devices and novel or optimised physical designs.

With this in mind, there are some successes in the field including analogue and digital electronics, antennas, MEMS chips, optical systems as well as quantum circuits. Section 3 presents an overview of some of these successes.

However, let’s step back for a minute and consider what evolving hardware should consist of. Figure 1 illustrates an example of EH where an accurate model of the device physics is applied to produce the fitness function used to evaluate the

[1] “Creating” refers to the creation of a physical entity.

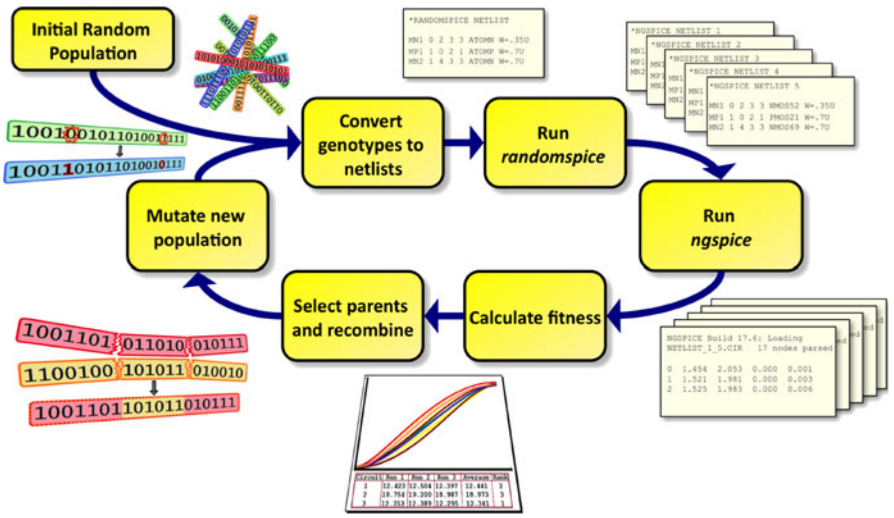
Initial Random Population
Convert genotypes to netlists
Run randomspice
Run ngspice
Calculate fitness
Select parents and recombine
Mutate new population

Fig. 1 An evolvable hardware cycle, for circuit design where high-fidelity simulation is used [13]

efficacy of the circuits. So while the evolutionary loop is using no hardware, the simulation models used are very accurate with respect to the final physical system (in this case transistors with variable characteristics). Other forms of realistic hardware designs may include device simulators, as in [11], or actual physical devices, as in [12].

Such work can be classed under the subfield of Evolvable Hardware Design. The goal is novel or optimised non-adaptive hardware designs, either on physical hardware or as solutions generated from realistic simulators.

The sub-field of Adaptive Hardware can be defined as the application of BIAs to endow real hardware, or simulations thereof, with some *adaptive characteristics*. These adaptive characteristics enable changes within the EH design/device, as required, so as to enable them to continue operating correctly in a changing environment. Such environmental pressure may consist of changes in the operational requirements i.e. functionality change requirements, or maintenance of functionality e.g. robustness, in the presence of faults. Examples of such Adaptive Hardware include an FPAA that can change its function as operational requirements change or a circuit on an FPGA that can “evolve” to heal from radiation damage.

The remainder of the paper is structured as follows: Sect. 2 provides a definition of the field together with the inherent advantages one can expect from such a field. Some actual success stories are reviewed in a little more detail in Sect. 3. Section 4 considers many of the challenges that still face the community. Some newer approaches that are, or might be, applied to evolvable hardware are discussed in Sect. 5. Section 6 provides some thoughts on the future of evolvable hardware and Sect. 7 concludes with a short summary.

2 Defining evolvable hardware

2.1 Evolvable hardware characteristics

In the literature, it is difficult to find a set of characteristics that the community has agreed upon for what characterises an EH system. Indeed most work does not address this issue at all. However, there are a number of fundamental characteristics that should be included in such a definition:

- The system will be evolved not designed (an obvious one, but worth pointing out) N.B.: The term “evolved” is used to encompass any BIA and does not assume that an EA is used
- It is often not an optimal design, but will be fit for the purpose (typical of biological systems in general—due to the evolution process)
- Typically evolved systems do show levels of fault tolerance not seen in designed systems (again typical of biological systems)
- They may be designed to be adaptable to environment change. Such adaptivity is dependent upon when the evolutionary cycle stops. This is not necessarily true for all evolved systems, for example those that stop once the goal has been reached (again this is a characteristic of all biological systems)
- They produce systems that are most often unverifiable formally; indeed in many cases it is difficult to analyze the final system to see how it performs the task.

2.1.1 What is Evolvable Hardware?

From considering the characteristics involved in EH one can start to be more precise as to what is and is not EH. Evolvable hardware will include a form of evolutionary loop, a population (even very small ones), variation operators and selection mechanisms. These will vary from one implementation to another but will be present. In addition to this there are a number of other characteristics that are important to consider:

- The final product of the evolutionary process (the final phenotype) should be hardware, or at least have the potential to be implemented in hardware. An obvious example would be evaluating the evolving design on a given FPGA device and for the evolved design to be implemented on that FPGA. A less obvious example might be the evolution of part or all of a VLSI device where the fabrication of such a device may not be feasible at that point in time i.e. in terms of cost, but could later be fabricated
- The evolutionary process may be applied to the investigation of some future technology medium (non electronics)
- The evolution can be either intrinsic (“on-chip”) or extrinsic (“off-chip”—applying a realistic simulator)
- The evolved design should solve or illustrate a viable approach to solving a “useful” problem—application/device. That is, not one that could be achieved by traditional means e.g. n-bit multipliers

- The final embodied implementation includes features that traditional designs could not produce e.g. inherent fault-tolerance

2.1.2 What is not Evolvable Hardware?

The following are not considered evolvable hardware in this paper:

- Simple optimisation processes (where "simple" refers to nothing more than when the optimization process itself is trivial)
- Where there is never any intention or reason to ever embed the final solution into hardware (whatever the platform)
- Where there are no benefits from evolving a solution
- Where hardware is applied purely as an accelerator

Taking a few examples from the literature and placing these into this context, we can summarise these in Table 1.

2.2 Possible advantages of evolvable hardware

Evolvable Hardware is a method for circuit or device design (within some media) that uses inspiration from biology to create techniques that enable hardware designs

Table 1 Comparison of different proposed evolvable hardware systems

	Analogue/ digital	Intrinsic/ extrinsic	Real hardware	Realistic simulator	Hardware accelerator?	EH?
Higuchi's team e.g. [2, 11, 52]	Digital	Intrinsic	Yes	No	No	Yes
JPL team e.g. [16, 32, 47]	Analogue	Intrinsic	Yes	No	No	Yes
N-bit digital circuit evolution e.g. [39, 48, 71]	Digital	Extrinsic	No	No	No	No
Lohn's team e.g. [34, 83, 91]	Analogue	Extrinsic	Yes	Yes	No	Yes
NanoCMOS project e.g. [13]	Analogue (for digital)	Extrinsic	Potentially	Yes	No	Yes
EvoDevo work e.g. [59, 69, 73]	Digital	Both	Yes	No	No	Yes
Koza's team e.g. [23, 26, 28]	Analogue	Extrinsic	No	Yes	No	Yes
Sekanina's team e.g. (101)	Digital	Both	Yes	No	No	Yes
Neural Network (NN) projects e.g. [5, 6, 61]	Both	Both	Sometimes	Sometimes	Yes	No[a]
Liquid chrystal evolution e.g. [84–86]	Analogue	Intrinsic	Yes	No	No	Yes

[a] If the structure and connections of an NN project are actually evolved and may be developed then it could be considered as EH

to emerge rather than be designed. At first sight this would seem very appealing. Consider two different systems, one from nature and one human engineered: the human immune system (nature design) and the computer that we are writing this paper on (human design). Which is more complex? Which is most reliable? Which is optimised most? Which is most adaptable to change?

The answer to almost all such questions is the human immune system. This is usually the case when one considers most natural systems. Possibly the only winner from the engineered side might be the question, "Which is optimised most?". However, this will depend on how you define optimised. While this is appealing, anyone that has used any type of evolutionary system knows that it is not quite that straightforward. Nature has a number of advantages over current engineered systems, not least of which are time (most biological systems have been around for quite a long time) and resources (most biological systems can produce new resources as they need them e.g. new cells). In addition, we need to be very careful as practitioners of bio-inspired techniques that we pay due regard to the actual biological systems and processes. All too often researchers read a little of the biology and charge in applying this to an engineering system without really understanding the biology or the consequences of the simplifications they make. On the other hand, we are not creating biological systems but rather artificial systems. An exact replication of a complex biological process may neither be needed nor appropriate for the hardware design in question. Improvements in BIAs for EH cannot be justified purely by their "improvements in biological realism", as may be seen in the literature. Such improvements should be reflected in improvements in the efficiency of the BIA or/and the hardware solutions achievable.

It is important to note that biological-like characteristics will be observed in an engineering system that is referred to as bio-inspired but that has in fact little or no real resemblance to the biology that "inspired" it. It is thus important to have a proper framework when using bio-inspired techniques. A framework for such systems was suggested in [14] and is illustrated in Fig. 2. We are, however, not suggesting that every EH researcher needs to turn to biological experimentation. However, the biological inspiration and validation does need to come from real biological knowledge.

To summarise, use evolvable hardware where appropriate, associate evolution with something that has some real connection with the hardware and if using a

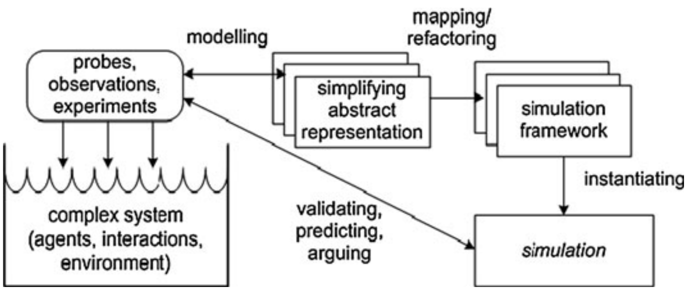
modelling
mapping/ refactoring
probes, observations, experiments
simplifying abstract representation
simulation framework
instantiating
complex system (agents, interactions, environment)
validating, predicting, arguing
simulation

Fig. 2 Conceptual framework [14]

simulator, make sure the simulator is an accurate one for the technology of choice and ensure that your biological inspiration is not only biologically inspired but also beneficial to the evolved hardware.

3 Platforms for intrinsic Evolvable Hardware

Evolvable hardware has, on the whole, been driven by the hardware/technology available at a particular time, and usually this has been electronic hardware. While the paper is not going to go into great detail on the different hardware platforms, it is at least an important enough driver for the subject to mention a few of the main platforms/devices. These are presented under Analogue and Digital Devices.

3.1 Analogue devices

3.1.1 Field programmable analogue arrays (FPAA)

The Lattice Semiconductor ispPAC devices are typical of what is currently available from manufacturers that allow reconfiguration of "standard" analogue blocks (in this case based around OpAmps). The isp family of programmable devices provide three levels of programmability: the functionality of each cell; the performance characteristics for each cell and the interconnect at the device architectural level. Programming, erasing, and reprogramming are achieved quickly and easily through standard serial interfaces. The device is depicted in Fig. 3. The basic active

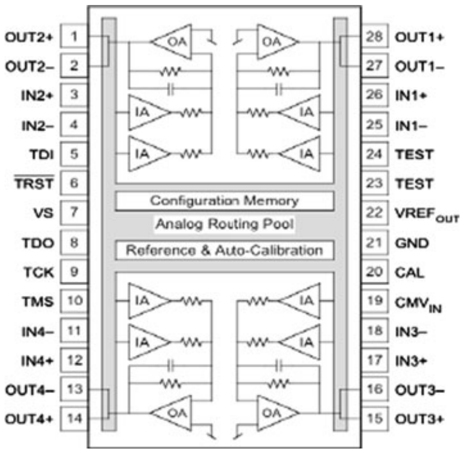

Fig. 3 PACell structure [15]

functional element of the ispPAC devices is the PACell which, depending on the specific device architecture, may be an instrumentation amplifier, a summing amplifier or some other elemental active stage. Analogue function modules, called PACblocks, are constructed from multiple PACells to replace traditional analogue components such as amplifiers and active filters, eliminating the need for most external resistors and capacitors. Requiring no external components, ispPAC devices flexibly implement basic analogue functions such as precision filtering, summing/differencing, gain/attenuation and conversion. An issue for someone wishing to undertake evolution on these devices is the fact that the changes that can be made are at a relatively high functional level i.e. at the OpAmp level. This, together with the overhead for changing functionality, have limited their use in the EH field.

3.1.2 JPL field programmable transistor array

The Field Programmable Transistor array, designed by the group at NASA, was the first such analogue device specifically designed with evolvable hardware in mind. The FPTA has transistor level reconfigurability and supports any arrangement of programming bits without danger of damage to the chip (as is the case with some commercial devices). Three generations of FPTA chips have been built and used in evolutionary experiments. The latest chip, the FPTA-2, consists of an 8 $\times$ 8 array of reconfigurable cells (see Fig. 4). The chip can receive 96 analogue/digital inputs and provide 64 analogue/digital outputs. Each cell is programmed through a 16-bit data bus/9-bit address bus control logic, which provides an addressing mechanism to download the bit-string of each cell. Each cell has a transistor array (reconfigurable circuitry shown in Fig. 4), as well as a set of other programmable resources (including programmable resistors and static capacitors). The reconfigurable circuitry consists of 14 transistors connected through 44 switches and is able to

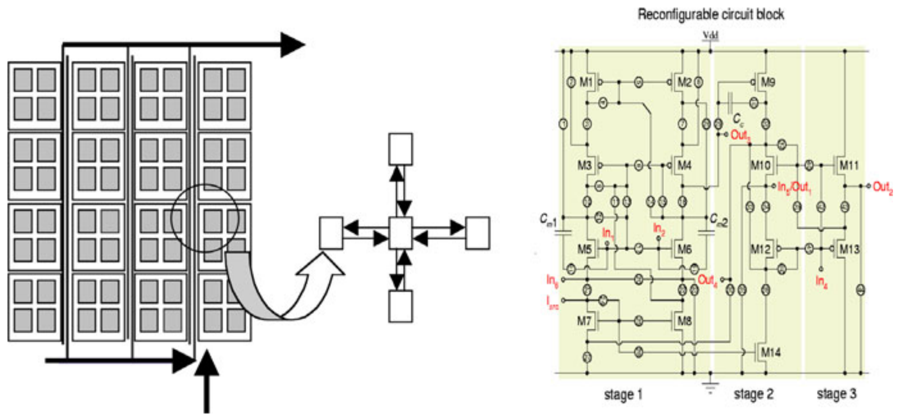

Fig. 4 The FPTA2 architecture. Each cell contains additional capacitors and programmable resistors (not shown) [16] © 2000 IEEE

implement different building blocks for analogue processing, such as two- and three-stage Operational Amplifiers, logarithmic photo detectors and Gaussian computational circuits. It includes three capacitors, Cm1, Cm2 and Cc, of 100fF, 100fF and 5pF, respectively.

3.1.3 Heidelberg field programmable transistor array (FPTA)

The FPTA consists of 16 × 16 programmable transistor cells. As CMOS transistors come in two types, namely N- and P-MOS, half of the transistor cells are designed as programmable NMOS transistors and half as programmable PMOS transistors. P- and N-MOS transistor cells are arranged in a checkerboard pattern, as depicted in Fig. 5. Each cell contains the programmable transistor itself, three decoders that allow the three transistor terminals to be connected to one of the four cell boundaries, Vdd or Gnd and six routing switches. Width W and Length L of the programmable transistor can be chosen to be 1, 2, …, 15 μm and 0.6, 1, 2, 4, or 8 μm, respectively. The three terminals: Drain, Gate and Source, of the programmable transistor can be connected to either of the four cell edges (N, S, E, W), as well as to Vdd or Gnd. The only means of routing signals through the chip is via the six routing switches that connect the four cell borders with each other. Thus, in some cases, it is not possible to use a transistor cell for routing and as a transistor.

3.2 Digital devices

3.2.1 Field programmable gate arrays (FPGA)

The FPGA is a standard digital device and is organized as an array of logic blocks, as shown in Fig. 6. Devices from both Xilinx and Altera have been applied to EH. Programming an FPGA requires programming three tasks: (1) the functions implemented in logic blocks, (2) the signal routing between logic blocks and (3) the characteristics of the input/output blocks i.e. a tri-state output or a latched input. All of these, and hence the complete functionality of the device (including

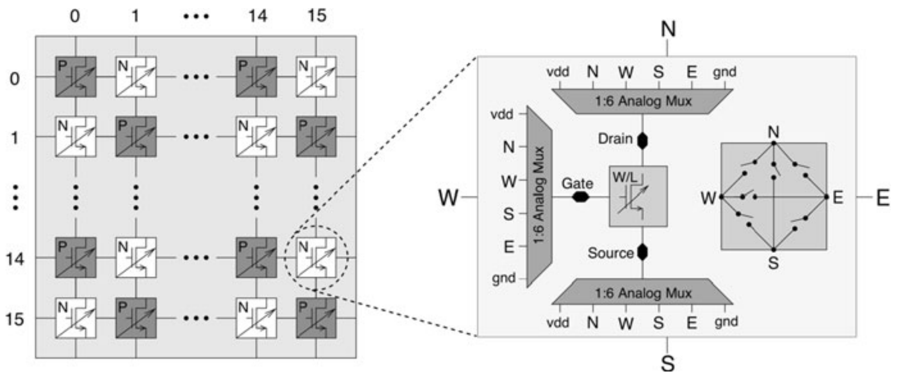

Fig. 5 Schematic diagram of the FPTA [17]

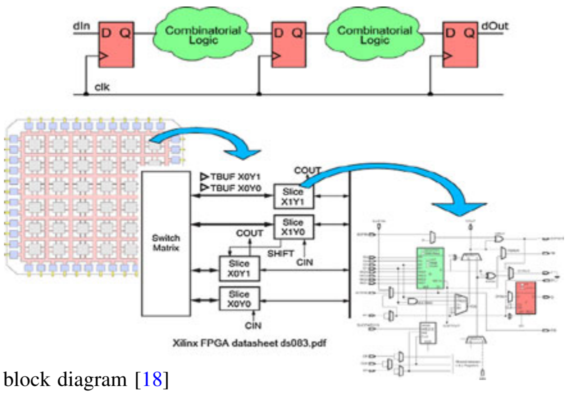

Fig. 6 Generic FPGA block diagram [18]

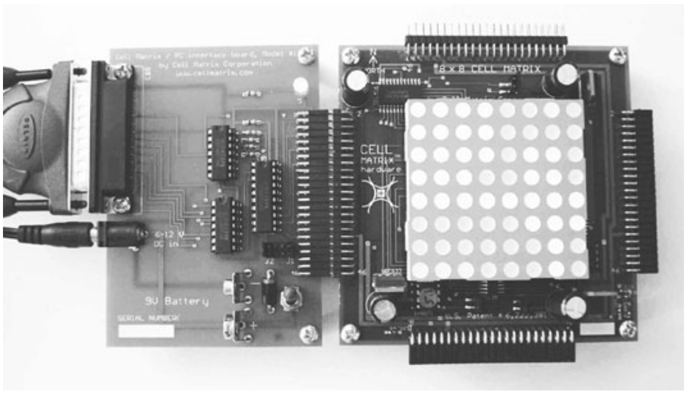

Fig. 7 The Cellmatrix MOD 88, 8 × 8 array [19]

interconnect), are defined by a bit pattern. Specific bits will specify the function of each logic block, other bits will define what is connected to what. To date FPGAs have been the workhorse of much of EH that has been achieved in the digital domain.

3.2.2 The CellMatrix MOD 88

As with analogue devices, the other path to implement evolvable hardware is to design and build your own device, tuned to the needs of Evolvable Hardware. The Cell Matrix, illustrated in Fig. 7, is a fine-grained reconfigurable device with an 8 × 8 array of cells, where larger arrays may be achieved by connecting multiple 8 × 8 arrays. Unlike most other reconfigurable devices, there is no built in configuration mechanism. Instead each cell is configured by its nearest neighbour, providing a mechanism for not only configuration but also self-reconfiguration and

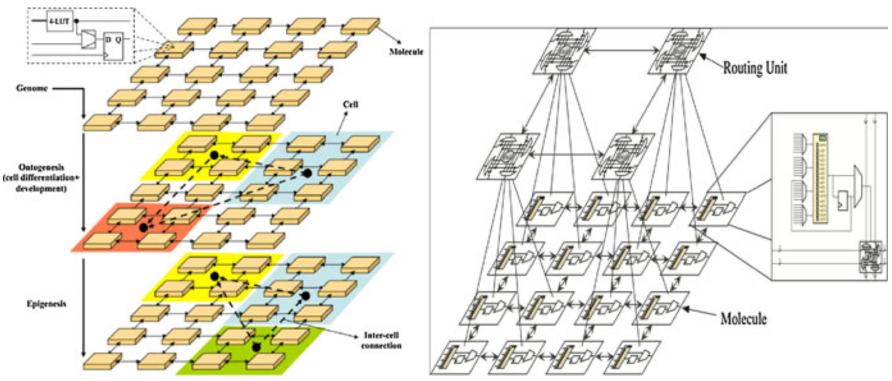

Fig. 8 Organic subsystem, schematic and organic views [21]

the potential for dynamic configuration. Further, the Cell matrix cannot be accidently damaged by incorrect configurations, as is the case for FPGAs if the standard design rules are not followed.

3.2.3 The POEtic device

The goal of the "Reconfigurable POEtic Tissue" ("POEtic") [20], completed under the aegis of the European Community, was the development of a flexible computational substrate inspired by the evolutionary, developmental and learning phases in biological systems. POEtic applications are designed around molecules, which are the smallest functional blocks (similar to FPGA CLBs). Groups of molecules are put together to form larger functional blocks called cells. Cells can range from basic logic gates to complete logic circuits, such as full-adders. Finally, a number of cells can be combined to make an organism, which is the fully functional application. Figure 8 shows a schematic view of the organic subsystem.

The organic subsystem is constituted from a regular array of basic building blocks (molecules) that allow for the implementation of any logic function. This array constitutes the basic substrate of the system. On top of this molecular layer there is an additional layer that implements dynamic routing mechanisms between the cells that are constructed from combining the functionality of a number of molecules. Therefore, the physical structure of the organic subsystem can be considered as a two-layer organization (at least in the abstract), as depicted in Fig. 8.

3.2.4 The RISA device

The structure of the RISA cell enables a number of different system configurations to be implemented. Each cell's FPGA fabric may be combined into a single area and used for traditional FPGA applications. Similarly, the separate microcontrollers can be used in combination for multi-processor array systems, such as systolic arrays [10]. However, the intended operation is to use the cell parts more locally, allowing the microcontroller to control its adjoining FPGA configuration. This cell structure is inspired by that of biological cells. As illustrated in Fig. 9, the microcontroller

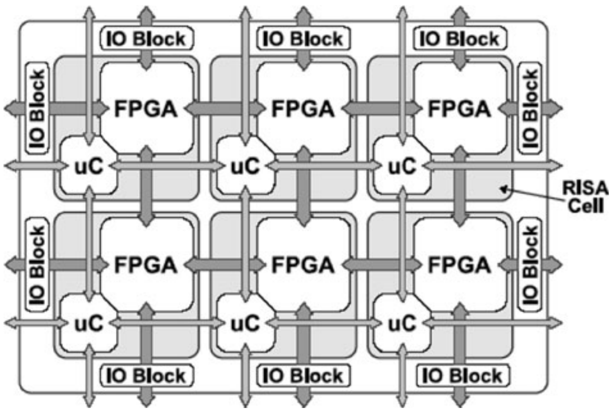

Fig. 9 The RISA architecture comprising an array of RISA Cells [22]

provides functionality similar to a cell nucleus. Each cell contains a microcontroller and a section of FPGA fabric. Input/output (IO) Blocks provide interfaces between FPGA sections at device boundaries. Inter-cell communication is provided by dedicated links between microcontrollers and FPGA fabrics.

4 Success stories

There have been some real successes within the community over the past 15 years. This section gives a short review of some of these.

4.1 Extrinsic analogue

Some of the earliest work on using evolutionary algorithms to produce electronic circuits was performed by Koza and his team [23–27]. The majority of this work focused on using Genetic Programming (GP) to evolve passive, and later active, electronic circuits for "common" functions such as filters and amplifiers. All of this work was extrinsic, that is performed in software with the results generally presented as the final fit individual. Much of the early work considered the design of passive filters with considerable success [25]. Figure 10 illustrates such results. The interesting aspect in this particular paper (that was carried on and developed in subsequent work) was the use of input variables (termed *free variables*) and conditional statements to allow circuit descriptions to be produced that were solutions to multiple instances of a generic problem, in this case filter design. In Fig. 10 this is illustrated by specifying, with such input variables, whether a low-pass or high-pass filter is the required final solution of the GP.

A more sophisticated example of evolving electronic circuits is shown in Fig. 11 [28]. Here the requirement was to create a circuit that mimicked a commercial amplifier circuit (and even improve on it). There are a number of both subtle and

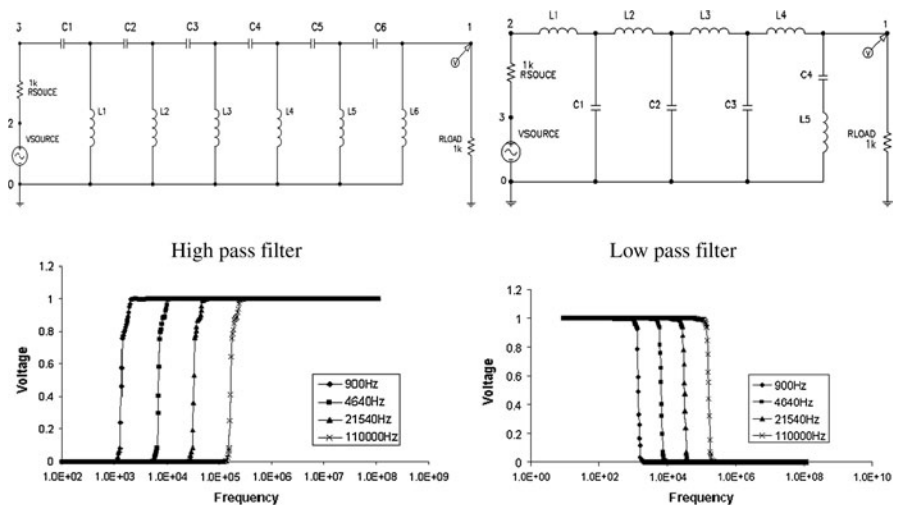

Fig. 10 Examples of filter circuits and frequency plots obtained by evolutionary processes [25] @2000 IEEE

complex points that come out of this paper that are of interest and use to others trying to use evolvable hardware. The paper illustrates how the use of domain knowledge, both general and problem-specific, is important as the complexity of the problem increases. New techniques, or improvements in actual GP techniques were developed to solve this problem. Finally, and potentially most important, the use of multi-objective fitness measures are shown to be critical for the success of evolution.

In this case 16 objective measures were incorporated into the evolutionary process, including: gain, supply current, offset voltage, phase margin and bias current. When evolving for real systems in real environments the use of multi-objective fitness criteria is shown to be extremely important.

Koza and his group continue to evolve electronic circuits in an extrinsic manner and have many results where the evolutionary process has replicated results of human designs that have been patented in the past.

4.2 Intrinsic digital

One of the very first research teams to apply evolvable hardware in an intrinsic manner was Higuchi and his group in Japan [1, 2, 29–31]. This work included applying evolvable hardware (almost always intrinsically) to myoelectric hands, image compression, analogue IF filters, femto-second lasers, high-speed FPGA I/O and clock-timing adjustment. Here we will give a little more detail on the application of clock-timing adjustment as an illustration of the work conducted by Higuchi and his group.

As silicon technology passed the 90 nm process size new problems in the fabrication of VLSI devices started to appear. One of these is the fact that, due to process variations cause by the fabrication process, the clock signals that appear

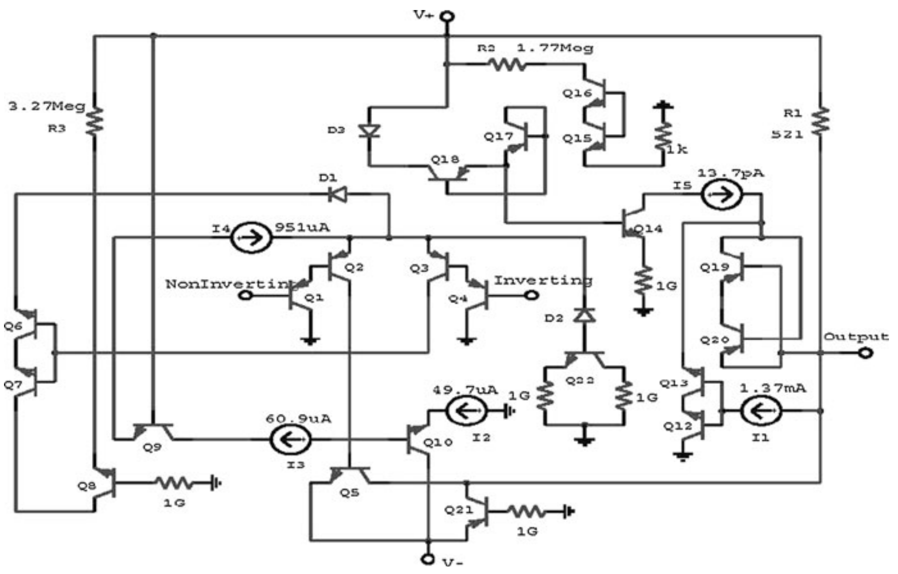

Fig. 11 Best run of an amplifier design [28]

around the chip are not always synchronised with each other. This issue is often called clock skew and can be a major issue in large complex systems. This is a very difficult issue for chip designers since it is a post fabrication problem where there is a significant stochastic nature to the issues. One solution would of course be to slow the clocks down, but that is generally an unacceptable solution. Another possibility is to treat this as a post-fabrication problem, where actual differences in paths and actual speeds can be measured.

Figure 12 illustrates the basic philosophy behind the idea of using evolution to solve this problem. Typically clock signals are transmitted around complex VLSI devices, in a hierarchical form, often known as clock-trees, as illustrated in the left-hand picture in Fig. 12. At the block level (middle picture), sequential circuits should receive clock signals at the same time for the circuit to perform correctly. The idea behind this work was to introduce programmable delay circuits (right-hand picture) that were able to fine-tune the delay on all the critical path lengths that the clock propagated. The issue is, what delay should be programmed into each individual path on each individual VLSI device? Each device is likely to be different due to the fabrication process and we will not know what this is until after fabrication. However, the delay can be controlled (programmed) by a short bit pattern stored in a register, on-chip. This bit pattern can form part of a genome that is used within an evolutionary loop that is using a Genetic Algorithm (GA) to optimise the delay in each of the critical paths in the circuit. Figure 13 illustrates in more detail the actual circuit elements required, and the final chip design, to achieve the required functionality for this evolvable chip. The results suggest that not only can the clock skew be optimised, and consequently the frequency that the device can run at be increased (in the paper by 25%) but that also the supply voltage can be reduced while maintaining a functioning chip (in some cases by more than 50%).

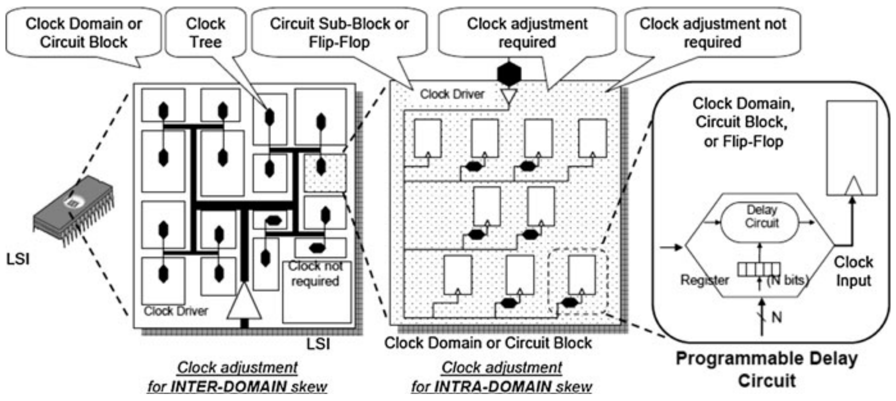
Clock Domain or Circuit Block
Clock Tree
Circuit Sub-Block or Flip-Flop
Clock adjustment required
Clock adjustment not required
LSI
Clock Driver
Clock not required
Clock Domain or Circuit Block
Clock Domain, Circuit Block, or Flip-Flop
Delay Circuit
Register
(N bits)
N
Clock Input
Programmable Delay Circuit
Clock adjustment for INTER-DOMAIN skew
Clock adjustment for INTRA-DOMAIN skew

Fig. 12 Basic process of clock skew adjustment using evolvable hardware [29]

4.3 Intrinsic analogue

Section 3 has already given a brief outline of the work performed by the team at NASA JPL on the design and manufacture of a number of analogue programmable devices aimed at assisting with intrinsic analogue evolution, Field Programmable Transistor Arrays (FPTAs) [16]. Here, one of these devices is presented to illustrate, not only the evolution of useful circuits (in real environments), but that through the continued use of evolution throughout the lifetime of the system, fault tolerance is increased. In this case the functionality considered is that of a 4-bit digital-to-analogue converter (4-bit DAC). The basic evolvable block used in these

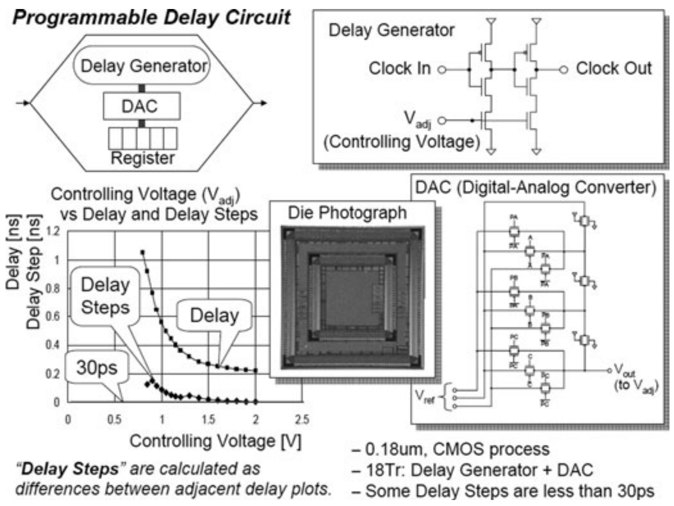
Programmable Delay Circuit
Delay Generator
DAC
Register
Delay Generator
Clock In
Clock Out
V_adj (Controlling Voltage)
DAC (Digital-Analog Converter)
Controlling Voltage (V_adj) vs Delay and Delay Steps
Delay [ns]
Delay Step [ns]
Delay Steps
Delay
30ps
Controlling Voltage [V]
Die Photograph
V_ref
V_out (to V_adj)
"Delay Steps" are calculated as differences between adjacent delay plots.
– 0.18um, CMOS process
– 18Tr: Delay Generator + DAC
– Some Delay Steps are less than 30ps

Fig. 13 Realisation of clock skew adjustment on VLSI device [29]

experiments refers back to Fig. 4 in this paper and the results obtained in the work can be seen, summarised, in Fig. 14.

Hardware experiments use the FPTA-2 chip, with the structure illustrated in Fig. 4. The process started by evolving first a 2-bit DAC—a relatively simple circuit. Using this circuit as a building block, a 3-bit DAC was then evolved and again reusing this a 4-bit DAC was evolved. The total number of FPTA cells used was 20. Four cells map a previously evolved 3-bit DAC (evolved from a 2-bit DAC), four cells map a human designed Operational Amplifier (buffering and amplification) and 12 cells have their switches' states controlled by evolution. When the fault was to be injected into the operating circuit, the system opened all the switches of the 2 cells of the evolved circuit.

Figure 14 top-left: The faulty cells are shown in black. In these cells all switches were opened (stuck-at-0 fault). The 3-bit DAC, cells '0', '1', '2' and '3' map the previously evolved 3-bit DAC, whose output is O3. The Operational Amplifier cell (Label 'A') was constrained to OpAmp functionality only. The evolved cell (in grey) switches' states were controlled by evolution. O4 is the final output of the 4-bit DAC.

Figure 14, plots: The top-right plot illustrates the initial evolved 4-bit DAC with inputs 1, 2 and 3 (4 is missing due to number of oscilloscope channels) and output O4 which can be seen to be functioning correctly. The bottom left plot in Fig. 14 shows the response of the 4-bit DAC when two cells are made faulty. Finally, the plot at the bottom-right of Fig. 14 illustrates the response of the DAC after further evolution (after fault injection) has taken place. This response was achieved after only 30 generations of evolution. Again the only cells involved in the evolutionary

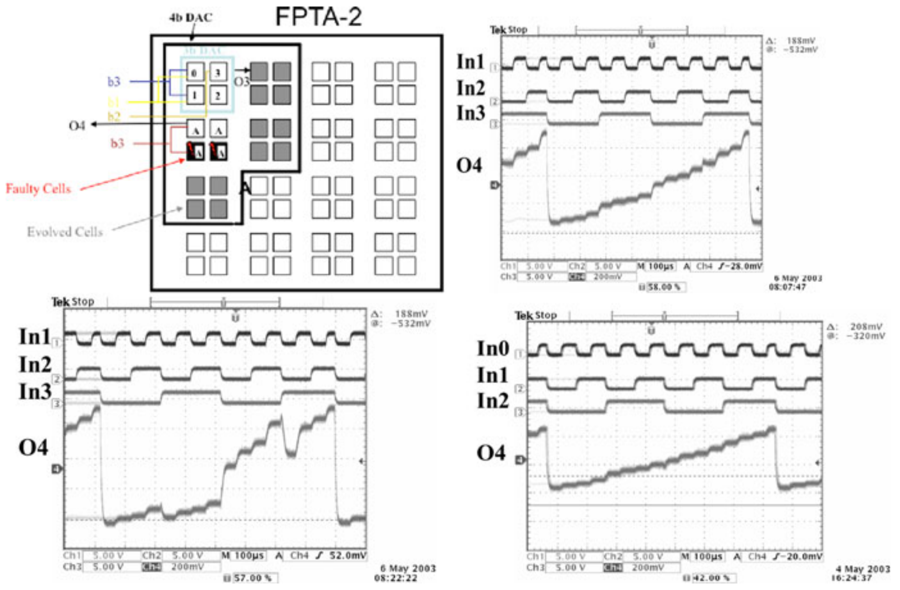

Fig. 14 FPTA topology (*top-left*), evolved 4-bit DAC response (*top-right*), evolved 4-bit DAC response with two faulty cells (*bottom-left*) and recovered evolved 4-bit DAC response with two faulty cells (*bottom-right*) [32] © 2004 IEEE

process are those in grey in Fig. 14. This example shows very well that recovery by evolution, in this case for a 4bit DAC, is possible. The system made available 12 cells for evolution. Two cells were constrained for the implementation of the OpAmp. Four cells were constrained to implement the simpler 3bit DAC. Two faulty cells were implemented by placing all their switches to the open position.

4.4 Extrinsic and intrinsic analogue

As a final example in this section, the domain of evolved antenna design [33] is presented. In the work presenter in [34], a GA was used in conjunction with the Numerical Electromagnetic Code, Version 4 (NEC, standard code within this area) as the simulator to create and optimize wire antenna designs. These designs not only produced impressive characteristics, in many cases similar or better to those produced by traditional methods, but their structure was often very different from traditional designs (see Fig. 15). The 'crooked-wire' antennas consisted of a series of wires joined at various locations and with various lengths (both these parameters were determined by the GA). Such unusual shapes, unlikely to be designed using conventional design techniques, demonstrate excellent performance both in simulation and physical implementation [34].

Apart from the difference in domain, an interesting feature of this application of EH is the relatively large number of constraints (requirements) that such systems impose on a design. Requirements of a typical antenna include [34]: Transmit frequency; receive frequency; antenna RF input; VSWR; antenna gain pattern; antenna input impedance, and grounding. From this list, which is not exhaustive, it can be seen that this is certainly a non-trivial problem requiring a complex objective function to achieve appropriate results.

Given the nature of the problem and the solution (wire antenna), a novel representation and construction method was chosen. The genetic representation was a small "programming language" with three instructions: place wire; place support

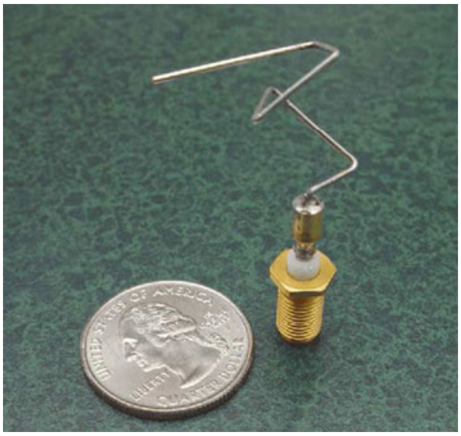
Fig. 15 Prototype evolved antenna, which on March 22, 2006 was successfully launched into space [34] © 2003 IEEE

and branch. The genotype specifies the design of one arm in a 3D-space. The genotype was a tree-structured program that builds a wire form. Commands to undertake this process were: Forward (length radius); rotate_x (angle); rotate_y (angle) and rotate_z (angle). Branching in genotype equates to branching in wire structure. The results were very successful (see Fig. 15) and the characteristics (electrical and mechanical) of the final antenna more than met the requirements list.

The previous example was situated within the extrinsic world, that is, evolution ran within a simulation environment and only when a successful individual was found, was a physical entity built. The authors then went on to consider how you might do evolution intrinsically. The goal now was to explore the in situ optimisation of a reconfigurable antenna. One motivation behind this work, was to consider how a system might adapt to changes in spacecraft configuration and orientation as well as to cope with damage. A reconfigurable antenna system (see Fig. 16) was constructed. The software that controlled the reconfiguration, evolved relay configurations. In the initial stages of this work, the user subjectively ranked designs based on signal quality but the plan was to automate this process in future work. In the system shown in Fig. 16-left, the system consisted of 30-relays that were used to create the antenna designs. It was shown that the system was able to optimise effectively for frequencies in the upper portion of the VHF broadcast TV band (177–213 MHz). A simple binary chromosome was selected, 1 s and 0 s, corresponded to closed and open switches respectively. The connections between the pads and the switches defined the chromosome mapping (Fig. 16-right). The antenna was able to automatically adapt to different barriers, orientations, frequencies, and loss of control and in a number of cases showed superior performance over conventional antennas.

5 Challenges

The previous section considered a range of successes for evolvable hardware. While these are not the only successes, they do represent a fair range of achievements: from analogue to digital electronics and from extrinsic to intrinsic evolution.

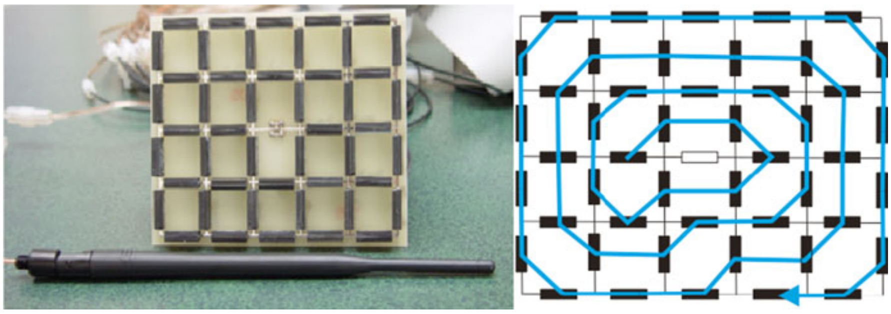

Fig. 16 Reconfigurable antenna platform (*left*) and mapping of chromosome to structure (*right*) [33] © 2002 IEEE

However, these successes have been few relative to the efforts put into the field so far. So where, if anywhere, does evolvable hardware go from here? The rest of this paper gives some speculative suggestions to the answer to this question.

5.1 From birth to maturity

There are many fundamental challenges in the field but one major challenge, which is perhaps rarely mentioned, is the challenges of any field: the need to publish. In the early stages of a new field, there is much uncertainty but at the same time there are many ways to achieve simple exciting results, raising the field's prominence quickly. Although publishing is a challenge in a new area, there is also the excitement of a new venture. As the newness wears off, without realistic results matching the more traditional field, you need to rely on specialised workshops or conferences. Such specialised workshops and conferences do provide a very valuable venue for development of the field. Specialised researchers are able to recognise more subtle, but important, contributions and their value to the field. Unfortunately this can often lead to over acceptance of small advancements whilst the main goal is forgotten. That is, EH needs to one day compete with traditional electronic design techniques, either in general or for specific sub-areas of electronics or other mediums. As the field becomes more mature, special sessions/tracks become easier to establish. However, if the results are not there, such sessions begin to disappear and the numbers at conferences begin to wane.

Such a process is quite typical of a new field and EH is no exception. The main challenges have either not been solved or perhaps even addressed so as to enable the field to take the bigger steps forward.

Neural networks provide us with a sub-field of AI that almost died out in the late 1960s—in fact for a decade. Here it was the work of Minsky and Papert [35] that highlighted a major non-solvable challenge to the field. Although in the early 1970s, the work of Stephen Grossberg [36] showed that multi-layered networks could address the challenge and the work of Bryson and Ho [37] solved the challenge through the introduction of backpropogation, it wasn't until the mid 1980s, and the work of Rumelhart et al. [38], before backpropogation gained acceptance and the field was revived. However, in the process a significant decline in interest and funding resulted from the publishing of the Minsky and Papert's 1969 book. A further relevant example is that of the field of AI itself. The field paid a high price due to the overhype in the mid 1980s to mid 1990s.

We do not consider that EH is dying out, but the field is in a critical stage. There is certainly no argument to suggest that the field cannot go forward. However, it is also not clear that there is a path forward which will allow the field to progress in the way that neural networks have progressed. Also, is the field suffering from overhype, similar to that of AI?

Section 3 highlights a number of success stories. However, as already stated, this far from represents the vast research that has gone on in the field. In the early stages of a new field, results are simply that—results, enabling the field to take some small steps forward. In theory small steps should enable researchers to build on these results and take bigger and bigger steps. However, on the whole, this has not been the case with EH. Bigger steps have proven to be very hard to achieve and fewer researchers have been willing to push the boundaries and use the time to achieve such steps. The reality is that small steps give results and publications, bigger steps are much more uncertain and research is becoming more and more publishing driven.

In our view this focus on smaller steps has not only hampered progress in the field but also caused industry and traditional electronics to turn away from EH. In general, one can say that industry is fed up seeing results from “toy problems”. They want results that will make a difference to them. Further, funding challenges in many countries mean that research fields have to rely more and more on industrial funding, providing more pressure to take the bigger steps towards EH becoming a viable alternative to traditional electronics so as to secure funding for further development of the field.

5.2 Scalability

One of the goals of the early pioneers of the field was to evolve complex circuits. That is, to push the complexity limits of traditional design and, as such, find ways to exploit the vast computational resources available on today’s computation mediums. However, the scalability challenge for Evolvable Hardware continues to be out of reach.

Scalability is a challenge that many researchers acknowledge but it is also a term widely and wrongly used in the rush to be the one to have solved it. Evolving an (n + 1)-bit adder instead of a previously evolved n-bit adder is not solving the scalability challenge, especially when one considers the fact that traditional electronics can already design such circuits. Scaling up structures alone, i.e. the number of gates, without a corresponding scaling of functionality, is also not solving the scalability challenge.

The scalability challenge for Evolvable Hardware may be defined as finding a refinement of bio-inspired techniques that may be applied in a way so as to achieve complex functions without a corresponding scaling up of resource requirements (number of physical components, design time and/or computational resources required).

Examples of one-off complex designs may be seen, such as Koza’s analogue circuits [28]. However, here functionality has been scaled up at the expense of huge computational resources—clusters of computers (a brute force solution). Although such work provides a proof of concept that complex designs are possible, much refinement is needed to the techniques to provide a truly scalable technique for designs of such complexity.

So if incrementally increasing complexity i.e. from an n-bit adder to an (n + 1)-bit adder, is not the way forward, how should we approach solving this challenge that has evaded EH researchers for 15 years?

The goal of much EH is to evolve digital and analogue designs (at least if we stick to electronic engineering applications). Little is known about the phenotypic search space of digital and analogue designs. Since any search algorithm provides implicit assumptions about the structures of the search space, our lack of knowledge of the phenotypic space suggests these assumptions and how they match the problem in hand i.e. electronic circuits, and whether the problem is effectively represented gives us little chance of efficiently solving the problem.

Evolvable Hardware has suffered from the lack of theoretical work on the evolutionary techniques applied to this particular problem so as to provide stronger guidelines as to how the field might approach the achievement of a scalable technique. Hartmann et al. [39], studied the Kolmogorov complexity (complexity of a bit string) of evolved fault tolerant circuits through the application of Lempel–ziv complexity [40] to these circuits. The complexity analysis involved analysing the compressibility of strings (representing circuits), and comparison of such analysis to the complexity spectrum (simple to highly complex strings). The complexity of the circuits analysed was found to be in a significantly smaller region of complexities than that covered by the evolutionary search, indicating that it would be beneficial to bias the search towards such areas of the complexity search space.

It should be borne in mind that Kolmogorov complexity may or may not be the most suitable theoretical measure for the analysis of EH complexity and this is just one example of a possible approach. However, this work is presented to highlight how theoretical approaches to EH can help us to understand our application area and help us refine our techniques to such applications.

5.3 Measurements

Measurement refers to the metric used to evaluate the viability of a given solution for the problem in hand. In traditional electronics, for example, terms such as functional correctness (which may involve many sub-functions) as well as area and power costs are general metrics applied. Other metrics applied may relate to the issue being resolved e.g. reliability.

Although power issues are very important in traditional electronics, such metrics are only starting to appear in EH work [41]. However, the application of area metrics, especially gate counts is often applied, but may be questioned. There are two key issues here: The majority of EH solutions are intended for reconfigurable technology and this focus on evolving small circuits is in contrast to the relatively large resources available. Further, although area count may show the efficiency of EH compared to a traditional technique, for a given generalized solution, one can easily say that a good traditional designer can often match the evolved solution given such an optimisation task. It should, therefore, be borne in mind that the power of evolution as an optimization technique is when the search space is very large, when the number of parameters is large and when the objective landscape is non-uniform. Thus for such small evolved circuits, gate area—although possibly interesting, is not a metric which should be used to illustrate the viability of the solution.

A further challenge with metrics, highlighted in the above description, is the ability to compare solutions between traditional and evolved designs. A traditional designer needs to know whether a given evolved solution is comparable to the traditional designs in terms of the particular evaluation criteria the designer is interested in. A criticism of EH may certainly thus be directed to the use of “number of generations" as a metric which has no relevance to any performance metric in traditional design.

A common design feature addressed in EH and also part of the grand challenges described by the ITRS roadmap 2009 [42] is fault tolerance i.e. reliability. Traditionally the reliability metric measures the probability that a circuit is 100% functional and thus says nothing about how dis-functional the circuit is when it is not 100% functional. To apply such a metric as a fitness criteria to EH would not provide sufficiently fine grained circuit feedback so as to separate potential solutions and, therefore, limits evolution's search to a rougher search. More commonly in EH, reliability is expressed in terms of how correct a solution is, on average, across a spectrum of faults where the faults may be explicitly defined [43] or randomly generated [39]. Such a metric provides evolution with more information about practical solutions and thus supports evolution of solutions. However, as shown in [44], since the traditional metric is designed with traditional design techniques in mind and the EH metric is designed with evolution in mind, the metrics favour the design technique for which the metric was designed. As such, a comparison is difficult. It is thus important to find metrics that cross the divide between traditional and evolved designs.

5.4 Realistic environments or models

One area of interest for EH is device production challenges. Defects arising during production lead to low yield i.e. fewer "correct devices" and thus more faulty devices—devices that pass the testing process but where undetected defects lead to faults during the device's lifetime. There are many ways that evolution may be applied in an attempt to correct or tolerate such defects. However, the challenge remains: a need for real defect data or realistic models. Fault models are difficult to define due to the strong dependence between the design itself and the actual production house due to the complex physical effects that lead to defects.

It should be noted, however, that this challenge is being faced by the traditional design community too. Closer collaboration with industry, illustrating the power of EH is needed so as to secure realistic data and thus provide the basis for further research interest in this area and more realistic results.

In addition, application areas may be found where EH can solve issues with limited data where traditional techniques struggle. In [45] a future pore network CA machine was discussed. The machine would apply an EA to the sparse and noisy data from rock samples, evolving a CA to produce specified flow dynamic responses.

Rather than relying on models, EH can be applied to the creation of robust designs where realistic faults may be applied to the actual hardware and evolution enabled to create robust designs. Zebulum et al. [46] addressed the issue of environmental models. Although such models have become more refined, design and environmental causes are significant sources of failure in today's electronics. The work addressed the space environment and focused on improvements in design and recovery from faults, without the need for detection, through the application of

EH. In this case the underlying hardware was the FPTA, described in Sect. 3, and faults were injected through opening switches to simulate the stuck-at fault model.

A further example, may be seen in Thomson's work [12] on evolving robust designs in the presence of faulty FPGAs where the extreme conditions were physically imposed on the actual chips. The goal was to create robust designs that tolerate elements in the "operational envelope"—variations in temperature, power-supply voltages, fabrication variances and load conditions; by exposing the circuits to elements of such an operational window during fitness evaluations. Similarly the work of [47] involved placing an FPGA and a Reconfigurable Analogue Array (RAA) chip in an extreme temperature chamber to investigate fault recovery in a realistic extreme temperature scenario.

6 Newer approaches

For the last 10 years a number of researchers have been addressing the scalability problem, focusing on finding ways to simplify the task for evolution in terms of finding more effective ways to explore the search space for more complex applications.

6.1 Divide and conquer

The first set of approaches build on the traditional notion of divide and conquer, providing evolution with simpler sub-problems to solve at a time i.e. reducing the search space for evolution. There are two driving forces back such approaches in terms of resource usage: (a) reduction in the number of evaluations needed to find a solution and (b) reduction of evaluation time to evaluate a single individual. The latter point has strong consequences for intrinsic evolvable hardware as the evaluation time to fully test a design grows exponentially with the number of inputs tested.

Various alternatives may be found in the literature including Function-level Evolution [30, 48], Increased Complexity Evolution [49, 50] and Bidirectional Evolution [51].

Function-level evolution [30, 48, 52], does not require a specific division of the problem domain. Instead, the focus is on the selection of larger basic units i.e. adders or multipliers, enabling a scaling up of potential evolved solutions. A study of fitness landscapes by Vassilev and Miller [53] illustrated that using sub-circuits as building blocks, rather than simple gates, speed up of the evolutionary process can be achieved but at the cost of an increased number of gates. Further, the selection of appropriate components is domain specific and limiting evolution to larger components does limit the search space available to evolution.

In Increased Complexity Evolution (ICE) [49, 50] (also termed Divide and Conquer and Incremental Evolution), each basic unit may be a simple gate or higher-level functions. These blocks from the units for further evolution, incrementally increasing the complexity of the task at each evolutionary run i.e. the block evolved in one run can then be applied as a basic block in a consequent run. As such, it is a bottom-up approach to increasing complexity and can be said to build on the work of Gomex and Miikulainen [54] on incremental evolution and the biological fact that humans perform more and more difficult tasks on route to the adult state [55]. However, it is quite a challenge to find ways in which an EH design may be split into simple tasks that can be incrementally increased and thus also to find suitable fitness measures for each stage. A possible solution is to divide the application based on the number of outputs [56]. However, such a solution is only advantageous in the case of multiple outputs.

Bidirectional Evolution (BE) [51] is similar to increased complexity evolution except that incremental evolution is both applied to decompose the problem into the sub tasks and then incrementally make the tasks more challenging. That is, an automatic identification of sub-tasks is provided for through the application of output and Shannon decomposition.

In both ICE and BE, dividing the application by the number of outputs does not reduce the number of test vectors required to fully test the circuits since the number of inputs in unchanged. As such, although a reduction in the number of evaluations may be obtained, the challenge of fitness evaluation remains. Generalised Disjunction Decomposition [57, 58] is a method that can be applied to improve BE by decomposing the problem based on inputs rather than outputs.

6.2 Development and modularisation

A second set of approaches, are inspired by nature's way of handling complexity, i.e. that of artificial development. Here a genome similar to DNA, far from a blueprint of the organism to be developed, is rather a somewhat dynamic recipe for the organism that can be influenced by external factors in the growth and specialization of the cells of the organism. Evolution operates on the DNA-like circuit description of the genome and in the evaluation phase, each individual DNA is developed into a fully-grown phenotype (a circuit in electronics). The introduction of development to an EA provides a bias to modular iterative structures that already exist in real-world circuit designs whilst allowing evolution to search innovative areas of the search space [59]. Further, introducing development to an evolutionary process may be said to enhance exploration, rather than optimisation of solutions [60].

Although the idea of artificial development already existed in the literature [61–66] when the field of evolvable hardware picked up this attractive notion for improving scalability, algorithms for development were few and far between and far from thoroughly researched. A number of EH research groups are involved in developmental approaches to EH [20, 59, 67–73], including theoretical and practical investigations into the feasibility of development as an EH technique [74–77]. However, there is much research to be conducted before such a technique can be said to really be moving along the path towards an efficient technique for EH.

The concept of iterative structures need not just be obtained through developmental approaches but similar to Koza and Rice's Automatically Defined Programs (ADFs) [23], EAs for EH may be designed in a way so as to protect partial solutions that may contribute to the final solution. In this way, instead of defining the functions, as in function-level evolution, such functions are generated by evolution and protected and encapsulated as a function for re-use. The importance of such reuse in EH was discussed in [70]. Embedded Cartesian Genetic Programming (ECGP) [78, 79] is an EA that may be applied to EH enabling modularization.

6.3 Fitness evaluation improvements

The third set of approaches address the evaluation of individuals, a key concern in intrinsic evaluation. From the earliest days of EH, filtering of signals has been shown to be a successful application area. Here, only a single fitness test is required for each individual where the frequency characteristics of the evolved system are compared to the test signals. However, in many applications involving multiple input circuits, the number of evaluations per individual is exponentially related to the number of inputs.

One approach to this issue is to focus on ways to reduce the number of test vectors required whilst being able to provide a reliable evaluation method that allows for prioritising of individuals. Such an approach is not new in that it is in approach commonly applied in neural networks and classification systems. That is the set of input vectors are divided into a set of training vectors and a set of test vectors. Creating EH through the application of Neural Networks may use a similar process, verifying the evolved circuits by applying the test vectors.

In [80] a random sample of input vector tests for digital truth tables was applied to assess the evolvability of functionality with such limited evaluations. Unfortunately the results showed that with the setup involved, little merit was found in such an approach and it was clear that the EA needed the complete truth table so as to provide sufficiently detailed information so as to evolve a 100% fit individual.

Instead of considering an incomplete search, another possibility is to identify applications that require a single evaluation similar to that of signal filtering. In [81] it was shown that if the circuit sought could be interpreted as a linear transformation i.e. that the basic elements were linear operators (add, shift, subtract, etc.), then only a single test vector was needed. The work investigated a multiple constant multiplier. Although not a complex application in its own right, it is a useful component in more complex circuits. Further investigation is needed to identify more complex applications for such a technique.

6.4 Newer materials

As discussed in Sect. 2, various digital and electronic platforms have provided the basis for EH research. Specialized devices have on the one hand provided devices more tuned to the needs of BIAs, whilst on the other hand provided an experimental platform and results with very limited access and applicability to the broader community, except in the case of Cell Matrix where both the device and supporting software were made available.

In general, non-specialized devices have provided a more common platform, enabling more exchange of knowledge between different research groups and different experiments. However, today’s technology is far from suitable for BIAs. The early work of Thomson [82] already showed that evolution was capable of not only exploiting the underlying physics of FPGAs but further showed that evolution created a design solution for a task that the chip was not designed to handle—that of a tone discriminating circuit [83]. That is, evolution explored areas of the design space beyond that which the device was designed for. However, as may be seen from the success stories highlighted in Sect. 4, where all the experiments were based on non-standard devices, there are many challenges to be faced in achieving EH on standard components.

The seminal work of [82] did, however, provide a proof of concept that BIAs may be applied to search areas of the design space beyond that which today's electronic technology is designed to meet. Similarly, the work of Harding and Miller applied evolution to liquid crystal, exploiting the underlying physics [84] so as to achieve various functions (a tone discriminator [85] and a robot controller [86]). Here, evolution was applied to an unconventional computation medium—treated as a black box, causing functionality to emerge from within the liquid crystal. That is, evolution exploited the structure and inherent dynamics of the liquid crystal so as to achieve functionality.

Under the realms of the MOBIUS project, Mahdavi and Bentley [87] investigated Smart materials for robotic hardware. Nitonal wires (alloy made up of nickel and titanium) were used as the muscle hardware of a robotic snake, applying an EA to evolve the activations for the wires controlling the snake-like movement. Further, inherent adaptation to failure was seen after one of the wires broke during experimentation. Nitonal is a shape memory alloy which has two forms at different temperatures such that changing from one temperature to the other causes a phase transformation and the alloys are super elastic and have shape memory. Such an adaptive solution (EA + shape memory alloy) may be useful not only for snake-like robots but for devices in varying environments that may need to change their morphology accordingly.

A further medium suggested by Olteam [88] is switchable glass (smart windows), controlling the transmission of light through windows by the application of variable voltages. Included under such a classification is liquid crystal. Other possibilities include electrochromic devices and suspended particle devices—rods (particles) suspended in a fluid. The former relies on chemical reactions where the translucency (visibility) of the glass may be adjusted through the application of various voltages. The latter relies on field effects where the rods align to a given electric field and varying the voltage can control the amount of light transmitted.

7 The future

Although the newer approaches described in Sect. 6 are still facing many challenges, these illustrate that benefits can be gained by looking again at biology and even combining both traditional and biologically-inspired techniques. Nature does not have all the answers for us since we are in fact designing an artificial organism i.e. a hardware device/circuit.

In Sect. 4, the scalability challenge was discussed. It is important that researchers continue to reach out to both natural and artificial inspirations for solutions to our challenges. However, for real progress it is important that researchers address the fundamental issues with any new technique and keep the scalability issue in mind. A new technique may be interesting to work with but the real issues to be solved and how these should be approached so as to scale up the technique are critical for the future of the field.

7.1 Applications

Looking to potential applications has always been a fundamental issue for the EH community. One possible approach is to turn the attention away from complex applications to applications which may not necessarily be inherently complex but in some way provide challenges to more traditional techniques. There are many opportunities within design challenges that should be given further attention. Fault tolerance [89], fault detection and repair [90–92], although already a focus in EH provide many opportunities for further discoveries. Moving from faults to testing—a significant challenge facing the electronic industry, provides many opportunities such as in test generation [93] and testing methodology benchmark circuits [94, 95]. Surprisingly, although a major focus within the electronic industry, power usage [96] is only recently gaining attention in the EH field. A further increasing challenge in industry is yield due to the challenge of production defects [78, 97].

Within such design challenges, certain application areas provide further challenges. For example, in space, manned and unmanned vehicles will require to not only tolerate single faults—a traditional fault tolerance scenario; but also multiple faults [98]. Space provides hard application areas for any technique whether traditional or non-traditional, especially with respect to robustness and adaptivity requirements. Here the EH approach is not being applied to "reinvent the wheel" but rather applied in the search for novel or more optimal solutions where traditional techniques have either failed or are still challenged. Of course the inherent real-time constraints are inherently hard for such stochastic search algorithms.

One area that is still relatively untouched, but where EH has a real future is in adaptive systems. Such adaptivity may be seen in the light of failures i.e. adaption to faults or adaptive functionality. The goal being to adapt in real time to environmental changes requiring changes in functionality and/or stability of functionality. Two key application areas already under focus are both digital and analogue challenges in space applications—where access to reconfiguration is limited, and also in autonomous robots—reacting in real time to environmental influences. There are many challenges facing the field within true adaptive design i.e. evolving a new design and replacing the old design in real time.

Instead of focusing on the challenge of achieving adaption "on-the-fly", another option is to adapt to pre-defined solutions. A newer approach to adaptive solutions which may be said to capture some of the features of EH together with more traditional approaches to adaptive design is metamorphic solutions proposed by Greenwood and Tyrrell [99]. Such an approach borrows the key elements from evolvable design with the exception that many of the adaptive solutions are pre-designed. Similar to more traditional adaptive design, real-time adaptivity is sought through the process of selection from a pool of pre-designed solutions. However, metamorphic systems extends the concept of pre-defined solutions by allowing for components implemented in various technologies.

7.2 Platforms

Evolvable Hardware has mainly focused on digital and analogue electronic platforms that may be far from the most suitable medium for BIAs. Other mediums are of course viable candidates for EH. Miller and Downing [100] proposed that the effectiveness of EAs can be enhanced when the underlying medium has a rich and complicated physics. Further, that such new technologies and even new ways to compute may be devised, possibly even at the molecular level, providing many advantages over conventional electronic technology.

It is important to note from the work of Harding and Miller [85, 86] that (a) a black box approach can achieve computation, (b) the choice of material was biased by the commercial availability of the material and that electrical contacts were present, (c) the liquid crystal in fact stopped working—possibly indicating limited reconfigurability and (d) it is still a black box—how it worked no one knows.

Such black box approaches will always be limited due to natural scepticism to the application of BIAs, exploiting materials of any kind, where verification of the functionality achieved is all but impossible. Conventional design starts with an abstract computation model or framework for the design and a technology created to meet the requirements of the model, enabling a simple transformation from design to implementation. On the other hand, an evolved design is focused on a different mapping: that is a mapping between inputs and outputs of the design itself and pays little attention to either the computation model or the technology in hand (except where some form of optimization of the design is required). Thus, the difficulty of identifying the computational model behind an evolved design is perhaps not so surprising. One approach to address such an understanding is to study the computational properties of the design through theoretical approaches as in [101]. A further approach suggested by Stepney is that evolutionary techniques may be applied, not only to achieving functionality from a black box but also as a technique to identify such computational characteristics of the underlying material [102].

Looking to the ITRS Roadmap 2009 [42] we see that possible future materials for mainstream electronics include materials such as carbon-based nano-electronics, spin-based devices and ferromagnetic logic. Investigation of the suitability of such materials for the application of BIAs would be well worth investigation.

A further medium, not currently being investigated within the field of Evolvable Hardware, but highly relevant, is the within the field of synthetic biology where synthetic DNA is being generated to form logical functions. Further, libraries of such functions are under construction with the goal of creating biological circuits. BIAs may be able to play a key role in solving some of the challenges ahead. Rather than following the synthetic biology approach to achieving computation in a fashion similar to today’s electronics, a different approach would be to apply BIAs directly to the DNA substrate to enable computation to emerge.

Other possible platforms include molecular computing, optical systems and mixed mechanical–electrical systems. However, to really succeed in many such areas we need collaborators: (a) to help identify the problems and (b) help in the subject specific issues (for example what wet-ware is and is not possible).

8 Summary

While evolvable hardware does have its place, we now know that, at least in terms of silicon design, it is much more limited than was previously suggested. However, we still have a research area that is full of possibilities and potential. Turning the lessons learnt and technological advances made into real progress is still a challenge.

One thing is for certain, if we want to keep the field of evolvable hardware progressing for another 15 years we need to take a fresh look at the way we think about evolvable hardware: about which applications we should be applying it to; about where it will actually be of real use; about where it might finally be accepted as a valid way of creating systems and about what medium it should be based in.

We wish us all luck in this, the second generation of evolvable hardware.

References

1. X. Yao, T. Higuchi, Promises and challenges of evolvable hardware. IEEE Trans. Syst. Man. Cybern. C **29**(1), 87–97 (1999)
2. T. Higuchi, M. Iwata, I. Kajitani, H. Iba, T. Furuya, B. Manderick, Evolvable hardware and its applications to pattern recognition and fault tolerant systems. Towards Evolvable Hardw. Evol. Eng. Approach, LNCS **1052**, 118–135 (1996)
3. S.D. Scott, A. Samal, S. Seth, HGA: a hardware based genetic algorithm, in *Proceedings of ACM/SIGDA 3rd International Symposium on FPGA's*, 1995, pp. 53–59
4. M. Salami, G. Cain, Implementation of genetic algorithms on reprogrammable architectures, in *Proceedings of the Eighth Australian Joint Conference on Artificial Intelligence (AI'95)*, 1995, pp. 121–128
5. X. Yao, Evolutionary artificial neural networks. Int. J. Neural Syst. **4**(3), 203–222 (1993)
6. X. Yao, Y. Liu, Evolving artificial neural networks for medical applications, in *Proceedings of 1995 Australia-Korea Joint Workshop on Evolutionary Computation*, 1995, pp. 1–16
7. X. Yao, Y. Liu, Towards designing artificial neural networks by evolution, in *Proceedings of International Symposium on Artificial Life and Robotics (AROB)*, 1996, pp. 265–268, 18–20 Feb 1996
8. X. Yao and Y. Liu, Evolving artificial neural networks through evolutionary programming, in *The Fifth Annual Conference on Evolutionary Programming* (MIT Press, 1996), pp. 257–266
9. M.A. Rosenman, An evolutionary model for non-routine design, in *Proceedings of the Eighth Australian Joint Conference on Artificial Intelligence (AI'95)*, (World Scientific Publ. Co., Singapore, 1995), pp. 363–370
10. L. Davis, *Handbook of Genetic Algorithms* (Van Nostrand Reinhold, New York, NY, 1991)
11. T. Higuchi et al., Real-world applications of analog and digital evolvable hardware. IEEE Trans. Evol. Comput. **3**(3), 220–235 (1999)
12. A. Thompson, On the automatic design of robust electronics through artificial evolution, in *Proceedings of the International Conference on Evolvable Systems: From Biology to Hardware*, 1998, pp. 13–24
13. J.A. Walker, J.A Hilder, A.M. Tyrrell, Evolving variability-tolerant CMOS designs, in *Proceedings of the International Conference on Evolvable Systems: From Biology to Hardware*, 2008, pp. 308–319
14. S. Stepney, R.E. Smith, J. Timmis, A.M. Tyrrell, Towards a conceptual framework for artificial immune systems. Artif. Immune Syst., LNCS **3239**(2004), 53–64 (2004)

15. *ispPAC30 Data Sheet* (Lattice Semiconductor Corporation, 2001), http://www.latticesemi.com/lit/docs/datasheets/pac/pacover.pdf
16. A. Stoica, D. Keymeulen, A. Thakoor, T. Daud, G. Klimech, Y. Jin, R. Tawel, V. Duong, Evolution of analog circuits on field programmable transistor arrays, in *Proceedings of NASA/DoD Workshop on Evolvable Hardware (EH2000)*, 2000, pp. 99–108
17. J. Langeheine, J. Becker, F. Folling, K. Meier, J. Schemmel, Initial studies of a new VLSI field programmable transistor array, in *Proceedings 4th International Conference on Evolvable Systems: From Biology to Hardware*, 2001, pp. 62–73
18. *Virtex Field Programmable Gate Arrays Data Book Version 2.5*, (Xilinx Inc., 2001). http://www.xilinx.com
19. *User Manual and Tutorials for the CELL MATRIX MOD 88*. http://www.cellmatrix.com
20. E. Sanchez, D. Mange, M. Sipper, M. Tomassini, A. Perez-Uribe, A. Stauffer, Phylogeny, ontogeny, and epigenesis: three sources of biological inspiration for softening hardware, in *Evolvable Systems: From Biology to Hardware*, *ICES 96*, 1996, pp. 35–54
21. A.M. Tyrrell, E. Sanchez, D. Floreano, G. Tempesti, D. Mange, J.M. Moreno, J. Rosenberg, A.E.P. Villa, POEtic tissue: an integrated architecture for bio-inspired hardware, in *Proceedings of 5th International Conference on Evolvable Systems*, (Trondheim, 2003), pp. 129–140
22. A.J. Greensted, A.M. Tyrrell, Extrinsic evolvable hardware on the RISA architecture', in *7th International Conference on Evolvable Systems*, (Wuhan, China, 2007), pp. 244–255, September 2007
23. J. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs* (MIT Press, Cambridge, MA, 1994)
24. J. Koza, M. Keane, M. Streeter, What's AI done for me lately? Genetic programming's human-competitive results. IEEE Intell. Syst. **18**(3), 25–31 (2003)
25. J. Koza, J. Yu, M.A. Keane, W. Mydlowec, Use of conditional developmental operators and free variables in automatically synthesizing generalized circuits using genetic programming, in *Proceedings of the Second NASA/DoD Workshop on Evolvable Hardware*, 2000, pp. 5–15
26. M. Keane, J. Koza and M. Streeter, Automatic synthesis using genetic programming of an improved general-purpose controller for industrially representative plants, in *Proceedings of the 2002NASA/DOD Conference On Evolvable Hardware*, ed by A. Stoica, 2002, pp. 113–122
27. M. Streeter, M. Keane, J. Koza, Routine duplication of post-2000 patented inventions by means of genetic programming, in *Genetic Programming: 5th European Conference*, *EuroGP 2002*, ed by J. Foster et al. (eds), 2002, pp. 26–36
28. J. Koza, L.W. Jones, M.A. Keane, M.J. Streeter, S.H. Al-Sakran, Toward automated design of industrial-strength analog circuits by means of genetic programming, in *Genetic Programming Theory and Practice II*, Chap. 8, 2004, pp. 121–142
29. E. Takahashi, Y. Kasai, M. Murakawa, T. Higuchi, Post fabrication clock-timing adjustment using genetic algorithms, in *Evolvable Hardware*, (Springer, 2006), pp. 65–84
30. M. Murakawa, S. Yoshizawa, I. Kajitani, T. Furuya, M. Iwata, T. Higuchi, Hardware evolution at function level, in *International Conference on Parallel Problem Solving in Nature*, *PPSN 1996*, 1996, pp. 62–71
31. I. Kajitani, T. Hoshino, N. Kajihara, M. Iwata, T. Higuchi, An evolvable hardware chip and its application as a multi-function prosthetic hand controller, in *Proceedings of 16th National Conference on Artificial Intelligence (AAAI-99)*, 1999, pp. 182–187
32. A. Stoica, T. Arslan, D. Keymeulen, V. Duong, X. Gou, R. Zebulum, I. Ferguson, T. Daud, Evolutionary recovery of electronic circuits from radiation induced faults, in *IEEE Congress on Evolutionary Computation, IEEE CEC*, 2004, pp. 1786–1793
33. D. Linden, Optimizing signal strength in situ using an evolvable antenna system, n *Proceedings of the 2002 NASA/DOD Conference On Evolvable Hardware*, 2002, pp. 147–151
34. J.D. Lohn, G. Hornby, A. Rodriguez-Arroyo, D. Linden, W. Kraus, S. Seufert, Evolutionary design of an x-band antenna for NASA's space technology 5 mission, in *3rd NASA/DoD Conference on Evolvable Hardware*, 2003, pp. 1–9
35. M.L. Minsky, S.A. Papert, *Perceptrons* (MIT Press, Cambridge, MA, 1969)
36. S. Grossberg, Contour enhancement, short-term memory, and constancies in reverberating neural networks. Stud. Appl. Math. **52**, 213–257 (1973)
37. E. Bryson, Y.-C. Ho, Applied Optimal Control: Optimization, Estimation, and Control. (Blaisdell Publishing Company, New York, 1969)

38. D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Letters to nature. Nature **323**, 533–536 (1986)
39. M. Hartmann, P.K. Lehre, P.C. Haddow, Evolved digital circuits and genome complexity, in *NASA International Conference on Evolvable Hardware 2005*, 2005, pp. 79–86
40. J. Ziv, A. Lempel, A universal algorithm for sequential data compression. IEEE Trans. Inform. Theory **IT-23**(3), 337–343 (1977)
41. K. Kobayashi, J.M. Moreno, J. Madrenas, Implementation of a power-aware dynamic fault tolerant mechanism on the Ubichip platform, in *International Conference on Evolvable Systems: From Biology to Hardware (ICES10)*, 2010, pp. 299–399
42. *International Technology RoadMap for Semiconductors* (2009)
43. A. Thompson, Evolutionary techniques for fault tolerance, in *International Conference on Control*, 1996, pp. 693–698
44. P.C. Haddow, M. Hartmann, A. Djupdal, Addressing the metric challenge: evolved versus traditional fault tolerant circuits, in *the 2nd NASA/ESA Conference on Adaptive Hardware and Systems*, 2007, pp. 431–438
45. T. Yu, S. Lee, Evolving cellular automata to model fluid flow in porous media, in *2002 NASA/DoD Conference on Evolvable Hardware*, 2002, pp. 210–217
46. R.S. Zebulum et al., Experimental results in evolutionary fault recovery for field programmable analogue devices, in *Proceedings of the NASA/DOD International Conference on Evolvable Hardware*, 2003, pp. 182–186
47. A. Stoica et al., Temperature-adaptive ciruits on reconfigurable analog arrays, in *IEEE Aerospace Conference 2007*, 2007, pp. 1–6
48. T. Kalganova, An extrinsic function-level evolvable hardware approach. Genetic Programming, Lecture Notes in Computer Science, vol. 1802, pp. 60–75 (2004)
49. J. Torresen, A scalable approach to evolvable hardware, in *The International Conference on Evolvable Systems: From Biology to Hardware*, *(ICES98)*, 1998, pp. 57–65
50. J. Torresen, Scalable evolvable hardware applied to road image recognition, in *The second NASA International Conference on Evolvable Hardware*, 2000, pp. 245–252
51. T. Kalganova, Bidirectional incremental evolution in extrinsic evolvable hardware, The second NASA/DoD Workshop on Evolvable Hardware, 2000, pp. 65–74
52. W. Liu, M. Murakawa, T. Higuchi, ATM cell scheduling by functional level evolvable hardware, in *Proceedings of the First International Conference on Evolvable Systems*, 1996, pp. 180–192
53. V.K. Vassilev, Scalability problems of digital circuit evolution: evolvability and efficient design, in *Proceedings of the 2nd NASA/DoD Workshop on evolvable Hardware*, 2000, pp. 55–64
54. F. Gomaz, R. Miikulainen, Incremental evolution of complex general behaviour, in *Special issue on Environment Structure and Behaviour, Adaptive Behaviour* , vol 5, Issue 3–4. (MIT Press, 1997), pp. 317–342
55. R.A. Brooks et al. Alternative essences of intelligence, in *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)* (AAAI Press, 1998), pp. 961–967
56. J. H. Hong, S.B. Cho, MEH: modular evolvable hardware for designing complex circuits, in *IEEE Congress on Evoutionary Computation*, 2003, pp. 92–99
57. E. Stomeo, T. Kalganova, C. Lambert, Generalized decomposition for evolvable hardware. IEEE Trans. Syst. Man Cybern. B **36**(5), 1024–1043 (2006)
58. E. Stomeo, T. Kalganova, Improving EHW performance introducing a new decomposition strategy. *2004 IEEE Conference on Cybernetics and Intelligent Systems*, 2004, pp. 439–444
59. T. Gordon, P.J. Bentley, Towards development in evolvable hardware, in *Proceedings of the NASA/DoD Conference on Evolvable Hardware*, 2002, pp. 241–250
60. P.J. Bentley, Exploring component-based representations ? the secret of creativity by evolution, in *Fourth International Conference on Adaptive Computing in Design and Manufacture*, 2000, pp. 161–172
61. F. Gruau, Neural network synthesis using cellular encoding and the genetic algorithm, PhD thesis, France, 1994
62. H. Kitano, Designing neural networks using genetic algorithm with graph generation system. Complex Syst. **4**, 461–476 (1990)
63. P.J. Bentley, S. Kumar, Three ways to grow designs: a comparison of embryogenies for an evolutionary design problem. in *Genetic and Evolutionary Computation Conference (GECCO 99)*, 1999, pp. 35–43

64. H. Kitano, Building complex systems using development process: an engineering approach. in *Evolvable Systems: From Biology to Hardware*, ICES, Lecture Notes in Computer Science, (Springer, 1998), pp. 218–229
65. A.A. Siddiqi, S. Lucas, A comparison of matrix rewriting versus direct encoding for evolving neural networks. in *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, 1998, pp. 392–397
66. P. Eggenberger, Creation of neural networks based on development and evolutionary principles, in *Proceedings of the International Conference on ANNs*, 1997, pp. 337–342
67. H. Hemmi, J. Mizoguchi, K. Shimohara, Development and evolution of hardware behaviours. Towards Evolvable Hardw. LNCS **1062–1996**, 250–265 (1996)
68. C. Ortega, A.M. Tyrrell, A hardware implementation of an embyonic architecture using virtex FPGAs, in *Evolvable Systems: From Biology to Hardware*, ICES, Lecture Notes in Computer Science, 2000, pp. 155–164
69. P.C. Haddow, G. Tufte, P. ven remortel, Shrinking the genotype: L-systems for EHW? *International Conference on Evolvable Systems: From Biology to Hardware*, 2001, pp. 128–139
70. J. Koza, M.A. Keane, M.J. Streeter, The importance of reuse and development in evolvable hardware, in *Proceedings of the 2003 NASA/DoD Conference on Evolvable Hardware*, 2003, pp. 33–42
71. J.F. Miller, P. Thomson, A developmental method for growing graphs and circuits, in *Proceedings of the 5th International Conference on Evolvable Systems (ICES03)*, 2003, pp. 93–104
72. G. Tufte, P.C. Haddow, Towards development on a silicon-based cellular computing machine. J. Natural Comput. **4**(4), 387–416 (2005)
73. H. Liu, J.F. Miller, A.M. Tyrrell, Intrinsic evolvable hardware implementation of a robust biological development model for digital systems, in *Proceedings of the 2005 NASA/DoD Conference on Evolvable Hardware*, 2005, pp. 87–92
74. P. van Remortel, J. Ceuppens, A. Defaweux, T. Lenaerts, B. Manderick, Developmental effects on tunable fitness landscapes, in *Proceedings of the 5th International Conference on Evolvable Systems, ICES2003*, 2003, pp. 117–128
75. D. Roggen, D. Federici, Multi-cellular development: is there scalability and robustness to gain? in *Proceedings of Parallel Problem Solving from Nature 8*, *PPSN 2004*, 2004, pp. 391–400
76. P.K. Lehre, P.C. Haddow, Developmental mappings and phenotypic complexity, in *Proceedings of the Congress on Evolutionary Computation (CEC2003)*, 2003, pp. 62–68
77. G. Tufte, Phenotypic developmental and computation resources: scaling in artificial development, in *Genetic and Evolutionary Computation Conference*, 2008, pp. 859–866
78. J.A. Walker, J.F. Miller, The automatic acquisiation, evolution and re-use of modules in Cartesian genetic programming. IEEE Trans. Evol Comput **12**(4), 1–21 (2008)
79. J.A. Walker, J.F. Miller, Evolution and acquisition of modules in Cartesian genetic programming, in *Proceedings of the 7th European Conference Genetic Programming (EuroGP 2004)*, vol. 3003, Lecture Notes in Computer Science, 2004, pp. 187–197
80. J.F. Miller, P. Thomson, Aspects of digital evolution: geometry and learning, in *Proceedings of the International Conference on Evolvable Systems: From Biology to Hardware*, 1998, pp. 25–35
81. Z. Vazilicek et al., On evolutionary synthesis of linear transforms in FPGA, in *International Conference on Evolvable Systems: From Biology to Hardware 2008*, LNCS, **5216**, 141–152 (2008)
82. A. Thompson, An evolved circuit, intrinsic in silicon, entwined with physics, *1st International Conference on Evolvable Systems 1996* (Springer, 1996), pp. 390–405
83. J. Lohn, G. Hornby, Evolvable hardware using evolutionary computation to design and optimize hardware systems, IEEE Computational Intelligence Magazine, feb., 2006, pp. 19–27
84. S.L. Harding, J.F. Miller, E.A. Rietman, Evolution in Materio: exploiting the physics of materials for computation. Int. J. Unconv. Comput. **4**(2), 155–194 (2008)
85. S.L. Harding, J.F. Miller, Evolution in materio: a tone discriminator in liquid crystal, *Congress on Evolutionary Computation,* 2004, pp. 1800–1807
86. S.L. Harding, J.F. Miller, Evolution in materio: investigating the stability of robot controllers evolved in liquid crystal, *The international Conference on Evolvable Systems: From Biology to Hardware*, 2005, pp. 155–164
87. S. H. Mahdavi, P. Bentley, Evolving motion of robots with muscles, in *Applications of Evolutionary Computing*, LNCS, **2611**, 149–155 (2003)
88. M. Oteam, Switchable glass: a possible medium for evolvable hardware, *First NASA/ESA Conference on Adaptive Hardware and Systems*, 2006, pp. 81–87

89. A. Thompson, Hardware evolution: automatic design of electronic circuits in reconfigurable hardware by artificial evolution, Distinguished Dissertation Series, Springer, 1998
90. M. Garvie, A. Thompson, Evolution of combinatonial and sequential on-line selfdiagnosing hardware, in *Proceedings of the 5th NASA/DoD Workshop on Evolvable Hardware*, 2003, pp. 177–183
91. J.D. Lohn, G.V. Larchev, R.F. Demara, A genetic representation for evolutionary fault recovery in Virtex FPGAs, in *Proceedings of the 5th International Conference on Evolvable Systems: From Biology to Hardware (ICES)*, 2003, pp. 47–56
92. K. Zhang, R.F. Demara, C.A. Sharma, Consensus-based evaluation for fault isolation and on-line evolutionary regeneration, in *Proceedings of the 6th International Conference on Evolvable Systems: From Biology to Hardware (ICES05)*, 2005, pp. 12–24
93. F. Corno, G. Cumani, M.S. Reorda, G. Squillero, Efficient machine-code test-program induction. in *Proceedings of the Congress on Evolutionary Computation (CEC)*, *IEEE*, 2002, pp. 1486–1491
94. T. Pecenka, Z. Kotasek, L. Sekanina, J. Strnadel, Automatic discovery of RTL benchmark circuits with predefined testability properties, in *Proceedings of the NASA/DoD Conference on Evolvable Hardware*, 2005, pp. 51–58
95. T. Pecanka, L. Sekanina, Z. Kotasek, Evolution on synthetic rtl benchmark circuits with predefined testability. ACM Trans. Design Autom. Electron. Syst. **13**(3), 1–21 (2008)
96. K. Kobayashi, J.M. Moreno, J. Madreas, Implementation of a power-aware dynamic fault tolerant mechanism on the Ubichip platform, *International Conference on Evolvable Systems: From Biology to Hardware*, 2010, pp. 299–309
97. A. Djupdal, P.C. Haddow, Evolving efficient redundancy by exploiting the analogue nature of CMOS transistors, *Fourth International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS)*, 2007, pp. 81–86
98. D. Keymeulen, R.S. Zebulum, Y. Jin, A. Stoica, Fault-tolerant evolvable hardware using field-programmable transistor arrays. IEEE Trans. Reliab. **49**(3), 305–316 (2000)
99. G. Greenwood, A.M. Tyrrell, Metamorphic systems: a new model for adaptive systems design, in *Proceedings of the Congress on Evolutionary Computation*, 2010, pp. 3261–3268
100. J.F. Miller, K. Downing, Evolution in materio: looking beyond the silicon box, *NASA/DoD Conference on Evolvable Hardware (EH'02)*, 2002, pp. 167–178
101. L. Sekanina, Evolvable hardware: from applications to implications for the theory of computation, Unconv. Comput. LNCS, **5715**, 24–36 (2009)
102. S. Stepney, The neglected pillar of material computation. Physica D **237**(9), 1157–1164 (2008)