# EVOLUTIONARY SEARCH APPLIED TO RECONFIGURABLE ANALOGUE CONTROL

*Kester Clegg, Susan Stepney* *

Dept. of Computer Science
University of York, UK. YO10 5DD
email: {kester, susan}@cs.york.ac.uk

*Tim Clarke*

Dept. of Electronics
University of York, UK. YO10 5DD
email: tim@ohm.york.ac.uk

## ABSTRACT

The new breed of reconfigurable integrated circuits (ICs) offer switched-capacitor based analogue circuits whose functionality can be altered during run-time. Rapidly changing the functionality of an analogue circuit provides interesting opportunities for control systems. It also opens a large design space in which decisions have to be made regarding the frequency and form of reconfiguration. We present a bio-inspired architecture to facilitate the automated search for circuits on these platforms, based on well-established evolutionary algorithms. Unlike previous attempts at evolving single solutions, our genomes contain multiple solutions and use feedback provided by the circuit output to trigger reconfiguration of the IC.

## 1. INTRODUCTION

We present a bio-inspired architecture for analogue circuits that adaptively reconfigure to environmental changes. This reconfiguration process is inspired by the biological process of gene regulation: each cell in a biological organism has the same DNA, yet different genes are expressed in different contexts, and so cells behave differently in different places and at different times. In our architecture, analogue circuit components correspond to the "proteins" expressed by digital genes. Different subsets of these components are configured to make a full analogue circuit that behaves in a manner appropriate for the current context. We use evolutionary search to discover a suitable set of components and reconfiguration conditions for the different elements of a task.

The field of evolutionary computation has tended to focus on solving search-based optimisation tasks, leading to criticism that some of the most interesting properties of evolutionary search have been neglected as a result [1, 2]. Thompson's work in the mid-1990s [3, 4] inspired many to follow the route of evolving logic functions in hardware, although few stayed true to his aims of using evolution to exploit physical domains that are inaccessible to human design (with the notable exception of Miller [5, 6]). Instead, logic functions such adders were evolved to see if optimal usage of gates could improve on human designs or to compare the speed of evolutionary strategies. Indeed, the design of a one or two-bit adder has become a benchmark for gauging the performance of evolutionary algorithms (for example, [7, 8]). However, some of the issues raised by Thompson's work were key to our choice of analogue platform.

Evolutionary computation has a track record in analogue electronics, with patentable successes by John Koza and his colleagues [9]. As the construction of many thousands of distinct analogue circuits for fitness evaluation is impractical, Koza evaluated his evolving circuit populations using software simulations. However, software simulation, using programs such as SPICE, is generally several orders of magnitude slower than testing the same circuit in hardware, making the simulated evolution a slow process. Additionally, simulation imposes modelling constraints, not allowing the full richness of the physically embodied system to be exploited by the evolutionary process. Our platform avoids some of these issues and gives us the advantages of being able to rapidly test circuits in hardware.

Instead of evolving single solution analogue circuits, we evolve an architecture inspired by the process of gene expression. Genes code for proteins. Whether a gene actually expresses a protein or not is determined by its context: in particular, by whether certain other proteins *bind* to expression or inhibition sites on the DNA controlling that gene. For more on the biology of gene expression, see [10, 11, 12].

In section 2 we describe our architecture, and provide details of the platform and implementation. In section 3 we describe a prototype task for which we have evolved a reconfiguring solution.

## 2. ARCHITECTURE AND PLATFORM

In our architecture, the configuration of analogue components in a circuit is specified by the corresponding genes in a digital genome (Fig. 1). The context in which a par-

*Special thanks to Dave Lovell at Anadigm for helping us with reconfiguration of the FPAA and host application. The research is funded by the EPSRC and Microsoft Research UK. The hardware used was purchased from pump-prime funding for the York Centre for Complex Systems Analysis at the University of York.

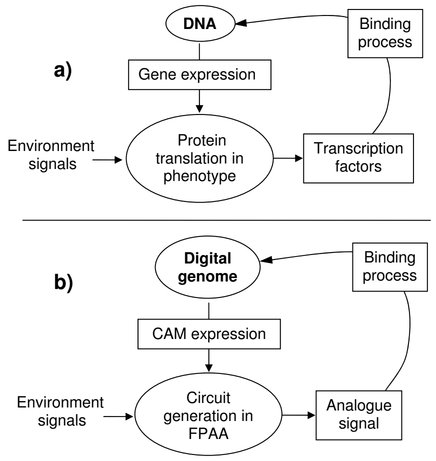

**Fig. 1**. Diagram (a) shows natural gene expression where expression is affected by the presence of transcription factors in the cell. (b) shows our implementation of the analogous process.

ticular component is deployed (expressed) is determined by the gene's binding site on the genome matching the binding protein's signature. The components expressed in any one context are configured to form an analogue circuit. The behaviour of that circuit depends on the context of its deployment (that is, on its input signal). The "binding protein" signature is constructed from the circuit's output signal. As that signal changes, new "protein" signatures are created, allowing new components to be expressed and so reconfiguring the circuit. The evolutionary process alters both the genes (details of component characteristics) and binding sites (context in which the components are deployed). The full details of our mapping from the biological process of context-specific gene expression to our architecture can be found in [13].

Switched-capacitor based ICs allow reconfigurable hardware analogue circuits. These ICs are similar in operation to FPGAs. Configurable Analogue ICs, sometimes called Field Programmable Analogue Arrays (FPAA)[1], implement an analogue circuit by downloading a configuration bitstream onto an IC. Downloading a new bitstream results in a new circuit being configured. An adaptable FPAA application can thus be implemented as a set of analogue circuits, with some host application controlling when the IC should reconfigure to a new circuit. In our architecture, the digital genome resides on the host, and as the binding signatures change component expression, a new configuration is calculated and downloaded onto the FPAA.

The Anadigm AN221E04 FPAA is split into 4 CABs (Configurable Analogue Blocks). Each CAB supports one or more Configurable Analogue Modules (CAMs). CAMs can contain circuits functioning as various band pass filters, multipliers, integrators, differentiators and so on. CAMs are configured by setting options, floating point parameters and clock speeds using the AnadigmDesigner software or via its API. As this API can be controlled by a host application, the host application is able to both create circuits and control the reconfiguration process.

The genomes and circuit specifications they encode are an adapted form of Cartesian Genetic Programming (CGP) [14]. The genotype in CGP is represented as a list of integers that encode the function and connection of each node in a feed-foward, directed graph.

In our genome, each gene encodes the CAM's ID and parameters, its connections to other CAMs, and the binding signature. There are around 200 CAM "primitives":[2] within most primitives, a range of parameterised behaviour is possible. We encode 4 parameter values; those unused in a particular CAM are ignored. Each parameter is encoded as a percentage of the parameter range. The connections to other CAMs can refer to CAMs earlier or later in the graph (unlike the strictly feed-forward structure of conventional CGP). The binding site is a string representing some combination of the 4 bases (ACGT) in DNA. Reconfiguration is triggered by the output signal changing and generating a new "binding protein" signature, which may match the binding sites of different genes, allowing them to be expressed. (Construction of the binding protein's signature is explained later.) The genotypes in an evolving population have a fixed number of genes, but the number of CAMs in a circuit depends on what is expressed in a given context.

Fig. 2 shows the stages of decoding a genome into a circuit description, depending on the binding signature. Stage 1 decodes the linear genome description into the corresponding directed graph representation. Stage 2 replaces each node with the corresponding CAM description. A CAM may have fewer input and output ports than are specified in the graph: connections to non-existent inputs are removed; unused outputs are left "open". This produces a description of a 'fully expressed' phenotype. Stage 3 matches the binding signature against each gene's binding site, which defines the subset of the CAMs that are actually expressed in the current context. Stage 4 turns this subset into a circuit configuration using the wiring connections of the fully expressed genome, rewiring to compensate for any missing CAMs. The circuit input is defined as that of the first expressed CAM; the wiring algorithm traces through the circuit to the first "open" output on a CAM, which is defined as

[1]Alternative names for these ICs include Field Programmable Transistor Arrays (FPTA) or Dynamically Programmable Analogue Signal Processors (*dp*ASP).

[2]When different configuration options are taken into account. The number is actually larger; however, for reasons involving the design of the API it easier to build up the CAM library omitting some of the option configurations.

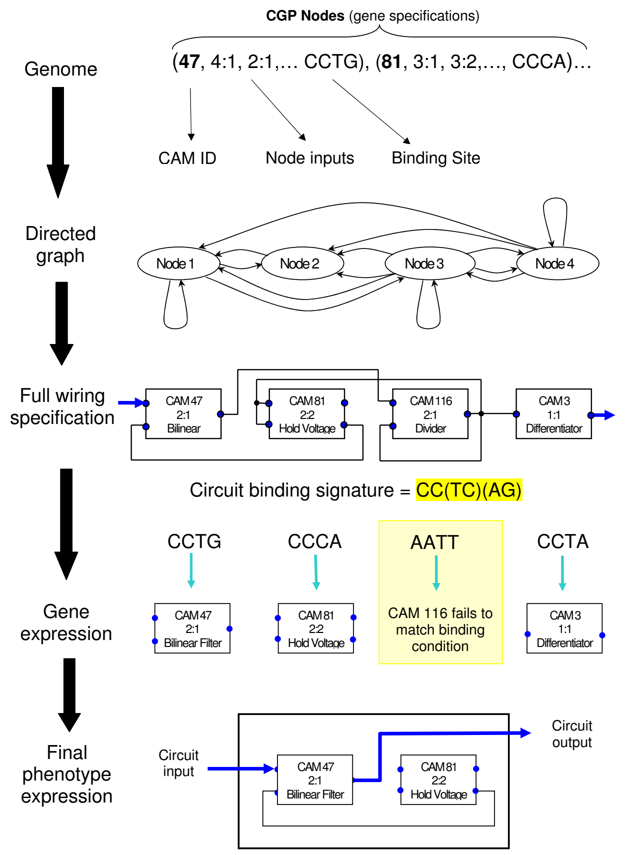

**Fig. 2**. An example of the genome decoding process. 1) 4 genes decoded into a directed graph of 4 nodes. 2) Nodes replaced by CAMs, giving fully expressed phenotype. 3) In this context, the binding signature matches the binding sites of 3 genes. 4) Corresponding subset of CAMs wired using fully expressed phenotype's wiring; one CAM ends up disconnected, leaving the final circuit containing two CAMs, only one of which is effective.

the circuit output. Note that this decoding process provides a highly indirect mapping from genome to phenotype.

The binding process provides feedback from the functional domain to the genome. The process converts the circuit's analogue output into a digital form by taking a wavelet transform of the output, normalising the coefficients and thresholding the values, to get a binary valued matrix with 4 rows and an application dependent number of columns. The 4 rows represent the 4 bases (ACGT). One value in a column represents the corresponding base. More than one value in a column represents a 'wildcard' that can match several bases. If a column has no values above the threshold, then it is ignored. The grid is read column by column to produce a string of bases and wildcards (the binding protein signature), which is matched against the gene binding sites.

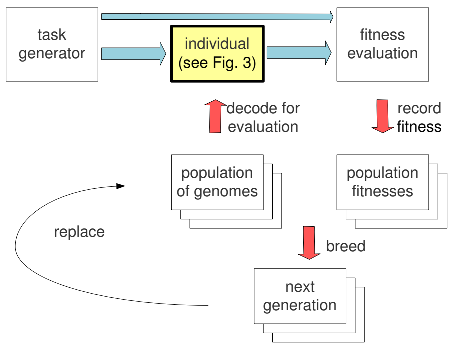

**Fig. 4**. The evolutionary harness.

As many positions are read as required to match a signature length. If the binding protein signature matches a binding site on the genome, that gene is expressed, resulting in a circuit reconfiguration.

## 3. PROTOTYPE APPLICATION

We have implemented a prototype of the architecture described and shown in Fig. 3. This is placed within an evolutionary harness to evolve a genome suitable to perform a particular task (Fig. 4). During evolution, an input signal is "ramped" up through a series of frequencies, starting at around 500Hz and rising to 20kHz. The task is to maximise power output for low and high frequency inputs, but to minimise power output for mid-range frequency inputs (specified by a fitness function on the Fourier transform of the output signal measuring the power at three test frequencies). This results in reconfiguration as the input signal increases in frequency. A 4+1 evolution strategy is used, with variable rates of mutation being applied to parts of the genome. The binding signature is calculated from the output signal every 0.5 seconds. When the signal changes and matches a new configuration, the genome is decoded and the new configuration is downloaded onto the FPAA. The process evolves genomes that successfully reconfigure to perform the task.

Some interesting behaviour was noted in the prototype. Certain configurations result in low output signals. The corresponding binding signature is largely random due to the amount of noise, as the wavelet coefficients are scaled up to give valid signatures. The effect of this is that the binding process starts "hunting" with random binding signatures, resulting in random expression. However, as soon as a good configuration is downloaded the "hunting" stops, an output signal is restored, and the binding process settles down. It is too early to tell whether evolution will make use of this aspect of the system.

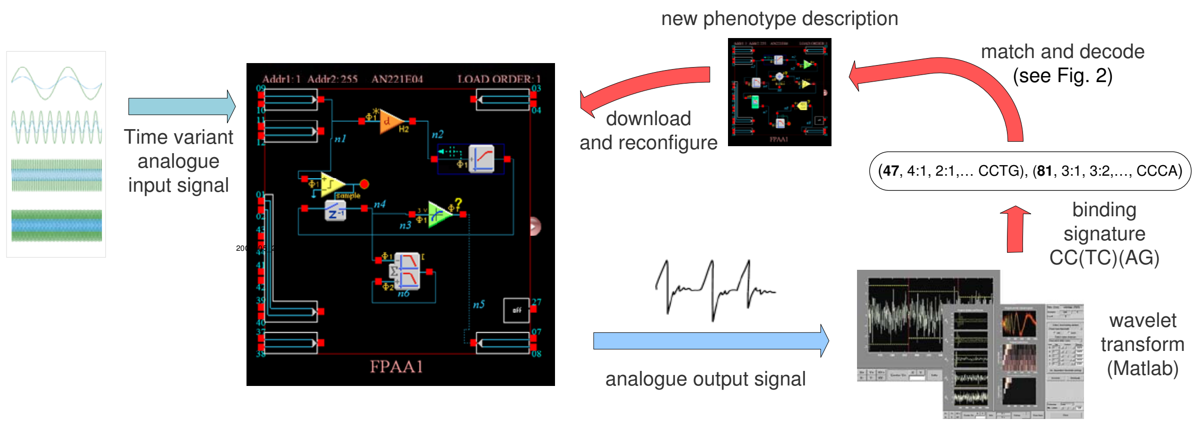
new phenotype description
match and decode (see Fig. 2)
download and reconfigure
Time variant analogue input signal
FPAA1
(**47**, 4:1, 2:1,... CCTG), (**81**, 3:1, 3:2,..., CCCA)
binding signature CC(TC)(AG)
wavelet transform (Matlab)
analogue output signal

**Fig. 3**. The prototype implementation, showing how output signals bind to new gene expressions through the feedback binding process.

## 4. CONCLUSION

This architecture incorporating feedback is a first attempt to physically reconstruct the exploratory mechanism of genetic regulatory networks. Our aim is to use this platform as a basis from which to evolve self-configuring analogue control systems. The prototype has demonstrated a proof of concept of many of the components of the architecture. We believe that without the mechanisms of interaction and feedback, digital genomes cannot guide themselves across functional search spaces in a way that fully exploits a domain's resources, and this is particularly true where that domain includes the complexity provided by real-world physics.

## 5. REFERENCES

[1] W. Banzhaf, G. Beslon, S. Christensen, J. Foster, F. Kepes, V. Lefort, J. Miller, M. Radman, and J. Ramsden, "Guidelines: From artificial evolution to computational evolution: a research agenda." *Nature Reviews Genetics*, vol. 7, no. 9, p. 729, September 2006.

[2] S. Kumar and P. Bentley, Eds., *On Growth, Form and Computers*. Elsevier Academic Press, 2003.

[3] A. Thompson, "Silicon evolution," in *Genetic Programming 1996: Proceedings of the First Annual Conference*. MIT Press, 1996, pp. 444–452.

[4] ——, "An Evolved Circuit, Intrinsic in Silicon, Entwined with Physics," in *Proc. 1st Int. Conf. on Evolvable Systems (ICES'96)*. Springer, 1997, pp. 390–405.

[5] S. Harding and J. F. Miller, "Evolution in materio : A real-time robot controller in liquid crystal," in *Proc. 2005 NASA/DoD Conference on Evolvable Hardware*. IEEE Press, 2005, pp. 229–238.

[6] J. F. Miller and K. Downing, "Evolution in materio: Looking beyond the silicon box," in *Proc. 2002 NASA/DoD Conference on Evolvable Hardware*. IEEE Computer Society, 2002, pp. 167–176.

[7] S. Kazadi, Y. Qi, I. Park, N. Huang, P. Hwu, B. Kwan, W. Lue, and H. Li, "Insufficiency of piecewise evolution," in *Proc. 2001 NASA/DoD Workshop on Evolvable Hardware*, 2001, pp. 223–231.

[8] T. Fogarty, J. Miller, and P. Thomson, "Evolving Digital Logic Circuits on Xilinx 6000 Family FPGAs," *Soft Computing in Engineering Design and Manufacturing*, pp. 299–305, 1998.

[9] J. R. Koza, L. Jones, M. Keane, and M. Streeter, "Towards industrial strength automated design of analog electrical circuits by means of genetic programming," in *Genetic Programming Theory and Practice II*, U.-M. O'Reilly, T. Yu, R. L. Riolo, and B. Worzel, Eds. Kluwer, 2004, ch. 8.

[10] S. Carroll, *Endless Forms Most Beautiful: The New Science of Evo Devo and the Making of the Animal Kingdom*. Weidenfeld & Nicolson, 2006.

[11] L. Wolpert, "Relationships between development and evolution," in *On Growth, Form and Computers*, P. Bentley and S. Kumar, Eds. Elsevier, 2003.

[12] S. B. Carroll, J. K. Grenier, and S. D. Weatherbee, *From DNA to Diversity*. Blackwell, 2001.

[13] K. Clegg, S. Stepney, and T. Clarke, "Using Feedback to Regulate Gene Expression in a Developmental Control Architecture," in *Proc. GECCO 2007*, (to appear).

[14] J. F. Miller and P. Thomson, "Cartesian Genetic Programming," in *Genetic Programming, Proceedings of EuroGP'2000*, ser. LNCS, R. Poli, W. Banzhaf, W. B. Langdon, J. F. Miller, P. Nordin, and T. C. Fogarty, Eds., vol. 1802. Springer, 2000, pp. 121–132.