Analogue Circuit Control through Gene Expression

Kester Clegg, Susan Stepney

Dept. of Computer Science, University of York
{kester|susan}@cs.york.ac.uk

Abstract. Software configurable analogue arrays offer an intriguing platform for automated design by evolutionary algorithms. Like previous evolvable hardware experiments, these platforms are subject to noise during physical interaction with their environment. We report preliminary results of an evolutionary system that uses concepts from gene expression to both discover and decide when to deploy analogue circuits. The output of a circuit is used to trigger its reconfiguration to meet changing conditions. We examine the issues of noise during our evolutionary runs, show how this was overcome and illustrate our system with a simple proof-of-concept task that shows how the same mechanism of control works for progressive developmental stages (canalisation) or adaptable control (homoeostasis).

1 Background and motivation

We present a system for the automated discovery and deployment of software configured analogue circuits. High level circuit components (for example, analogue filters) are modelled as 'genes'. Genes are 'expressed' in response to circuit output; as conditions change new genes are expressed, triggering the deployment of a new circuit. Gene characteristics are defined in a genome representation discovered using evolutionary algorithms.

Evolutionary computation has a proven record as a technique for search-based optimisation [1–3]. The method has also been used to automate design discovery [4, 5]. To date most work in the field focuses on the *performance* of evolutionary search. Fewer researchers are questioning the precepts on which stochastic, population-based algorithms are founded, whether their evolutionary model corresponds to modern biological thinking [6] or the role of developmental processes in evolution [7]. The latter is particularly important with regard to how gene expression explores the functional search space [8].

Thompson's ground-breaking work in the mid-1990s established that search algorithms could be used to discover novel hardware configurations that lie outside conventional engineering knowledge [9]. Thompson concentrated on *innovation* rather than rapid discovery, and was careful to design his experiments so that the search could exploit physical characteristics not normally incorporated into an engineer's design [10]. Following on from this, further examples using evolutionary algorithms in software configured physical media have come from

Miller and Harding [11, 12], and in both real and simulated physical environments from Stoica and others [13].

The problem of analogue circuit design has been met using evolutionary computation by John Koza and his colleagues [14]. In Koza's work, the evolutionary design process and fitness evaluations are run as software simulations. However, physical simulation by software imposes limits on what search algorithms can find. If one wants to expose the algorithms to resources outside conventional design knowledge, it is necessary to allow software to interact with the richer physics provided by real hardware. An alternative example using analogue circuit simulation and evolutionary computation comes from Mattiussi and Floreano [15, 16]. They propose a generic model to represent genetic regulatory networks, metabolic networks, neural networks and analogue electronic circuits as 'analogue networks'. The representation and variable strength of links between components are found using evolutionary algorithms, and like Koza's discoveries, the network designs are claimed to be human-competitive [16].

Most representations in evolutionary computation evolve a single solution tested against a static problem, often with an almost one-to-one correspondence between the genotype and phenotype. This rigid translation contrasts with one of the most remarkable qualities of natural organisms: the ability of their encoding to permit adaptability in different conditions, particularly during developmental stages, yet retain a large degree of homoeostasis or canalisation. The eventual form of a natural organism is taken from a multitude of potential 'solutions' within its DNA, each of which differs and only one of which is expressed. We have tried to increase the distance between digital genotypes and physical, decoded phenotypes. In our architecture, the indirect translation gives genes different roles according to their context of expression. The model of gene expression drew inspiration from the descriptions of evolutionary developmental processes by Carroll, Wolpert and others [8, 17]. Details of the model and how it maps to this biology can be found in [18], with more details on the hardware in [19].

2 Platform and System description

Switched-capacitor based arrays allow the implementation of software configurable analogue circuits. Sometimes called Field Programmable Analogue Arrays (FPAA),[1] these platforms implement an analogue circuit by downloading a configuration bitstream onto an integrated circuit (IC). A typical FPAA application can be thought of as a set of analogue circuits, with a hosted application controlling when to reconfigure to a new circuit. Circuits on the Anadigm AN221E04 FPAA are composed of Configurable Analogue Modules (CAMs) that can contain filters, multipliers, integrators, differentiators, etc. or even signal or power sources. They are configured by setting options, floating point parameters and clock speeds. An application hosted on a PC is able use the Anadigm API to both create circuits and control the reconfiguration process.

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

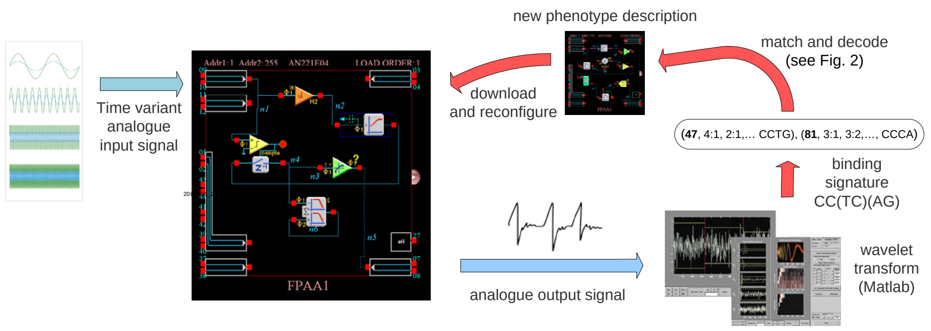

Fig. 1. System overview: analogue outputs are converted into digital binding signatures. Genes with binding sites that match the binding signature are expressed in the next circuit reconfiguration.

Our genome circuit specifications are represented using an adapted form of Cartesian Genetic Programming (CGP) [20] in a 4+1 evolutionary scheme. The genotype in CGP is a list of integers that encode the function and connection of each node in a directed graph. Nodes correspond to genes in our genome. Each gene encodes a CAM ID, its parameters, connections to other CAMs and its binding site. The binding site for each gene is a string representing some combination of the 4 bases (ACGT) in DNA. Reconfiguration is triggered by circuit output changing and generating a new 'binding signature', which is matched against gene binding sites in the genome. A gene that matches is expressed.

Due to the risk of exceeding the on-chip resources if all genes are expressed, we limit each genome to 3 genes giving 7 possible circuits (plus the 'empty' circuit which is ignored). In our experiments, the number of possible circuits is more than twice the number that can be deployed, ensuring there a degree of redundancy in the genome. As the wiring specification of the full genome does not match the wiring of partially expressed subsets of genes, the decoding process gives highly context dependent results: being expressed does not guarantee a gene then plays a functional part in a circuit, as it may have lost input or output connections depending on which other genes were also expressed [19].

The binding process provides feedback from the functional domain to the genome. It converts the circuit's analogue output into a binding signature by taking a wavelet transform of the output, normalising the coefficients and thresholding to get a binary valued matrix, with 4 rows corresponding to the 4 DNA bases, and a task dependent number of columns. More than one value in a column represents a 'wildcard' position that can match several bases, e.g. A{AC}GT is a signature of length 4 with 2 wildcards in position 2. If a column has no values above the threshold it is ignored. The grid is read column by column to produce a binding signature of fixed bases and wildcards which is continuously matched against the binding sites (see Fig. 1). As many positions are read as required to match the binding site lengths.

3 Task description

The prototype was set a task with two primary objectives a) to evolve reliable reconfiguration b) to configure the IC in stages according to different fitness criteria. To meet these objectives, it was decided to input a series of frequencies (1kHz, 5kHz and 10kHz) in steps of fixed duration . The fitness criteria were to minimise power output from the chip on the 1kHz and 10kHz steps, and maximise power output from the 5kHz step.

Each phenotype starts against the output of a 'bare wire' (i.e. a straight connection between input and output), so that for the population to move to higher fitness a reconfiguration must take place. A frequency step is input for 5 seconds to allow the system to settle, after which the wavelet transform is converted to a binding signature and checked against each gene binding site. Each match is added to the list of genes that will be expressed to make up the next circuit. Reconfiguration takes place while input continues for a further 5 seconds, allowing the system to settle again. Finally, the Fourier transform and power reading for that circuit is taken and converted into a fitness score. The input is then changed to the next frequency step and the process repeated, or a new test is started.

The fitness scores are based on a formula that moves the selection process in the direction of circuits that maximise power in the middle frequency band:

$$Fitness = (B + R) - (A + C) \tag{1}$$

where B, A and C are the power readings at mid, lower and upper frequencies respectively. R is the reconfiguration bonus. As mentioned previously, the phenotype must initially respond to the output of a bare wire that passes the input signal through unaltered (except for a slight reduction in voltage). If no reconfiguration occurs, the power readings return negative fitness values. Due to noise in the system and a result of elitism in a 4+1 evolutionary scheme, phenotypes that do nothing (i.e. fail to reconfigure) can get selected above those that reconfigure badly if power readings are used exclusively as the basis of selection. As this imposes an unacceptable delay for the search process to 'get started', we penalise any phenotype that fails to reconfigure from the initial setting. Conversely phenotypes that reconfigure — even to a bad circuit — have their fitness scores augmented by a small bonus (usually less than 10% of final fitness score). On runs for random sequence frequency steps (see §6), more reconfigurations are required so bonuses are accordingly reduced.

4 Preliminary Runs

4.1 Noise

There are two principal areas for noise to affect testing phenotypes. The first is during wavelet transform, where wavelet coefficients measure similarity of the wavelet to a signal as it is time shifted at different scales. Part of our binding

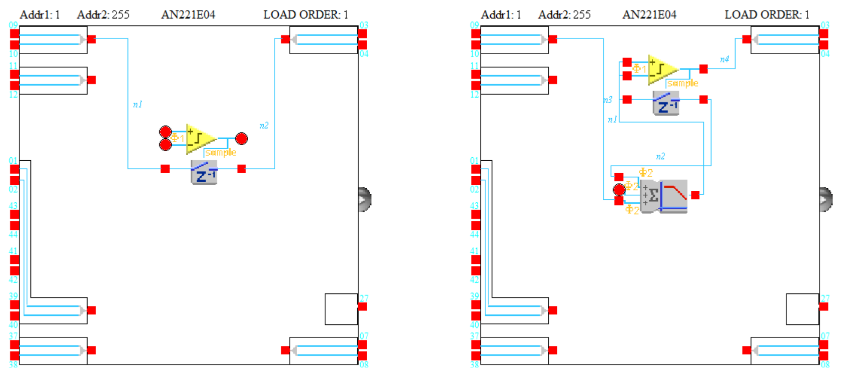

Fig. 2. The circuits deployed by an unreliable champion phenotype. Left circuit (HoldVoltageControlled) was downloaded at 1kHz, the circuit on the right (with SumFilter added) was downloaded at 5kHz and 10kHz.

process converts the wavelet coefficients into a four-row grid of binary values determined by a threshold. The more noise present in a signal, particularly a very weak signal, the more likely it will have coefficients scaled over the threshold. This translates into columns with more than one value — giving ‘wildcards’ for that binding position — meaning that more genes will be expressed as a result.

During evolutionary runs, no attempt is made to ensure gene expressions make valid circuits and it is possible that circuits may be broken (no output) or the signal output may be severely reduced or otherwise made noisy. In these cases, as coefficients are scaled, the noise introduced into the binding process results in many wildcards being produced for all positions. An unexpected effect of this is that a particular circuit can be downloaded by a ‘lucky’ binding sequence (i.e. one brought about by random noise due to a previous stage’s bad circuit). The new circuit may get a high fitness score. However, the binding that caused the good circuit to download might never be repeated. Elitism results in the evolutionary process being ‘conned’ into keeping and mutating the poor solution for many generations. The mutations never make successive phenotypes improve above the lucky fitness score of the original phenotype, as the binding that allowed the good circuit to be downloaded occurred by chance, leaving the evolutionary process stranded on a false peak of high fitness.

A second source of noise in the system comes from the Fourier transforms at each frequency step. These readings are susceptible to fluctuations due to variations in heat and interference from surrounding electrical and computer appliances. Power readings of the same circuit configuration will therefore always vary from one test to another. These fluctuations are small but can vary by as much as 40%, although the figure is generally closer to 10% depending on the CAMs involved.

Aside from these examples, a more serious source of fluctuations was noticed after the first week of preliminary runs. Despite the degree of expected noise, some champion phenotypes would not reproduce anything close to the scores they achieved during the evolutionary run. For example, a phenotype that had scored 4130 during one run was re-tested five times at the end of the run. It scored successively -1067, -1036, 300, 4096, -1993. On closer inspection, it was noticed that despite the huge variation in fitness scores, the phenotype was reconfiguring using the same circuits at each frequency step. Two circuits were used (see Fig. 2) by the phenotype. The tests were repeated with the circuits loaded separately to observe their behaviour at each step. Neither circuit produced the high power rating in any of the frequency steps. In fact the circuits seemed inert and produced no output at all. Each of the circuits was then downloaded while input signals were being put through the chip. At first nothing happened, but then at random the switch in one circuit latched producing a high power output. Once downloaded the circuit reversed the switch at high frequency so that no output signal was produced at that frequency and no reconfiguration was required to produce a high fitness score. The reason for this behaviour lies in the fact that CAMs such as those shown in Fig. 2 exhibit hysteresis. Certain CAM behaviour depends on the input to a comparator (GainSwitch, GainPolarity, etc.). If the two inputs of the comparator are connected together then even small amounts of noise are going to lead to the CAM behaviour 'flipping'.[2]

Rather than exclude such CAMs from the pool of 'primitives' that evolution could select from, we decided to test each phenotype 5 times, taking the median result for our fitness scores. This strategy, although increasing the length of each run fivefold, had the effect of reducing the worst variability due to noise or unexpected CAM behaviour such as hysteresis. The results can be seen in Fig 3, where the heavy lines are the champion phenotype scores, while the light lines are the best of a generation. Runs that tested each phenotype once show best of generation scores varying considerably from the current champion. Taking the median of 5 tests gives best of generation scores much closer to the current champion, and produces final solutions that seem more stable with respect to the forms of noise mentioned.

4.2 Evolutionary parameters

Tests were performed over 75 generations using a 4+1 evolutionary scheme. In this scheme, the best of previous generations is cloned, mutated and the selection process repeated. In our scheme, it was necessary to introduce variable mutation. Variable mutation is based on the numbers in each generation, so that the first clone receives no mutations per gene, the second one mutation per gene, the third two mutations and so on. This gave a maximum of 4 mutations per gene — enough to move poor phenotypes some distance from their parents, but also allow gradual mutations of good phenotypes. This can be verified from the selection history, where early progress is often made by the most heavily mutated phenotypes, but later stages rely on small improvements to good solutions.

[2] Our thanks to Dave Lovell of Anadigm for this explanation.

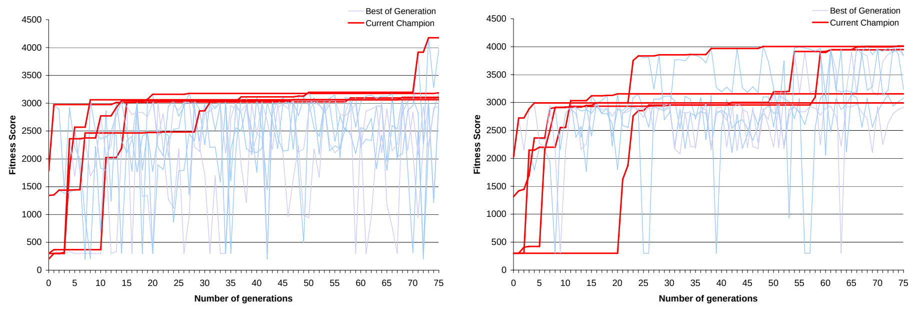

Fig. 3. Five runs using a binding signature length of two: one test per phenotype (left) and median of 5 tests per phenotype (right). The charts show current champion score (bold lines) against the best of each generation. Taking the median results in more stable phenotypes.

Initial tests tried up to 500 generations, but further improvement was rare after 50 or 60 generations. This may be because the tests were too easy for the phenotypes to solve, as it became clear that there were many ways of achieving similar levels of high fitness. Another possibility is that relatively few mutations were required to reach the higher levels of fitness once an initial solution had been found.

In addition to reconfiguration bonuses, further 'bootstrapping' assistance was provided by allowing genes to have 'seeded' binding sites. Thus for a bare wire configuration, the binding signature produces a set of signatures that run through the frequencies (if no reconfiguration occurs) from AA, C/G, to TT. Having binding sites of length 2 gave a good chance of matches being found, but to speed up the search we seeded initial generations with binding site bases known for signatures with a bare wire configuration at the initial frequency step. For example, the first generation in a step-up specialist environment might have all binding sites seeded with A or C bases, as the bare wire output signature for 1kHz is generally AAAA (only first two positions used).

5 Experiment

As part of our interest in gene expression during developmental stages, we set up a simple hypothesis to test not only if phenotypes could be adaptable, but also whether having such adaptability incurs a fitness cost.

Hypothesis: phenotypes that evolve to cope an in unpredictable environment perform less well than phenotypes that evolve as specialists when both phenotypes are placed in a predictable environment.

To test this, the three frequency steps were considered developmental stages during which different behaviour was required from the phenotype. In specialist environments the steps in frequency either increased or decreased, in non-

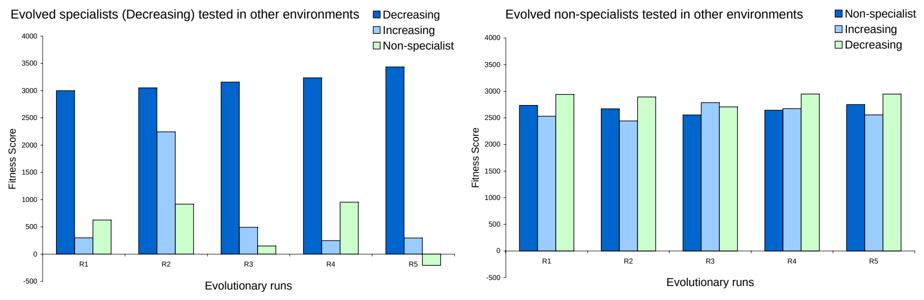

Fig. 4. Results of specialists (left) and non-specialists (right) tested across all environments. Dark bars show phenotype scores in their own environment.

specialist environments all possible transitions occurred. In all environments the fitness test remained the same: maximise power in the 5kHz band, minimise it elsewhere. As a phenotype uses circuit output to trigger a reconfiguration, the current configuration is crucial to how the phenotype configures the next step. For example, a specialist phenotype may have configured to a high pass filter with gain suitable for high fitness during a 5kHz stage. On stepping up to 10kHz, this filter produces a binding signature that the phenotype can use to configure to the next circuit. However, if the same step occurs as part of a sequence that runs from 10kHz to 5kHz and back to 10kHz, the previously unknown step from 10kHz to 5kHz may cause the phenotype to configure the chip to another circuit. The step from 5kHz to 10kHz now results in a different output signature and the previously used genes for the good circuit no longer match. Specialists evolving in predictable environments can therefore both profit and suffer from previously encountered steps in a non-specialist environment. To counter this potential bias, the non-specialist environment starts with a bare wire configuration at 5kHz and tries to avoid known steps where possible.

6 Results

Each of the specialist and non-specialist phenotypes were evolved over 75 generations. The champion phenotype was then tested in environments it had not evolved in. The tests used the average of 2 results (each the taking median of 5 scores, as during evolutionary runs). The results, shown in Fig. 4, show poor capability for specialists in their 'opposite' specialist environment (where the frequency steps are reversed) and in the non-specialist environment. The reason seems largely the result of a strategy employed by phenotypes in the final stage of specialist environments. Circuits that reconfigure to 'broken' last circuits (at 1kHz or 10kHz stages) help their fitness scores and have nothing to lose from reconfiguring to a circuit that no longer uses circuit input or one that produces no output. But the same phenotypes suffered badly if they needed to recover from that stage in the non-specialist environment, and generally failed to reconfigure.

The runs for non-specialist phenotypes show a slightly reduced fitness in their environment compared to specialists. This would seem to back the hypothesis given in §5, however it should be borne in mind that reconfiguring to manage seven frequency steps is more difficult than reconfiguring for three, and the added difficulty may have led to lower fitnesses. This conclusion has some backing from test results for non-specialists in specialist environments. In all cases, the non-specialists performed well. Not as well as specialist phenotypes, but their scores were higher than the scores they achieved in the environment they evolved in, leading us to suspect that the specialist environments were easier. Examination of the reconfiguration patterns show that the non-specialists were able to go back and forth using the same good circuits in both forms of specialist environment, demonstrating robust homoeostasis.

7 Conclusions and future work

Our system has demonstrated the potential of using a feedback mechanism as a means of reconfiguring solutions specified by an evolved genome representation. The representation is capable of maintaining multiple solutions and a large degree of redundancy. This redundancy has the potential to be expressed if conditions change, something which could be valuable where evolution is carried out *in situ* (see Stoica et al in [13]). The downsides to the technique are that the total number of potential gene expressions quickly rises as genome length increases. In such cases, it may be impossible to predict what solutions may be expressed if the conditions used to trigger reconfiguration change unexpectedly.

Biological control through stages of development relies on gene expression triggered by the context of transcription factors present in the nucleus of the cell at that moment in time. Homoeostatic control relies on gene expression controlled in a similar fashion, but with the option of reversibility. In our experiment, we wanted to see if our system could devise mechanisms of control of either type. The specialist phenotypes show in many cases the 'blindness' of their future proofing after evolutionary runs have left them stranded up specialist peaks of perfection. Non-specialists cannot afford the luxury of such high fitnesses, as it means they won't remain adaptable in the face of unexpected changes in the environment.

Our next plans are to take our system and investigate the effects of scaling upwards. Long genomes, with many thousands of potential solutions, require an equally wide set of reconfiguration triggers to allow phenotypes to explore functional search spaces. We want to understand the role of developmental stages in these large evolutionary searches, and by scaling up see whether our system can evolve adaptability in more complex environments.

References

1. Whitley, D.: An overview of evolutionary algorithms: practical issues and common pitfalls. Information and Software Technology **43**(14) (2001) 817–831
2. Coley, D.: An Introduction to Genetic Algorithms for Scientists and Engineers. World Scientific Publishing (May 1999)
3. Coello, C., Lamont, G., van Veldhuizen, D.: Evolutionary Algorithms for Solving Multi-objective Problems . Springer-Verlag (30 Nov 2006)
4. Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J., Lanza, G.: Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Kluwer Academic Publishers (2003)
5. Streeter, M., Keane, M., Koza, J.: Routine human-competitive automatic synthesis using genetic programming of both the topology and sizing for five post-2000 patented analog and mixed analog-digital circuits. In: 2003 Southwest Symposium on Mixed-Signal Design. IEEE Circuits and Systems Society (2003) 5–10
6. Banzhaf, W., et al: Guidelines: From artificial evolution to computational evolution: a research agenda. Nature Reviews Genetics **7**(9) (September 2006) 729–735
7. Kumar, S., Bentley, P., eds.: On Growth, Form and Computers. Elsevier (2003)
8. Carroll, S.: Endless Forms Most Beautiful: The New Science of Evo Devo and the Making of the Animal Kingdom. Weidenfeld & Nicolson (2006)
9. Thompson, A.: Silicon evolution. In: Genetic Programming 1996: Proceedings of the First Annual Conference, MIT Press (1996) 444–452
10. Thompson, A.: An Evolved Circuit, Intrinsic in Silicon, Entwined with Physics. In: Proc. 1st Int. Conf. on Evolvable Systems (ICES'96), Springer-Verlag (1997) 390–405
11. Harding, S., Miller, J.: Evolution in materio: Initial experiments with liquid crystal. In: Evolvable Hardware, IEEE Computer Society (2004) 298–
12. Miller, J.F., Downing, K.: Evolution in materio: Looking beyond the silicon box. [13] 167–176
13. Proc. of NASA/DoD Conference on Evolvable Hardware, IEEE Computer Society (2002)
14. Koza, J.R., Jones, L., Keane, M., Streeter, M.: Towards industrial strength automated design of analog electrical circuits by means of genetic programming. In: Genetic Programming Theory and Practice II. Kluwer (2004) 121–138
15. Mattiussi, C.: Evolutionary Synthesis of Analog Networks. PhD thesis, EPFL, Lausanne (2005)
16. Mattiussi, C., Marbach, D., Dürr, P., Floreano, D.: The Age of Analog Networks. AI Magazine (2007) (to appear).
17. Wolpert, L.: Relationships Between Development And Evolution. [7] chapter 2 47–62
18. Clegg, K., Stepney, S., Clarke, T.: Using feedback to regulate gene expression in a developmental control architecture. In Lipson, H., ed.: GECCO, ACM (2007) 966–973
19. Clegg, K., Stepney, S., Clarke, T.: Evolutionary Search Applied to Reconfigurable Analogue Control. In: Field-Programmable Logic and Applications: FPL07, Amsterdam, IEEE Press (2007)
20. Miller, J.F., Thomson, P.: Cartesian Genetic Programming. In: Genetic Programming, Proceedings of EuroGP 2000. Volume 1802 of LNCS., Springer (2000) 121–132