Designing a Morphogenetic System for Evolvable Hardware

Justin Lee and Joaquin Sitte

Smart Devices Laboratory
Faculty of Information Technology
Queensland University of Technology
GPO Box 2434, Brisbane, Qld 4001, Australia

Abstract. Traditional approaches to evolvable hardware (EHW), using a direct encoding, have not scaled well with increases in problem complexity. To overcome this there have been moves towards encoding a growth process, which however have not shown a great deal of success to date. In this paper we present the design of a morphogenetic EHW model that has taken the salient features of biological processes and structures to produce an evolutionary and growth model that consistently outperforms a traditional EHW approach using a direct encoding, and scales well to larger, more complex, problems.

1 Introduction

Evolvable hardware (EHW) uses simulated evolution to evolve circuits which are then evaluated for their fitness in producing the desired behaviour as required for solving a particular problem. EHW is generally implemented on reconfigurable hardware, such as field programmable gate arrays (FPGAs), which consist of a lattice of configurable logic blocks (CLBs), typically consisting of some arrangement of basic logic gates, multiplexors, and flip-flops, that can be configured to perform various digital logic functions. The functionality of the CLBs and the connections between CLBs can be configured by downloading a bitstream to the device to produce the desired circuit. FPGAs allow evolving solutions to be tested in situ, which is well suited to embedded applications such as robot controllers and image processing.

While evolvable hardware has proven to be successful in the evolution of small novel circuits, it has been limited in its applicability to complex designs, largely due the use of direct encodings in which the chromosome directly represents the device's configuration. A practical approach to solving this problem for specific application domains has been function-level evolution, involving the use of higher-level primitives such as addition, subtraction, sine, etc. (see [1–3] as examples). Although this scales the ability of EHW to solve more complex problems, it comes at the price of higher gate counts, designer bias and loss of potential novelty in solutions, thus countering some of the original motivations for EHW.

A separation between genotype (the chromosome) and phenotype (the generated circuit), and a way of generating the phenotype from the genotype (a growth process), is the approach taken by nature to evolve complex organisms, and has increasingly been seen as a means of scaling EHW to more complex problems without losing its ability to generate novelty. By encoding a growth process, known as morphogenesis, rather than an explicit configuration, the complexity is moved from the genotype to the genotype-phenotype mapping. Although there have been some successes with using morphogenetic approaches in generating neural networks [4–6] and tessallating patterns [7], there has been little success in EHW, and furthermore, existing approaches have not shown that morphogenesis does aid in scaling evolution to more complex problems.

We have undertaken an in-depth examination of biological development and by extracting the features from this that we judged to be useful and applicable to development within the constraints of EHW, we were able to come up with a complete bio-inspired morphogenetic EHW system for the Xilinx Virtex series of FPGAs, that includes biologically inspired genetic operators, chromosome structure and genetic code, along with cellular processes driven by gene expression and simple inter-cell signalling. Details of the design and design decisions are given in section two. Section three presents the results of using our morphogenetic system and compares this with a traditional direct encoding scheme, and in section four we conclude.

2 Design of a Cell-Based Morphogenetic EHW System

There are several issues that need to be resolved in the design of a developmental system for EHW. What abstractions can be usefully adapted from biology to EHW; what processes and structures should be emulated in a simulated cell; how to map from a cellular model to the FPGA hardware and at what level of abstraction should the hardware be manipulated; and, what genetic model and encoding should be used. These issues are dealt with in the following sections.

2.1 Mapping to FPGA

The level of abstraction for evolution and development to manipulate the FPGA resources can range from directly manipulating the bitstream, or the individual configurable components, to manipulating higher level constructs, as in functional EHW or with a hardware description language such as VHDL. We chose to manipulate the FPGA at the logic-gate level, using the Java JBits API provided by Xilinx for the Virtex [8], so as to avoid too much designer bias and to allow evolution freedom to explore novelty.

There is a spectrum of approaches as to how to map from a cellular model to the underlying hardware of the FPGA. On one extreme it may be a totally simulated cellular model, with no correspondance between the components and processes of development and the underlying hardware, with only the result of development being implemented on the FPGA. To the other extreme where all

aspects of development, such as proteins, signal pathways, etc, correspond to actual physical resources on the FPGA. After an in-depth look at both the Virtex architecture (see [9] for details) and biological developmental processes we decided on a model in which the developmental process is closely tied to the FPGA structure. Rather than trying to evolve or design simulated developmental mechanisms and structures, we directly use the underlying FPGA structure for implementing much of the developmental process. For example, rather than designing special signaling proteins and signal receptors, we let the FPGA routing resources act as signals to the connecting CLBs (see [10] for more details). This approach counters some of the difficulties involved in having simulated developmental processes distinct from the underlying medium. Problems such as the computational expense and arbitrariness of various aspects of development, such as rates of diffusion of signal proteins, determining which proteins are signals, what the extent of signal spread is, and matching them to receptors, which themselves need to be designed somewhat arbitrarily or evolved.

Cells too are mapped directly to the underlying FPGA architecture, so that each cell may correspond to either a CLB slice or logic element, according to the user's specification. The decision to map to slice or logic element, rather than CLBs, was made due to the functional independence of these elements in the Virtex: logic elements are functionally independent for most functions, while slices may either utilise two independent 4-input function generators or combine these into a single 5-input function generator.

2.2 Biological Developmental Processes

Biologically speaking, development is the process by which a multicellular organism is formed from an initial single cell. Starting from a single cell, a simple embryo is formed comprised of a few cell types organised in a crude pattern, which is gradually refined to generate a complex organism comprised of many cell types organised in a detailed manner. This model of the development process is known as epigenesis, and is comprised of five major overlapping processes: growth, cell division, differentiation, pattern formation and morphogenesis [11]. Growth brings an increase in the size of the organism through changes in cell size, or cell division, or depositing of materials (such as bone) into the extracellular matrix; cell division increases the number of cells, and may occur in conjuction with, or independently from the growth phase; differentiation is responsible for producing the cells with specialised structures and functionality through changes to cells' patterns of gene expression; pattern formation organises and positions cells to generate an initially rough body plan which is then further refined by morphogenesis, which is responsible for coordinating the dynamic behaviours of cells in the developing embryo to collectively generate tissues and organs of differing shapes and structures.

Although growth and cell division are essential in biological organisms, their applicability to our EHW model is limited by the fixed mapping from cells to hardware that we have chosen. The underlying FPGA hardware has a fixed structure, with FPGA logic cells (CLBs) being fixed in both shape and in their

physical relationship to each other (having a regular matrix pattern), with only their connectivity and function being variable. Although pattern formation is not directly applicable for the same reasons, there are some abstractions from pattern formation that are still relevant, such as axis specification and polarisation. In the biological process of pattern formation, one of the first tasks undertaken in the embryo is to determine the principal body axis, and polarisation. This may be determined by the asymmetric distribution of maternal gene products in the egg, or require a physical cue from the environment. For performing morphogenesis in EHW, axis specification and polarisation may also be inportant, and in our case is provided by axis specific simulated cytoplasmic determinant molecules preplaced at run-time. Morphogens are chemical gradients that produce graded responses from cells according to concentration. These are the primary biological mechanism for determining location along an embryonic axis. In our EHW system, simulated morphogens are also used for this purpose, to give information as to a cell's position relative to the input and output cells on the CLB matrix.

The processes which are most useful to our model are differentiation and morphogenesis, through which cells coordinate their behaviour. These two processes are closely entwined, and both have gene expression central to their functioning.

Gene Expression The whole developmental process is largely driven and controlled by the mechanics of gene expression. This is the process by which proteins signal cellular state to activate genes, which in turn are able to effect changes in cell state by the production of further proteins which may be used as signals or cellular building blocks. This provides a view of the cell as a set of parallel processing elements (genes) controlled by the interactions between their programs encoded in the chromosome and their environment as represented by proteins detectable by the cell.

There are two particular types of protein that have important roles in development, these being transcription factors (TFs) and components of signaling pathways. Transcription factors control gene expression by binding at sites on the regulatory regions of a gene, hence playing a major role in the coordination of developmental processes; whereas signaling pathways are necessary for cells to be able to perceive external signals [11]. The mechanics of signaling pathways are quite complex, and not necessary for EHW. What is necessary is that signals from other cells can be detected and effect the expression of genes within the receiving cell. In our model all proteins are treated as transcription factors, so that choosing which effect gene expression can be decided by evolution (via elements that can bind at binding sites) rather than arbitrarily by the designer, however simulated TF molecules that are only used for controlling gene expression are also provided and correspond to non-coding messenger RNA in higher-level organisms, which are able to encode higher-level architectural plans [12].

Control of gene expression can take place at any of the intermediate stages of transcription, RNA processing, mRNA transport, mRNA degradation, and protein activity. Transcription of DNA, whereby the coding region of a gene

is transcribed to an RNA molecule prior to translation into a protein, is the first level of regulation of gene expression and hence the developmental process, and is generally the most important level of control in gene expression [13, 11]. As gene regulation at the transcription level appears to be the most important level of gene regulation, and for reasons of simplicity and limiting computational expense, we chose to regulate gene expression solely at this level in our system. Furthermore, the results achieved by Reil [14] who used a gene expression model with transcriptional regulation, demonstrated that gene regulation using a simple model is able to produce many of the properties exhibited in nature.

Cell Differentiation Generally speaking, cells all contain the same genetic information, however, their specialised structures and functionality differ according to the proteins present within the cell, and this is determined by which genes are expressed. Differentiation is the process by which different patterns of gene expression are activated and maintained to generate cells of different types. Which genes, and hence proteins, are expressed differs between cells according to what cytoplasmic determinants are inherited at cell division, and what extracellular signals are received. Cytoplasmic determinants are molecules, such as transcription factors, that bind to the regulatory regions of genes and help to determine a cell's developmental fate (i.e. pattern of gene expression that causes the cell to differentiate in a particular manner). Although cell division is not applicable to our developmental model, the use of pre-placed cytoplasmic determinants to differentiate cells, at specially designated IO cells for example, may be useful.

Induction, whereby a signal received from another cell is able to affect the receiving cell's developmental fate, is used to control differentiation and pattern formation in development. An inductive signal may be used to instruct a cell to choose one fate over others, or be required to allow a cell already committed to a particular developmental pathway to continue towards differentiation. Inductive signals may occur over various ranges and may produce a single standard response in the responding cell, or a graded response dependent on signal concentration, in which case it is called a morphogen [11]. Induction and other forms of signaling (both from within and without the cell) can be readily applied to EHW with fixed cell structures, and along with gene expression, are probably the most important mechanisms of developmental biology in their applicability to EHW.

Morphogenesis Morphogenesis is the process by which complex structures, such as tissues and organs, are generated through the utilisation of cell behaviours. Cells are able to produce many individual and collective behaviours, such as changes of shape and size, cell fusion and cell death, adherence and dispersion, movements relative to each other, differential rates of cell proliferation, and cell-extracellular matrix interactions [11].

Obviously many of these behaviours are not directly applicable to developmental processes in EHW where there is a fixed mapping between cells and the underlying hardware structure. Cell behaviours here are limited to changes in

connectivity (routing) and function. Of the biological behaviours listed above, only cell-cell and cell-extracellular matrix interactions are applicable to our EHW system. Cell death, was not used in our system, but would be simple to implement, by disabling connections to and from the dead cell, and could be used to isolate faulty regions of the underlying hardware. The notion of an extracellular matrix, a network of macromolecules secreted by cells into their local environment, also has some relevance to our system, for interactions between cells and the matrix inducing differentiation. The extracellular matrix could be used to correspond to the inter-CLB cell routing resources, specifically the programmable interconnection points (PIPs) used to connect lines from other CLBs to lines that can connect to the local CLB's inputs or outputs. Cell-cell interactions, in contrast, deal only with directly connectable lines between CLBs. In the current version of our EHW system, only directly connectable single-length lines (between neighbouring CLBs) are used, providing cell-cell interactions, but ruling out cell-extracellular matrix interactions.

2.3 Cell Model

Biological cells contain structures such as a cell wall to stop swelling or shrinking and a cell membrane that allows the passing of small molecules, and are filled with a substance known as the cytoplasm. These and other cell structures do not need to be explicitly represented in simulated cells with a fixed size and mapping to the underlying FPGA hardware. This also applies to the metabolic processes required for cell maintenance in biological systems. We have chosen a simple cell model loosely based on prokaryote (bacterial) cells, containing a single chromosome, proteins and a number of RNA polymerase enzymes (currently based on the number of genes in the chromosome). Ribosome, which is required to translate messenger RNA to proteins, does not need explicit representation, as the transcription-level gene regulation model we use only requires the simulation of the RNA polymerase enzyme responsible for transcription. Translation is treated as an indivisible part of the transcription process, which, although not biologically correct, meets the functional requirements of our model.

There are three kinds of proteins detectable within the cell, two of which are present in the cell, these being the simulated transcription factors and the non-simulated FPGA gate-level logic settings, and the other is the receiving end of a signaling pathway that corresponds to a shared FPGA routing resource. All proteins that correspond to underlying FPGA gate-level settings, have one protein per logic resource present in the cell (signaling pathways are present in the originating cell), and these are present for the entire duration of the cell's life. Simulated transcription factors, however, don't need to be unique within the cell, nor does every possible TF need to be present in the cell, and TFs have a lifespan.

2.4 Genetic Model

Chromosome Model In the design of the encoding of the chromosome and genes, one of the first considerations was allowing a variable number of genes and preconditions for their expression. Requiring a preset number of genes would introduce unnecessary designer bias and constrain evolution's ability to find optimal solutions.

Another important factor that was taken into consideration is the importance of neutral mutations and junk DNA. When a mutation to the genotype makes no change to the resulting phenotype, this is known as a neutral, or silent, mutation. This occurs frequently in nature due to the redundancy of the genetic code (the mapping from the triplet sequences of bases of RNA, known as codons, to amino acids), where a mutation (or "wobble") at the third position in a codon often codes for the same amino acid [15]. Neutral mutations are important as they allow movements in genotype space with no changes in fitness, which gives the population the ability to take mutations that are not immediately beneficial. Thus the population is able to drift along neutral ridges [16], rather than sacrificing its current fitness, which may significantly aid the evolutionary search. Instead of becoming trapped in sub-optimal regions of the landscape a population is able to continue moving through genotype space in search of areas that allow further increases in fitness. Neutral mutations have been shown to significantly aid evolutionary search for evolving telecommunications networks [17] and evolvable hardware [16, 18].

Junk DNA is used to denote sections of the chromosome that have no function, and so can collect mutations, but may later be activated. This may happen, for example, through gene duplication where a mutation on a gene's promoter site deactivates that gene (acting as a gene switch), allowing neutral mutations to take place on the duplicate, which may later be reactivated [19]. See also the work of Burke et al. [20] on the exploitation of variable length genomes for utilising junk DNA, and Harvey and Thompson's work which utilised 'junk' sections of the genome in EHW [16].

To exploit these factors we decided on a variable length chromosome and a base-4 encoding along with a codon-based genetic code. A variable length chromosome allows evolution to decide how many genes are required for the problem at hand, while also providing space for junk DNA. A base-4 chromosome was chosen as it allows us to constrain the mutation operators and gives more redundancy for neutral mutations. A codon-based genetic code was decided on to facilitate neutral mutations on genes' coding region: most single base mutations will result in either no change or a change to a related gene product, especially for mutations in the third base of a codon.

Gene Model Genes are bounded by initiation and terminator sites. The initiation sites contain zero or more regulator regions and a promoter region. RNA polymerase recognises promoter regions as starting points for gene transcription, which continues until the terminator site is reached. Regulatory elements control where and when the gene is transcribed, and the rate of transcription. With this

in mind a gene structure loosely based on that of prokaryotes (in particular the operon model) was decided on, giving a gene structure as shown in Figure 1.

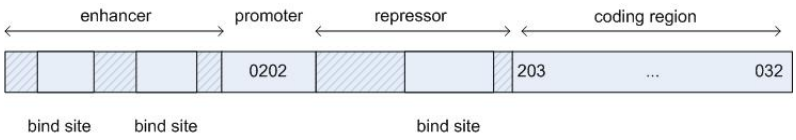

Fig. 1. Gene Structure

Enhancers and repressors are determined by their location on the gene relative to the promoter. Enhancers are located upstream of promoter, where they act to attract the polymerase enzyme for gene transcription, while repressors are located between the promoter and gene coding region, thus blocking polymerase from transcription. Transcription factors (either simulated or proteins that correspond to the underlying FPGA hardware resources) bind to repressors and enhancers to effect the activation of the associated gene. Within regulatory regions (enhancers and repressors), there are bind sites to which these can bind. These are identified by special signature sequences, allowing a variable number of bind sites per regulator. FPGA resources are able to be bound to several bind sites concurrently, but TFs are only able to bind to a single bind site, and remain attached for their remaining lifespan.

The gene coding region encodes for FPGA gate-level logic and simulated molecules, and allows multiple of these to be encoded per gene. There are, however, no introns and exons, only a sequence of codons which encode gene products. A start codon, analogous to the AUG start codon in nature, is used to indicate where to start transcription, and a stop codon (eg UGA, UAA, UAG) is used to indicate where transcription terminates. Gene products are decoded to an intermediate format (analogous to a chain of amino acids) by mapping each resource type (such as 'LUT') and attribute (eg LUT 'F'), to a specific codon, as given in the genetic code, and then by further decoding that resource's settings (eg a 16 bit LUT configuration) from the following codons according to the resource's own code. This format is then further decoded to produce JBits class constants and settings values for manipulating the FPGA configuration bitstream. Our genetic code was specifically designed for use with EHW systems where the number of resources to be set per CLB is not predetermined, such as when encoding a growth process.

Binding of FPGA resources to bind sites on genes' regulatory regions is done using a related coding scheme, with the only difference being in the first codon of the code, which differs slightly to allow the differentiation between local-to-cell FPGA resources, and connecting resources which may have the originating cell's state queried, as required for implementing signalling pathways.

Genetic Operators Evolution cannot occur without genetic operators to search the genotype space. The most commonly used operators in evolutionary computation are crossover and mutation. Although these were inspired by biological counterparts, biological crossover in particular has little resemblance to its simulated operator. In nature crossover requires two DNA molecules with a large region of homology (nearly identical sequence), usually hundreds of base pairs long, so that the exchange between two chromosomes is usually conservative [21]. Taking inspiration from this, we have implemented a homologous crossover operator that uses a variant of the longest common substring, implemented using Ukkonen's algorithm for constructing suffix trees in linear time [22], but with a random common substring being chosen and biased towards longer matches. 1-point crossover is then performed at the boundary of the randomly chosen subsequence.

Mutation in our system is also biologically inspired. Mutations of a single base may involve a mutation of an existing base to another character in the DNA alphabet, and may be of two kinds: transversions (A-T or G-C) and transitions (A-G or T-C). Many of these mutations will have no effect on the encoded protein due to the redundancy of the genetic code, and thus aid in evolutionary search. Other mutations may involve the insertion or deletion of bases in a sequence, which may cause a frame shift in the codons downstream, and will have a serious effect on the encoded protein which is generally deleterious [21]. Another type of mutation involves the reversal of a section of the chromosome and is known as inversion. We have provided analogs of each of these kinds of mutation.

3 Experiments

This set of experiments is aimed at testing whether our morphogenetic system can successfully generate complex circuit structures and to compare its performance and scalability against a direct encoding. We ran two sets of experiments, the first involved routing on a 5x5 CLB matrix (containing 100 cells) from an input in the center of the west edge of the matrix to an output at the center of the east edge of the matrix. Evolution must also connect the input and output cells to dedicated routing CLBs on the outside (of the evolvable region) neighbour. For the second set of experiments we increased the size of the CLB matrix to 8x8 (containing 256 cells), and the number of inputs and outputs to 4 each. Inputs are placed in the center of the West edge of the CLB matrix, 2 input locations per CLB, while outputs are spread evenly across the East edge of the CLB matrix, requiring evolution to learn not just how to connect horizontally across the matrix, but also how to spread vertically from the middle outwards.

To route from inputs to outputs would generally be trivial, and so we have severely constrained the routing lines available. Each cell is mapped to a logic element, giving 4 cells to a single CLB. Each cell is then limited to a slimmed down subset of resources, with only one input used per LUT, giving 4 possible LUT functions (output 0 or 1, pass or invert signal). Each cell is able to drive the input of a LUT in 3 or 4 of the neighbouring CLBs. The set of lines available

to each cell were chosen such that it is not possible to directly route horizontally from the West to East edges of a CLB matrix, and it is also necessary for lines to be routed through each of the 4 distinct cell (logic element) types. Fitness, in both experiments, was based on how much progress was made in routing a signal, possibly inverted, from the inputs to the outputs, noting that we don't care what the relationship between the different inputs and outputs are, only that all inputs are connected and one or more of these drives the outputs. See [10] for more details on the FPGA resources allocated to each cell type and the algorithm used for calculating fitness.

For each set of experiments twenty evolutionary runs were done with a population size of 100 and using a steady state genetic algorithm with tournament selection without replacement. The crossover rate was set at 80%, mutation at 2%, inversion at 5%, and for the variable length chromosomes used with the morphogenetic approach, a base insert/delete rate of 0.1% was used with 50-50 chance of insertion or deletion. Each evolutionary run was continued until a solution with 100% fitness was found or until a sufficient number of generations had passed without an improvement in the maximum fitness attained (1000 generations for the first set of experiments, and 1500 with a minimum of 5000 generations for the second). For the morphogenesis approach, growth is done for a minimum of 30 steps, with fitness evaluated at each step, and growth continued if the maximum phenotype fitness for this genotype increased in the last 15 (minimum growth steps/2) growth steps, or if phenotype fitness is increasing. The genotype's fitness is given by the maximum phenotype fitness achieved during growth. Note that TFs and morphogens were not used in these experiments.

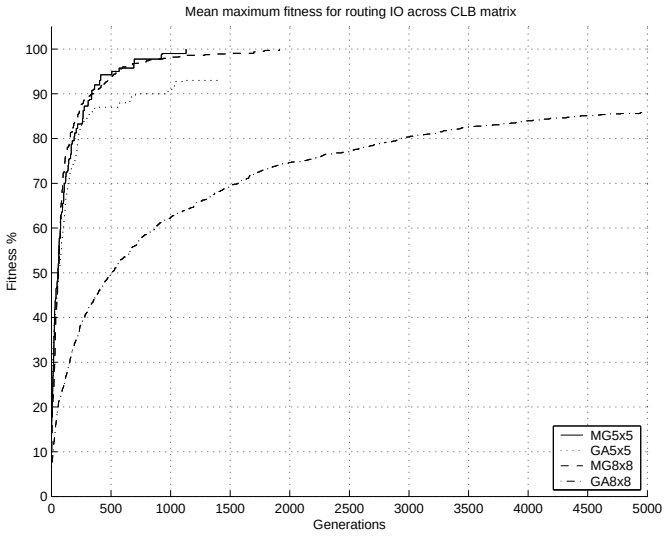

Fig. 2. Mean maximum fitness for routing IO across CLB matrix

In the first set of experiments the direct encoding approach was able to find a 100% solution in 13 out of the 20 runs, with the average number of generations required for successful runs being 531.0769 with a standard deviation (SD) of 340.5768. The morphogenetic approach was able to find a 100% solution every time, and took an average of 458.5 generations (SD=283.9556), 36.95 growth steps, and had on average 9.9 genes and a chromosome length of 5690.35 bases. In the second set of experiments the morphogenetic approach was again able to find a 100% solution on each run, taking an average of 1001.7 generations (SD=510.5647), 49.95 growth steps, and had 5.65 genes and chromosome length of 3461.4 bases. The direct encoding approach, however, was unable to find any 100% solution, with maximum fitness values reaching a mean of 86.6406% (SD=3.0930%), and taking on average 4647.1 generations (SD=1756.9). The highest fitness achieved by the direct encoding approach was 93.75% which occurred at generation 9954. This run was continued up to 35,000 generations and reached a maximum of 96.875% at generation 16,302. Figure 2 show the mean maximum fitness over all runs for both approaches on the two experiments (up to generation 5000).

From Figure 2 it is evident that the morphogenetic approach (denoted by MG) not only outperforms the direct encoding approach (denoted by GA), but also scales very well, with the more complex problem (MG8x8) keeping pace with the simpler problem (MG5x5) up until the last few fitness percentage points where there is a lag of around 800 generations until it catches up. This is in complete contrast to the direct encoding approach, where it took 5000 generations to reach the same point that took 500 generations on the simpler problem.

4 Conclusion

In this paper, we have introduced our morphogenetic system for evolvable hardware and shown how we chose its key characteristics based on an in-depth investigation of biological developmental and genetic processes. By closely coupling the gate-level state of the underlying hardware with a simple, yet flexible, gene expression model to drive development we have avoided introducing too many assumptions and overheads, while allowing a great deal of redundancy for neutral pathways through evolutionary space, and have come up with a system that not only outperforms a standard direct encoding approach to EHW, but scales well to increases in problem complexity.

References

1. Higuchi, T., Murakawa, M., Iwata, M., Kajitani, I., Liu, W., Salami, M.: Evolvable hardware at function level. In: IEEE International Conference on Evolutionary Computation. (1997) 187–192
2. Clark, G.R.: A novel function-level EHW architecture within modern FPGAs. In: Proceedings of the 1999 Congress on Evolutionary Computation (CEC99). Volume 2. (1999) 830–833

3. Kalganova, T.: An extrinsic function-level evolvable hardware approach. In: Proceedings of the Third European Conference on Genetic Programming (EUROGP2000), Lecture Notes in Computer Science. Volume 1802., Edinburg, UK, Springer-Verlag (2000) 60–75
4. Jakobi, N.: Harnessing morphogenesis. Technical Report CSRP 423, School of Cognitive and Computer Science, University of Sussex, Sussex (1995)
5. Eggenberger, P.: Cell interactions as a control tool of developmental processes for evolutionary robotics. In: Proceedings of SAB '96. (1996) 440–448
6. Roggen, D., Floreano, D., Mattiussi, C.: A morphogenetic evolutionary system: Phylogenesis of the poetic circuit. In Tyrrell, A.M., Haddow, P.C., Torresen, J., eds.: Proceedings of the 5th International Conference on Evolvable Systems: From Biology to Hardware ICES 2003. Volume 2606 of Lecture Notes in Computer Science., Trondheim, Norway, Springer (2003) 153–164
7. Bentley, P., Kumar, S.: Three ways to grow designs: A comparison of evolved embryogenies for a design problem. In: Proceedings of the Genetic and Evolutionary Conference (GECCO '99). (1999) 35–43
8. Guccione, S., Levi, D., Sundararajan, P.: Jbits: Java based interface for reconfigurable computing. In: Second Annual Military and Aerospace Applications of Programmable Devices and Technologies Conference (MAPLD), Laurel, MD (1999)
9. Xilinx Inc.: Virtex 2.5 V Field Programmable Gate Arrays: Product Specification, DS003 (V2.5). http://direct.xilinx.com/bvdocs/publications/ds003.pdf (2001)
10. Lee, J., Sitte, J.: A gate-level model for morphogenetic evolvable hardware. In: Proceedings of the 2004 IEEE International Conference on Field-Programmable Technology (FPT'04), Brisbane, Australia (2004)
11. Twyman, R.: Instant Notes in Developmental Biology. BIOS Scientific Publishers limited, Oxford (2001)
12. Mattick, J.S.: Non-coding RNAs: the architects of eukaryotic complexity. EMBO reports **2** (2001) 986–991
13. Reil, T.: Models of gene regulation - a review. In Maley, C., Boudreau, E., eds.: Artificial Life 7 Workshop Proceedings, MIT Press (2000) 107–113
14. Reil, T.: Dynamics of gene expression in an artificail genome - implications for biological and artificial ontogeny. In Floreano, D., Mondada, F., Nicoud, J., eds.: Proceedings of the 5th European Conference on Artificial Life, Springer Verlag (1999) 457–466
15. Crick, F.: Codon-anticodon pairing; the wobble hypothesis. Journal of Molecular Biology **19** (1966) 548–555
16. Harvey, I., Thompson, A.: Through the labyrinth evolution finds a way: A silicon ridge. In Higuchi, T., ed.: Proceedings of the First International Conference on Evolvable Systems: From Biology to Hardware, Springer-Verlag (1996) 406–422
17. Shipman, R., Shakleton, M., Harvey, I.: The use of neutral genotype-phenotype mappings for improved evolutionary search. BT Technology Journal **18** (2000) 103–111
18. Thompson, A.: Notes on design through artificial evolution. In Parmee, I.C., ed.: Adaptive Computing in Design and Manufacture V, London, Springer-Verlag (2002) 17–26
19. Ohno, S.: Evolution by Gene Duplication. Springer Verlag, Berlin (1970)
20. Burke, D.S., Jong, D., A., K., Grefenstette, J.J., Ramsey, C.L., Wu, A.S.: Putting more genetics into genetic algorithms. Evolutionary Computation **6** (1998) 387–410
21. Winter, P., Hickey, G., Fletcher, H.: Instant Notes in Genetics. 2nd edn. BIOS Scientific Publishers limited, Oxford (2002)
22. Ukkonen, E.: On-line construction of suffix trees. Algorithmica **14** (1995) 249–260