Research Paper Engineering

Evolvable Hardware in Theory and Implementation

Wadekar Ameya Rajendra | Student ,Jawaharlal Nehru Engineering College, Aurangabad.

ABSTRACT

Evolvable Hardware (EHW) refers to HW design and self-reconfiguration using evolutionary/genetic mechanisms. In contrast to conventional hardware where the structure is irreversibly fixed in the design process, evolvable hardware (EHW) is designed to adapt to changes in task requirements or changes in the environment, through its ability to reconfigure its own hardware structure dynamically and autonomously. This capacity for adaptation, achieved by employing efficient search algorithms based on the metaphor of evolution, has great potential for the development of innovative industrial applications. By exploring a large design search space, EHW may find solutions for a task, unsolvable, or more optimal than those found using traditional design methods. During evolution it is necessary to evaluate a large number of different circuits which is normally most efficiently undertaken in reconfigurable hardware. For digital design, FPGAs (Field Programmable Gate Arrays) find many applications. Thus, this technology is applied in much of the work with evolvable hardware. The paper introduces EHW and outlines how it can be applied for hardware design of real-world applications.

KEYWORDS EHW, FPGAs, Moore's law, Genetic Algorithm, Scalability

INTRODUCTION

Moore's law stated in 1965 that "The number of transistors that can be placed inexpensively on an integrated circuit doubles approximately every two years" [8].The number of transistors becoming available for designers continue to increase as Moore's law seems to be valid for the development of new computer hardware. Earlier we have seen a limit in the *size* of hardware devices. However, we may very well soon see a limit in *designability*. That is, designers are not able to apply all the transistors in the largest integrated circuits becoming available. To overcome this problem, new and more automatic design schemes would have to be invented. One such method is evolvable hardware (EHW) [4]. Instead of manually designing a circuit, only input/output-relations are specified. The circuit is automatically designed using an adaptive algorithm inspired from natural evolution. These are special tools for solving optimization tasks. The principles of their work are based on biological processes. The key problem of this domain is the search for suitable configuration; a suitable configuration produces a system (circuit) that provides required function. The design of evolvable hardware consists of two main tasks. The first task presents the selection of useful reconfigurable structure. The second one is about finding a suitable configuration for this structure; the system has to solve the optimization problem. [3]

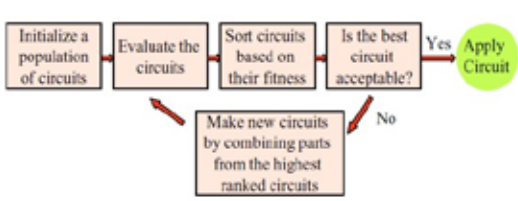


Fig. 1. The algorithm for evolving circuits.

EVOLUTIONARY ALGORITHMS

Evolutionary algorithms are often used to evolve a circuit on evolvable hardware to obtain a circuit that meets or exceeds the required fitness. Evolvable hardware implements evolutionary algorithms on a piece of hardware. The algorithm is illustrated in Fig. 1. In this algorithm, a set (population) of circuits i.e. circuit representations, are first randomly generated. The behavior of each circuit is evaluated and the best circuits are combined to generate new and hopefully better circuits .Thus, the design is based on incremental improvement of a population of initially randomly generated circuits. Circuits among the best ones have the highest probability of being combined to generate new and possibly better circuits. The evaluation is according to the behavior initially specified by the user. After a number of iterations, the fittest circuit is to behave according to the initial specification. The most commonly used evolutionary algorithm is genetic algorithm (GA) . The algorithm, which follows the steps described above, contains important operators like crossover and mutation of the circuit representations for making new circuits. The operations are very similar to those found in natural evolution as seen in Fig. 2. [3]

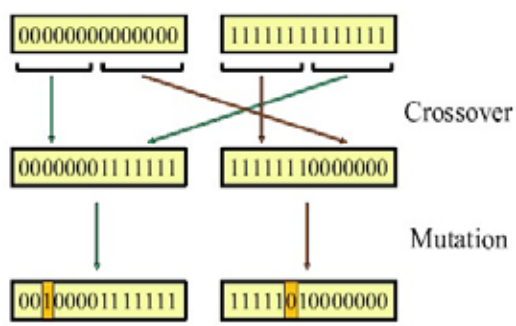


Fig. 2. The genetic algorithm operators.

EVOLVABLE HARDWARE

Before any actual calculations are made, the engineer describes a fitness function, a function that assigns each candidate a fitness value relating to how well that candidate meets or exceeds the solution specification. If a member has a fitness level greater than the target fitness level the hardware considers the problem solved and returns the acceptable circuit (member) and quits iterating. It is important to note that an algorithm generates a large group of individuals, models evolution in iterations, runs each individual against the fitness function, and continues this until an individual is evolved with a set of genes that passes the fitness function.

3.1 Intrinsic and Extrinsic Evolution

Post-tone-discriminator research has split evolvable hardware

into two approaches: intrinsic and extrinsic evolution. In the former, each candidate is implemented on physical hardware, very often an FPGA [2]. In the latter, each candidate is simulated on reconfigurable circuits, often FPGAs .The intrinsic approach has the benefit of being very accurate with regards to adapting theoretical solution to actual solution. Where intrinsic evolution takes a large amount of time to rework the physical circuit, extrinsic systems simply implement an evaluation and simulates the new circuit. In practice extrinsic evolution is more common to obtain a working or near-working solution which needs to be tweaked. [2][5]

4. FPGA IMPLEMENTATION OF EHW

At the core of all evolvable hardware is code that passes down the positive traits of past iterations. There are other tactics besides the passing of beneficial traits, but trait passing is at the heart of evolving a circuit to match a fitness function. What follows is a walkthrough of a hypothetical implementation of a genetic algorithm on an FPGA that uses tactics such as cloning and mutation in addition to trait-passing to evolve a circuit to meet a desired fitness. [9]

4.1 Field Programmable Gate Arrays

In general, an FPGA contains a grid (or field) of configurable logic blocks (CLBs) which can be either single logic operations (AND, NOR, etc) or groups of logic blocks that perform higher level functions (adder, multiplier, etc) depending on the sophistication of the FPGA. Each CLB is connected to the grid via a switch matrix that allows or disallows a connection to be made via a programmable interconnect point (PIP). The system is fed input which runs through the logic blocks in a manner determined by the state of the PIPs and gives an output which is evaluated with the fitness function.

4.2 Inheritance in Algorithms

For most of EHW, especially that on FPGAs, implementation code is an evolutionary algorithm. Generally speaking, the hardware starts out by randomly generating a group of circuits to act as the first generation. Each circuit is either Simulated (extrinsic) or constructed (intrinsic) and evaluated with the fitness function. It is important to note here

that the members chosen are not always the most fit, but more fit members are weighted heavier to pass on their genes than less fit members, hence the parallel to real world evolution. The processor of the FPGA weights each member based on fitness of genes. Two selected members will combine their complementing genes in what is normally called a crossover. Aside from direct crossovers which take genes solely from the parents, there are also randomly occurring mutations. Mutations serve the function of introducing new and possibly beneficial genes into the candidates gene pool. In this way, a gene that was erroneously eliminated earlier can be introduced into the population again. In an FPGA implementation adding or subtracting chromosomes is done by opening or closing PIPs [6]. Occasionally cloning will take place. Cloning in this sense is when a member, instead of mating with another member and producing an offspring of mixed traits, passes its full set of traits to the offspring, often only the case when a member has a very high fitness value.

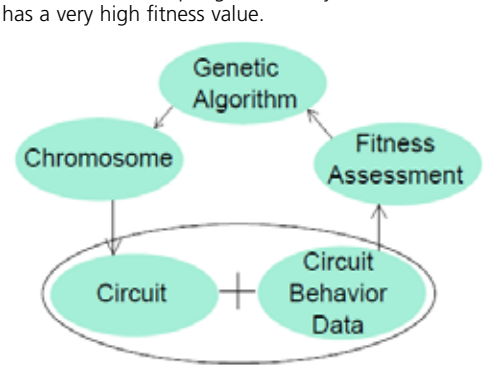


Fig.3 The life cycle of an evolutionary algorithm (here a genetic algorithm) that continually creates new circuits to test against a fitness function.

4.3 Generation Cycling

When the number of offspring reaches the number of parents, a generation cycle occurs. Described in more detail in Section 5.2, generally speaking, a generation cycle is when the offspring (group of just-generated candidates) become the parents for a new generation. Generation cycling is possible because each generation is stateless. That is, once a generation is generated it no longer requires the past generation so the past generation can be overwritten by a new generation .Typically, a given problem requires a number of generations in the tens or hundreds until a suitable candidate is discovered. The number of generations depends on many factors including how difficult it is to exceed a fitness bar given the requirements of the problem and the number of parameters each candidate must evolve. [1]

5. DIFFICULTIES AND LIMITATIONS OF FPGA BASED EHW

5.1 Scalability

Evolvable hardware implementing genetic algorithms does not, by design, scale well. The more complex a "parent" is the harder it is to generate a representative population of children. The way an evolutionary algorithm works when applied to evolutionary hardware, the parent generation generates offspring until there are as many children as there are members in the parent generation. At that time the generations cycle; the children become the parent generation and the former parent generation are deleted to make room for the next generation of offspring. One can easily see that the number of offspring the algorithm generates is limited not by how many offspring make good candidates but by how many members the hardware can support.[10][7]

5.3 Interconnect Faults

In FPGAs there exists four basic types of interconnect faults. Interconnect faults are very small hardware faults involving a switchable data path being permanently stuck open or stuck closed. Since FPGAs contain an extremely large amount of interconnects, it is feasible for errors to occur.. Line segment faults are faults within an FPGA between programmable interconnect points (defined previously). The faults come in two states; open and closed. Line segment open faults are caused when line segments lose their ability to establish the connection between the pair of PIPs they join Line segment closed faults are caused when line segments maintain a permanent connection between their pair of PIPs, regardless of whether they are supposed to or not. Programmable interconnect point faults are faults inside of the PIPs themselves. As discussed previously, PIPs have a series of connections inside them and are vulnerable to faults. PIPs also have two types of faults: stuck-open faults and stuck-closed faults. Stuck-open faults are a problem because a PIP will not be able to allow data through that point. Stuck-closed faults have the opposite problem; they will pass all information through that connection. [6]

Application	EA	TE	AR	BB	THW	FC	EV	SC
Adaptive Equalizer [18]	GA	D	CD	Neuron	Custom	ONL	On-chip	S
Ampl. and Filter Design [15]	GA	A	CD	T/R/L/C	Custom	OFL	Off-chip	S
Analog Circuit Synthesis [12]	GP	A	CD	R/L/C	Custom	OFL	Off-chip	S
Character Recognition [31]	GA	D	CD	Gate	Comm.	OFL	Off-chip	S
Clock Adjustment [27]	GA	D	CT	Gate	Custom	ONL	Off-chip	S
Digital Filter Design [16]	GA	D	CD	Gate	–	OFL	Off-chip	S
Gene Finding [39]	GA	D	CD	Gate	Comm.	OFL	Off-chip	S
IF Filter Tuning [17]	GA	A	CT	Filter	Custom	ONL	Off-chip	S
Image Compression [22]	GA	D	CT	Pixel	Custom	ONL	On-chip	D
Image Compression [23]	GA	D	CD	Gate	Comm	ONL	On-chip	D
Multi-spect. Image Rec. [20]	GA	D	CT	Function	Comm.	OFL	Off-chip	S
Number Recognition [7]	GA	D	CD	Gate	Comm.	OFL	Off-chip	S
Prosthetic Hand [9]	GA	D	CD	Gate	Custom	ONL	Complete	S
Road Image Rec. [33]	GA	D	CD	Gate	Comm.	OFL	Off-chip	S
Robot Control [10]	GA	D	CD	Gate	Comm.	ONL	Complete	D
Robot Control [28]	GA	D	CD	Gate	Comm.	ONL	Off-chip	S
Sonar Classification [38]	GA	D	CD	Gate	Comm.	OFL	Off-chip	S

Table 1. Characteristics of EHW applied to real-world applications.

6. APPEAL OF EVOLVABLE HARDWARE

Evolvable hardware has seen a love-hate relationship with the

academic research community and is slowly being adopted as a legitimate (and not novel) way of solving problems. First, evolvable hardware had to be proven possible on consumer hardware. It found a home in the FPGA community where it was discovered that one could overlay a genetic algorithm over an FPGA and obtain cheap, effective evolvable hardware. Then, evolvable hardware had to jump the hurdle of having real-world applications. That was shown to be possible to a large extend in 1996 by Adrian Thompson with his sub-40 gate tone discriminator. Currently, evolvable hardware is being held back by scalability issues caused by complexity growth. Scalability with regard to complexity is essential for evolvable hardware to be considered in real world applications (many of which are prohibitively complex for evolvable hardware to be employed [10]).

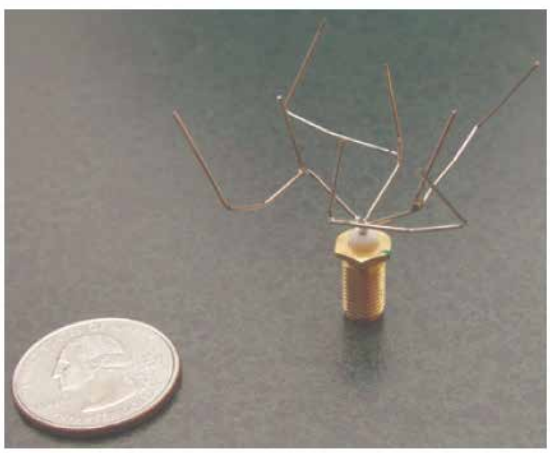


Fig 4.Orignal Evolved Antenna

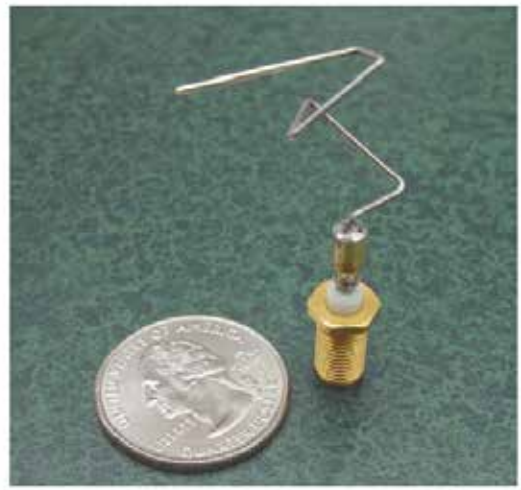


Fig 5. Later generation evolved antenna without branching

CONCLUSIONS

In this research, we have gone through the short history of evolvable hardware and shown an implementation on a major medium known as a field programmable gate array. Although EHW is young, it has been able to bridge the gap from novelty to real world solutions on many occasions [5]. There are, however, several hurdles currently being worked on which need to be overcame before EHW can provide solutions to general problems. As it stands, EHW has provided many compact and elegant solutions to application-specific problems. Future efforts, hopefully, will prove EHW as a tool used by researchers and end users to solve complex problems.

REFERENCES

[1] Evolvable Hardware: Evolution in FPGAs. ACME Seminar. | | [2] An Evolvable Hardware FPGA for Adaptive Hardware, La Jolla, CA , USA, 2000. The 2000 Congress on Evolutionary Computation. | [3] D. Goldberg. Genetic Algorithms in search, optimization, and machine learning. Addison. | Wesley, 1989. | | [4] T. Higuchi et al. Evolvable hardware: A first step towards building a Darwin machine. In Proc. | of the 2nd International Conference on Simulated Behavior, pages 417.424. MIT Press, 1993. | | [5] A Bird's Eye View of FPGA-based Evolvable | Hardware, Milano, Italy, 2011. 2011 NASA/ESA | Conference on Adaptive Hardware and Systems | (AHS-2011). | [6] Application-Independent Testing of FPGA | Interconnects, volume 24. Application-Independent | Testing of FPGA Interconnects, November 2005. | [7] On Dependability of FPGA-Based Evolvable Hardware Systems That Utilize Virtual Reconfigurable Circuits. The Third Conference on Computing Frontiers, 2006. | | [8] The Wider Impact of Moore's Law. IEEE, September 2006. | | [9] FPGA Circuits for Evolvable Hardware, volume 2. The Second International Symposium on Electrical and Electronics Engineering, ISEEE, 2008. | [10] J. D. Lohn and G. S. Hornby. Evolvable hardware: Using evolutionary computation to design and optimize hardware systems. IEEE Computational | Intelligence Magazine, pages 19{27, February 2006.} |