NTNU

Norwegian University of
Science and Technology

A Study of Evolvable Hardware

Øyvind Aukrust Rones

Submission date: April 18th, 2017

1 Literature Search Protocol

1.1 Research questions

To guide the initial literature search, a series of research questions related to Evolvable Hardware (EH) were posed.

RQ1 In what ways can EH implement fault tolerance?

RQ2 What are the main disadvantages or challenges of EH?

RQ3 What does the current research into evolvable hardware focus on?

RQ4 How effective is EH at adapting to new environments?

1.2 Review protocol/Search strategy

A review protocol was then defined by listing a set of search words for each of the research questions. Instead of searching several individual publishers, it was decided that using the search engine *google scholar* would suffice as it links to several of the best known publishers.

Search strings:

- "Evolvable hardware"+"advantages"
- "Evolvable hardware"+"challenges"
- "Evolvable hardware"+"repair"
- "Evolvable hardware"+"adaptive"
- "Evolvable hardware"+"fault tolerance"
- "Evolvable hardware"+"applications"

To aid in the selection of papers a series of selection criteria were proposed. *However, while searching for existing implementations of EH, papers dated as early 2000 was included.*

Inclusion criteria	**Exclusion criteria**
Specific applications clearly stated	Dated earlier than 2013
Relevant keywords	Too general

2 Introduction to Evolvable Hardware

There is an ever-increasing demand for robust, adaptable embedded systems. Although somewhat limiting, this is generally implemented in software in today's systems. Evolvable Hardware seeks to improve on this by letting the hardware itself adapt to changes.

2.1 Background

In the field of Evolvable Hardware (EH) the aim is to implement a system that can autonomously alter it's own hardware architecture and behavior through interacting with its environment. EH applies evolutionary algorithms to hardware design by simultaneously keeping a population of individual solutions. As in evolutionary algorithms each solution is assigned a fitness score and the next generation is produced by applying variation operators and selection mechanisms. EH can be implemented in two different ways [3, 9]:

Extrinsic Evolvable Hardware: The population is simulated in software with the best individual of the final generation the only one to be physically implemented.

Intrinsic Evolvable Hardware: Each individual is implemented and tested in hardware.

3 Reflection

The advantages and challenges of EH are many, some of which will be mentioned here.

3.1 Advantages

Improved search space Evolutionary algorithms excel at searching solutions spaces. Depending on the problem, they can find designs that are radically different from any human design, yet has significantly improved characteristics. Further, in the design process, the human mind often abstracts components to a manageable dimension, while EH can work at the lowest level. For instance, a human designer might consider a logic gate to be a perfectly digital device, while it is in fact a high gain analog device. In

many cases, EH systems have been based on the underlying physical aspects of electrical components (see section 4.1).

Adaptivity Although relatively untouched, one of the main attractions of EH is its potential for adaptability [3]. By letting the problem constraints and the fitness function depend on input from the surrounding environment, EH could be made highly adaptable.

Fault tolerance Related to adaptivity, fault tolerance and self repair is another area of interest. By having the EH adapt to faults, autonomous repair can be realized, or a new circuit can be implemented that circumvent the detected fault.

3.2 Challenges

Scalability One of the biggest problems of EH is the problem of scaling [1–3]. The number of generations required to evolve a circuit increases with the number of inputs [1], and the complexity of evolved circuits are limited compared to conventional design techniques.

Fitness evaluation Evaluating the fitness of an individual can be time consuming. First, the genotype of each individual must be translated to a phenotype and, in the of case an intrinsic system, must then be configured in hardware before fitness is finally calculated. This can all be quite resource intensive and the main bottle neck of all the operations in EH. This is especially the case when the model input is large as the evaluation time grows exponentially [3].

Online operation While offline operation can be run with virtually unlimited resources and without timing constraints, online operation is much more limited. Full or partial autonomy is required with little to no human oversight. The computational resources will be limited, and in addition the evolutionary algorithm might not be allowed to run indefinitely before a solution is needed. Additionally, the reason an adaption is needed, such as a fault or a change of the environment, is often unknown. Consequently, intrinsic EH is most often used [2].

3.3 Addressing advantages and challenges

When designing an EH system, the aforementioned challenges should considered as well as the advantages. Although far from being a fully developed field, there are still ways performance can be improved despite the challenges. Firstly, the objects to be optimized should be chosen appropriately. For instance, one would probably want to maximize the amount of correct output given a series of inputs to ensure that the solution behaves in accordance with the specifications. Additionally, other objectives could be proposed, like power usage or perhaps, in the case of offline circuit design, the cost of implementing the solution in hardware. For online operation however, the fitness evaluation should be designed with the available computational resources and timing constraints in mind. To partially circumvent the scalability problem, the search space can be reduced by increasing the complexity of the basic units the circuit is made out of, albeit at the cost of additional components. This would be done by basing the EH on an FPAA instead of an FPGA (mentioned in section 4.1.

Fault-tolerance could be implemented in several ways depending on how well known the intended environment is. By simulating the environment a series of different solutions could be implemented offline before operation. Should a circuit suddenly fail, another can easily take its place. If the environment is less known, one could use an initial solution and continuously attempt to improve it, then pick the best individual when a fault is detected. This has the added benefit of most likely having a circuit ready at the time a fault is detected, albeit at the cost of continuously requiring computational resources.

4 Applications

4.1 Existing implementations

EH is normally implemented by interconnecting several functional blocks in different patterns. Depending on the nature of the problem, several platforms can be used, with field programmable arrays being popular.

- FPAA (Field Programmable Analogue Arrays)
- FPGA (Field Programmable Gate Arrays) [3, 4, 6]
- FPTA (Field Programmable Transistor Arrays) [3, 5]

EH has been applied with success to fields such as image processing, control systems and robotic design. Salvador et al. [6] successfully implemented an evolvable, autonomous image filter on an FPGA. Interestingly, the evolved filter did not degrade the image in any way regardless of how many times it was filtered. A fault-tolerant system was also implemented [7].

Both a digital XNOR and an analog multiplier circuit designer were implemented on an FPTA by Keymeulen et al. [5]. The EH approach was shown to be equally robust as classical fault-tolerant designs, while requiring less computational time.

Another interesting application is found in the work of teerakittikul et al., and is a great example of how EH can be used to adapt to faulty hardware. A four-wheeled robot was controlled by a standard motor controller before suddenly being disconnected from one the wheels. At this point a new controller was evolved in an FPGA to adapt a new controller for each wheel. Instead of randomly initializing each individual, the population was evolved from a single, formerly trusted individual to improve the time required to find a new solution [8].

4.2 Future work

It is clear that one of the most promising areas of EH is found in its potential for autonomous adaptability and fault tolerance, and this is where it should be headed in the future. Yet, the problem of scalability will have to be addressed if EH is to compete, beyond the simplest circuits, with conventional silicon-based circuit design. Improving online operation speed is related to the scalability problem and should also be researched further.

5 Conclusion

The field of evolvable hardware has many exciting aspects and, more importantly, a lot of potential. It is clear that the improved search space of evolutionary algorithms and their innate adaptability may yield hardware designs that improve conventional designs in a multitude of ways, most notably in the way of fault-tolerance. The challenges are many however, with two of the biggest being the problem of scalability and efficient online operation. Consequently, this is where research should be focused in the future for this field to advance further.

References

[1] F. Cancare, S. Bhandari, D. B. Bartolini, M. Carminati, and M. D. Santambrogio. A bird's eye view of fpga-based evolvable hardware. In *Adaptive Hardware and Systems (AHS), 2011 NASA/ESA Conference on*, pages 169–175. IEEE, 2011.

[2] G. W. Greenwood and A. M. Tyrrell. *Introduction to evolvable hardware: a practical guide for designing self-adaptive systems*, volume 5. John Wiley & Sons, 2006.

[3] P. C. Haddow and A. M. Tyrrell. Evolvable hardware challenges: Past, present and the path to a promising future. In *Inspired by Nature*, pages 3–37. Springer, 2018.

[4] T. Higuchi and X. Yao. *Evolvable hardware*. Springer Science & Business Media, 2006.

[5] D. Keymeulen, R. S. Zebulum, Y. Jin, and A. Stoica. Fault-tolerant evolvable hardware using field-programmable transistor arrays. *IEEE Transactions on Reliability*, 49(3):305–316, 2000.

[6] R. Salvador, A. Otero, J. Mora, E. de la Torre, T. Riesgo, and L. Sekanina. Self-reconfigurable evolvable hardware system for adaptive image processing. *IEEE transactions on computers*, 62(8):1481–1493, 2013.

[7] R. Salvador, A. Otero, J. Mora, E. de la Torre, L. Sekanina, and T. Riesgo. Fault tolerance analysis and self-healing strategy of autonomous, evolvable hardware systems. In *Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on*, pages 164–169. IEEE, 2011.

[8] P. Teerakittikul, G. Tempesti, and A. M. Tyrrell. The application of evolvable hardware to fault tolerant robot control. In *Evolvable and Adaptive Hardware, 2009. WEAH'09. IEEE Workshop on*, pages 1–8. IEEE, 2009.

[9] M. A. Trefzer and A. M. Tyrrell. *Evolvable hardware: From practice to application*. Springer, 2015.