Evolvable Hardware

Xin Yao

School of Computer Science
The University of Birmingham
Edgbaston, Birmingham B15 2TT, UK
Email: x.yao@cs.bham.ac.uk
URL: http://www.cs.bham.ac.uk/~xin

Organisation of the Talk

1. What Is Evolvable Hardware (EHW)
2. EHW as an Alternative to Electronic Circuit Design
3. EHW as an Adaptive System
4. Other EHW-Related Work
5. Concluding Remarks

What Is Evolvable Hardware (EHW)

There are different views on what EHW is, depending on the purpose of EHW.

1. Some regard EHW as the application of evolutionary computation techniques to circuit synthesis.
2. EHW is hardware which is capable of on-line adaptation through reconfiguring its architecture dynamically and autonomously.
3. EHW refers to hardware that can change its architecture and behaviour dynamically and autonomously by interacting with its environment. (My preferred description)

EHW as an Alternative to Circuit Design

1. Designing VLSI chips and boards.
2. Evolving HDL (Hardware Description Language) programs.
3. Evolving PLDs (Programmable Logic Devices).
4. Unconstrained evolution of an electronic oscillator, etc.
5. Evolving analog circuits.
6. ... ...

Early Work and Two Major Approaches

- Early and some of the recent work related to EHW only dealt with *optimisation* of VLSI circuits, such as cell placement, logic minimisation and compaction of symbolic layout.
- Circuit *functions* were not designed/evolved by EAs.
- Recent work concentrates on evolving circuit architectures and thus functions. Two major approaches have been used:
  1. Indirect Approach, and
  2. Direct Approach

Indirect Approach to EHW Circuit Design

The indirect approach does not evolve hardware directly, but evolves an intermediate representation (e.g., trees) which specifies hardware circuits.

- Evolving digital circuits.
  E.g., SFL (Structured Function Description Language) programs (represented by production trees) can be evolved by a GA. A 4-bit binary adder was evolved successfully.
- Evolving analog circuits.
  E.g., work on evolving a lowpass "brick wall" filter, an asymmetric bandpass filter, an amplifier, etc. Trees were used to represent circuits. The results were competitive with human designs.

Direct Approach at Gate Level

- The direct approach evolves hardware circuit's architecture bits directly. It works well only with reconfigurable hardware, such as FPGA.
- The gate level evolution implies that the "atomic" hardware functional units are logical gates like *AND*, *OR*, and *NOT*. The evolution is used to search for different combinations of these gates.
- Typical examples include XOR, counters, FSMs (Finite State Machines), multiplexors, and an electronic oscillator.
- One argument for the direct approach is to exploit hardware resources by unconstrained hardware evolution.

Direct Approach at Function Level

- The gate level evolution runs into the scalability problem quickly.
- The function level evolution uses high-level "atomic" functions such as addition, multiplication, sine, cosine, etc., and thus is much more powerful.
- Typical examples: two-spiral, Iris, FSMs, image rotation.
- The work is better viewed as an attempt towards *adaptive hardware*, rather than as a design alternative.

Advantages of Evolutionary Design

1. Explores a larger design space.
2. Does not assume a priori knowledge.
3. Does not require exact specification and thus can design complex systems.
4. Constraints and special requirements can be imposed on the simulated evolution if necessary.
5. Some analog circuits might be too difficult or costly to design by human experts.

Scalability Issue

- Scalability of algorithm: Time complexity of the EA for EHW?
- Scalability of representation: Size of chromosomes vs. Size of EHW?
- Time is more crucial since the size of chromosome is usually polynomial in the size of EHW circuits.

Electronic Speed and the Scalability Problem

- There have been some expectations that the speed of simulated evolution would not be a problem in a few years as faster VLSI chips come out.
- This statement can be misleading. Electronic speed is important, but **not** a solution to the scalability problem.

The importance of the time complexity issue can be illustrated by an artificial example. If the time complexity of simulated evolution is $O(2^n)$, where n is the size of EHW, then an EHW with 10 components would need $2^{10} = 1024$ nanoseconds ($\approx 10^{-6}$ seconds) to evolve. An EHW with 100 components would need $2^{100} \approx 10^{30}$ nanoseconds ($> 10^{13}$ years).

Circuit Verification/Test and Fitness Evaluation

- How to verify the correctness of EHW? How to find a fitness function which guarantee the correctness of EHW? Is it practical to test all possible situations in which an EHW might be used?
- If all 4-bit numbers have been correctly added, would all 5-bit, 6-bit, etc., numbers be added correctly by the same circuit?
- Exploiting hardware resources is attractive. Has an EHW exploited something totally irrelevant, such as room temperature or minor Earth movement?
- How robust is EHW to minor environmental changes? Does it degrade gracefully?
- When to stop simulated evolution?

Summary on Evolutionary Circuit Design

Two fundamental issues:

1. Scalability. (It determines the size of EHW which are feasible.)
2. Circuit verification and fitness evaluation. (It determines the application domains of EHW.)

EHW as an Adaptive System

Current work on *adaptive EHW* can be classified into two major categories:

1. EHW controllers.
2. EHW recognisers and classifiers.

There are other types of work, such as compression, etc.

EHW Controllers

A number of control tasks can be performed by EHW, e.g., ATM control and robot control among others.

- Evolving an artificial ant to follow the John Muir Trail in simulation.
- Evolving a wall following robot in a simulated environment.
- Evolving an adaptive equaliser.
- Evolving an ATM traffic shaper.
- ... ...

EHW Recognisers and Classifiers

- Evolving FPGA to perform *learning* tasks, such as letter recognition, the comparator in a V-shape ditch tracer, two-spiral, Iris, FSMs, etc.
- Unlike most other studies, generalisation is explicitly emphasised in such work.
- A complexity (regularisation) term was usually included in the fitness evaluation function.

Some Challenges to Adaptive EHW

Scalability: Efficiency of simulated evolution.

Generalisation: Dealing with new environments.

On-line adaptation: Incremental evolution/learning.

Disaster prevention in fitness evaluation during on-line adaptation.

A Behavioural View Towards EHW

- What is being evolved? A circuit or the circuit's behaviours? In other words, what is actually being evaluated by a fitness function?
- Is it genetic evolution or behavioural evolution?
- The environment is crucial. Generalisation should be discussed with respect to environments.

Other EHW Work

1. Self-reproduction and self-repairing hardware.
2. Artificial “brains.”
   (a) CAM-BRAIN from ATR’s Department 6 (Evolutionary Systems).
   (b) Artificial Brain Systems at RIKEN. (No hardware implementation.)

Discussions

1. Population-based learning (simulated evolution) is good at following slow environmental changes, but not at real-time on-line adaptation. Individual learning should be introduced.
2. There is some existing work on evolutionary artificial neural networks and genetic programming which may be useful for EHW.
3. Co-evolution is a very promising approach to deal with the problem of fitness evaluation.

Concluding Remarks

1. EHW is a very promising research area although there are some open issues.
2. Evolutionary design is good at discovering novel circuits that are unlikely to be designed using the conventional approach.
3. Adaptive EHW has great potentials, but needs to deal with issues such as on-line adaptation and generalisation.
4. Evolvable hardware does not need to use evolutionary algorithms!