A Model for Intrinsic Artificial Development Featuring Structural Feedback and Emergent Growth

Martin A. Trefzer, Tüze Kuyucu, Julian F. Miller and Andy M. Tyrrell

***Abstract*—A model for intrinsic artificial development is introduced in this paper. The proposed model features a novel mechanism where growth emerges, rather than being triggered by a single action. Different types of cell signalling ensure that breaking symmetries is rather the norm than an exception, and gene activity is regulated on two layers: first, by the proteins that are produced by the gene regulatory network (GRN). Second, through structural feedback by second messenger molecules, which are not directly produced through gene expression, but are produced by sensor proteins, which take the cell's structure into account. The latter feedback mechanism is a novel approach, intended to enable adaptivity and environment coupling in real-world applications. The model is implemented in hardware, and is designed to run autonomously in resource limited embedded systems. Initial experiments are carried out to measure long-term stability, dynamics, adaptivity and scalability of the new approach. Furthermore the ability of the GRN to produce patterns of different symmetries is examined.**

I. Introduction

Since the invention of the field effect transistor in 1971, which rendered the first microprocessor possible, the complexity of electronic circuits has grown exponentially in terms of number of transistors and gates. This is the reason why it has become possible to build larger and larger electronic systems that comprise a great number of modules, which are complex themselves, and still operate in a correct and reliable fashion. However, the design of such increasingly large systems, which become less and less understandable by one designer alone, demands more and more sophisticated, automated design tools to master their complexity and hierarchies.

Given the complexity of such systems, automated electronic circuits and systems design has always been a challenging target application for evolutionary computation (EC). Unfortunately, unlike designed systems, the size and complexity of successfully evolved circuits has not grown to the same extent. To date, the largest circuits evolved from scratch are the prime number predictor (400 multiplexers) published in [24] and the 6-bit multiplier (186 gates) in [22]. The complexity of a circuit is assessed by expressing the number of its components in terms of the number of primitive gates (AND, NAND, OR, NOR), one multiplexer is thereby counted as four gates. Examples where the building blocks used a high degree of domain knowledge were discarded, e.g. when XORs are used in evolving parity.

Martin A. Trefzer, Tüze Kuyucu, Julian F. Miller and Andy M. Tyrrell are with the Department of Electronics, Intelligent Systems Group, University of York. {mt540, tk519, jfm7, amt}@ohm.york.ac.uk, http://www.bioinspired.com.

This work is part of a project that is funded by EPSRC - EP/E028381/1.

Previous research within the field makes it evident that competitive and scalable solutions can only be achieved by partitioning a given task or when indirect or generative genotype to phenotype mappings are applied. Multiple chromosomes are introduced to account for multiple outputs in cartesian genetic programming (CGP) in [25]. Other approaches are (automatically) defined functions in genetic programming and building blocks [12], [25], incremental evolution [22] and decomposition of the task [21]. Generative genotype to phenotype mappings are used, for instance, in lindenmayer systems [9], [10], [16] and cellular automata [19].

This work is based on yet another approach to indirect mappings, which is based on the principles of natural development [26]. Examples for this are random boolean networks (RBNs) and artificial genetic regulatory networks (GRNs) [2], [4], [6]. Developmetal systems usually feature regulatory and structuring proteins, which are produced by a set of genes, and in turn regulate which genes are expressed (actually produce proteins) in a dynamic fashion. Natural development forms patterns and shapes, where different regions (groups of cells) work together and specialise in required tasks to maintain and operate the whole organism. This process features–to a certain extent–many properties that are desirable to build complex artificial systems: scalability, robustness, adaptivity and self-repair. Not to mention the fact that description of complex structures and functionality is stored in a highly sophisticated and compressed manner within the genes. Thus, various research has been done to mimic these properties of nature in order to achieve the same properties in artificial (electronic) systems [1], [3], [7], [13], [17], [18], [20].

Although only relatively small numbers of cells are used and the organisms are usually matured for less than 50 developmental steps, the reported results show that artificial developmental systems have the potential to scale, self-repair and adapt. Thus, the work that has been undertaken encourages further investigation into how we can adapt the mechanisms of natural development to solve engineering problems. Numerous models of development have been implemented and are more or less successfully applied to different design problems. So far these models have been proven to be most successful in solving pattern formation tasks [3], [9], [10], [16], [18], [20]. However, it is neither yet clear which mechanisms of natural development should be included in artificial systems, nor is it obvious which ones can be left out without losing too much of development's potential.

A new artificial developmental system is introduced in this paper. In this case, the fundamental design goal is that the

978-1-4244-2959-2/09/$25.00 © 2009 IEEE

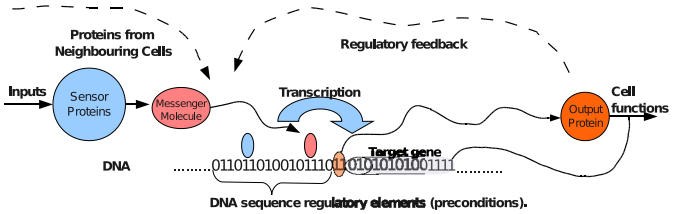

Fig. 1. The genes in a DNA sequence are activated when a sufficient amount of protein is produced by the GRN, which then causes transcription of the respective gene. In the example shown in the figure, the activity of the GRN results in the correct transcription factors (the promoter proteins) that bind to the sites of a gene sequence. This initiates the production of another protein, which can then affect the cell functionality using further information that is encoded in the gene. The produced protein also provides feedback to the GRN, which regulates the transcription of further genes, hence, dynamic gene regulation is achieved. Note that when any inhibitory condition is met, the gene is not expressed.

system has to be able to run efficiently in hardware whilst keeping it as close as possible to its biological inspiration. The developmental model is introduced in this paper and the decisions that have been made about which mechanisms from biology to adapt (and how) are discussed. As the ability to form and control patterns has been used to investigate the behaviour of development [3], [5], [9]–[11], [16], [20], [26] they are also ultimately chosen to demonstrate the behaviour of development in this paper. The floorplan of every electronic circuit is a pattern. However, in this work the long term behaviour of development is investigated for the first time; as one of our aims is to keep development running in the background to control and maintain the operation of a distributed, autonomous hardware system.

Basic pattern formation experiments with 2x2 cells are undertaken in order to measure long-term stability, dynamics and scalability of the proposed artificial developmental system. Furthermore, the ability of GRN to produce different patterns of different symmetries on a 6x6 cell tissue is investigated. Experiments are carried out intrinsically on the reconfigurable integrated system array (RISA) [8].

II. Model for Intrinsic Development

The main design considerations for the presented model for artificial development in hardware are: first, the fact that it actually runs in hardware, which imposes simplifications and the choice of suitable data types (boolean, integers). Second, to keep the selected mechanisms as close as possible to their biological counterparts within the boundaries of hardware. Third, as the structural genes will configure hardware resources in a useful fashion, regulatory feedback from these structures is desired. Fourth, a mechanism for interaction with the environment is proposed in order to achieve adaptivity.

A. Representation and Gene Regulation

The core of the proposed developmental model is represented by a GRN, as shown in figure 1. Genes are implemented as binary strings and they interact through proteins. The binary string that encodes the genes is also referred to as (artificial) deoxyribonucleic acid (DNA) and is evolved using a standard

TABLE I
The DNA is evolved using an evolutionary strategy (ES) with the parameters that are shown in the table.

EA parameter	Value
population size	7
no. of parents	2
evolutionary strategy	2+5
mutation rate	adaptive, co-evolved (0.5 . . . 10%)
maximum generations	5000

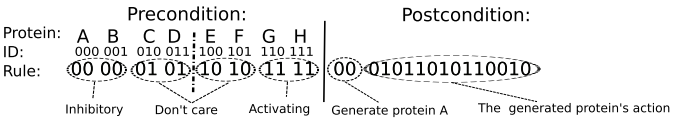

Fig. 2. An example gene formed of 32 bits is shown in the figure above. The first 16 bits are reserved for the preconditional part, which specifies the rules to activate the gene. There are eight proteins defined in this figure; the first four being structuring, Plasmodesmata (tunnelling), sensor and structuring proteins. The last four are second messenger molecules, which are not directly created by the GRN, but are produced by the sensor protein as a result of an interaction with the current cell structure. Each protein's required presence or absence is specified by a 2 bit number, that encodes 1 excitatory, 1 inhibitory and 2 don't care states. In the case of a don't care event, the presence or absence of a protein has no effect on the transcription of the particular gene. The second 16 bits of the gene are reserved for the postconditional part, which provides the ID of the protein produced as a 2 bit number, as only the proteins can be directly produced. The last 14 bits in the gene encode the action of the protein produced, i.e. if it's not a regulatory protein (further explained in table II). If it is a regulatory protein, the last 14 bits will be treated as junk.

evolutionary strategy (ES) (table I) with a relatively small population size, in order to fit it into the hardware. The EA parameters are based on values that are widely used in the field of evolvable hardware. The major difference to biology is that in this case genes are all of the same length and lined up to form the DNA, hence, are defined by their respective positions or addresses within the DNA. Natural genes however, are defined by certain upstream and downstream gene sequences (binding sites) that accept proteins to bind and transcribe their genetic code. The binding sites in natural DNA therefore allow for smooth binding, i.e. the probability that certain proteins (transcription factors) bind to the DNA is given by how well the binding sites of the protein matches the one of the DNA. There are examples of artificial GRNs where smooth binding is included [5], [11]. However, in the case of embedded hardware systems, the inclusion of this feature is extremely expensive in terms of memory requirement and computation effort. Thus, a hardware friendly, rule based model similar to the one introduced in [7] is chosen to implement gene regulation. The structure of one gene and the way in which protein concentration influences gene expression or suppression is described in figure 2.

In principle the system can work with an arbitrary number of proteins. However, currently the system is working with eight proteins: four primary proteins and four second messenger molecules. Gene regulation is described in figure 2. A description of the proteins and their roles is given in table II. Having four messenger molecules accounts for the fact that

TABLE II

THE CURRENT GRN WORKS WITH FOUR PRIMARY PROTEINS AND FOUR SECOND MESSENGERS. PRIMARY PROTEINS ARE DIRECTLY PRODUCED BY THE GRN, WHEREAS SECOND MESSENGER MOLECULES ARE ONLY PRODUCED AS A RESULT OF A 'MEASUREMENT' THAT IS PERFORMED BY THE PRIMARY SENSOR PROTEIN. THEIR ROLES IN DEVELOPMENT ARE DESCRIBED IN THE TABLE. PLEASE NOTE THAT ALL PROTEINS AND MESSENGER MOLECULES TAKE PART IN GENE REGULATION, IN ADDITION TO THEIR SPECIAL PURPOSE.

Protein/ Molecule	Role (apart from gene regulation)
Plasmodesma (primary)	Plasmodesma proteins provide a mechanism to form Plasmodesmata [15] in order to share their proteins with neighbour cells, and they initiate growth.
Structuring (primary)	Structuring proteins alter the part of the bit-string of the configurable target platform that is owned by the respective cell, based on the data that is encoded in the postconditional part of the gene. 6 out of the 14 postconditional bits determine an address within the 64 bit structure of the cell, two bits determine how many bits are written to the structure and up to four bits represent the actual data that is written to the structure. This way, the whole structure is built as a result of multiple structuring genes expressed.
Sensoring (primary)	Sensoring proteins provide a means to react to changes in the cell state. Secondary proteins (second messengers [26]) are produced, based on current cell function and possible rules that can be encoded in the genes.
Other (primary)	An additional regulatory protein that can be directly produced by the GRN.
Messenger (secondary)	Messenger molecules are produced as a result of the sensor protein being expressed and reacting according to the current cell state.

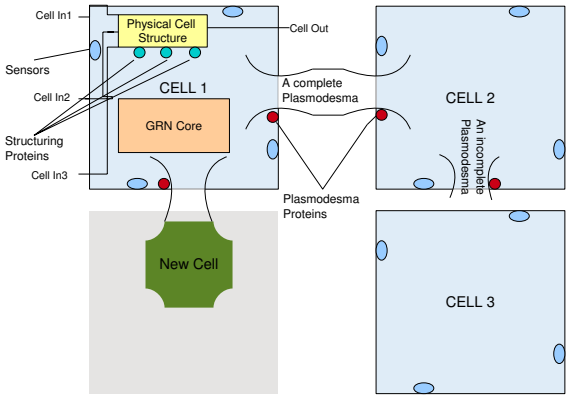

Fig. 3. In a multi-cellular environment using the four basic protein types a cell is able to: interact with its environment, multiply, structure itself, and form a complex multicellular organism. The basic functions of some proteins are demonstrated in this figure. Only cell 1 is drawn completely, certain components are omitted in other cells. In the example above, cells 1 and 2 both have active Plasmodesmata proteins, which cause the formation of a channel on both cells towards the other, creating a Plasmodesmata to allow free movement of proteins from one cell to other. Cells 1 and 2 both also have active Plasmodesmata proteins on their southern sides. Cell 1's southern neighbour is a dead cell, so the active Plasmodesmata protein initiates a growth process in that direction. However, cell 2's southern neighbour is an alive cell with no Plasmodesmata protein, thus cell 2 forms an unconnected channel on it's southern wall. The direction in which the Plasmodesmata proteins are active, is encoded in the postcondition of the gene, hence, there is one species of Plasmodesmata protein with four different behaviours. The four sensors drawn monitor the outside activity on four sides of each cell and produce different messenger molecules with changing environment. The structuring proteins are produced by the GRN to change the physical structure of the cell, which is connected to the physical inputs and outputs of the cell.

the cells can take on four different states. Depending on the cell state in each developmental step, the sensor protein can produce one of the four secondary molecules.

One type of protein at a time is produced per gene expressed. Whenever a protein is required to inhibit or express a gene, its level is decreased. In the presented experiments, production rate is set to 3% (0.5% for the Plasmodesmata protein) and consumption rate is set to 1.5% of the maximum protein level. Only when the protein level is above 50% of the maximum, the genes are taken into account to interpret the rules.

B. Cell Communication and Growth

Wolpert [26] describes three different types of cell signalling (communication): protein diffusion, direct contact of complementary proteins on the cell's surface, and gap junctions (Plasmodesmata). Two different types of cell signalling are realised in the proposed model. Protein diffusion has been implemented in a similar way as it takes place in physical systems, e.g. a drop of ink dissolving in water. Half of the cells protein is thereby distributed amongst its nearest neighbours in equal shares, i.e. each neighbour obtains $\frac{1}{8}$ of the cell's protein. Protein levels are credited and debited after the GRN has processed the next developmental step for all cells. Thus, the GRN always works with the original protein levels and the order of cell update should therefore not bias the course of development.

Diffusion is a signalling mechanism that helps maintaining symmetries within the system. The second cell signalling mechanism is an implementation of Plasmodesmata in plants (protein tunnels) [15]. The GRN produces a Plasmodesmata protein for one of the four directions N,S,E,W. If the neighbour cell is alive and produces the complementary Plasmodesmata protein, a tunnel will be established between the two cells with the effect that all proteins are equally shared between those cells, as depicted in figure 3. This tunnel is active as long as the necessary Plasmodesmata proteins are produced. Complementary to diffusion, the tunnelling mechanism should enable the GRN to break symmetries within the system, due to the fact that tunnels do not necessarily occur on all four sides of the cell at the same time.

Cell growth is achieved using the Plasmodesmata proteins. If the neighbour cell, which is targeted by the tunnel, is an empty or dead cell, it will be flagged to become alive for the next developmental step (see figure 3). Cell death is not implemented at the current stage.

Note that the cells are arranged in a toroidal fashion, i.e. the cell tissue appears to be 'infinite' for diffusion and tunnelling. It remains to be seen in future experiments, whether this is beneficial or not. Another approach to avoid boundaries is to provide a sufficiently large tissue where organisms never reach the border. However, apart from being infeasible in hardware, this potentially does not solve the boundary problem in cases where development is not stopped after a few steps.

C. Creating Functional Structures

As the GRN is intended to eventually create an executing physical system, it has to be able to configure the underlying hardware substrate. This is achieved by the structuring proteins. When expressed, these proteins use the encoded action of the gene (see figure 2) to produce a chunk of the configuration bit string for the cell's structure, i.e. its programmable hardware substrate shown in figure 4. Both the connectivity and the logic function of the cells are created by the GRN. For the experiments in this paper, the cells structure comprises two 4-bit look-up tables (LUTs) with attached flip-flops. The 2 bits in the flip-flops represent the result computed from the structural cell inputs applied to the LUTs. These two bits are used for both defining the cell state at each developmental step and as a computational output of the cell. This output is available to the four nearest neighbours of the cell. Thus, the cell states depend on the cell states in the neighbourhood and are in turn dependent on the cell states of their neighbours.

Currently, there are two unused function units per cell as illustrated in figure 4. The reason for this is that the RISA routing resources do not allow the use of all four function units in the same fashion as described above. Otherwise a 4-bit cell state would be possible. However, there are sufficient routing resources available to use the spare logic to build a LUT array, which could then perform additional computation. It would, for instance, be possible to implement an adder circuit as described in [7] under the condition that the GRN could produce a cell pattern, which could then be mapped to a suitable set of building blocks.

D. Structural Feedback

Key features that are expected from development are self-repair and adaptivity. Due to this, it is crucial that gene regulation can be affected by structural change or damage to enable the GRN to react. In the proposed system this is achieved by introducing sensor proteins that read the cell state and produce second messenger molecules according to rules, which can either be canonical—i.e. four cell states, four second messengers—or encoded in the gene action. The presented experiments are carried out using four second messengers; one dedicated second messenger molecule is produced for each cell state.

As in biological cells, the second messenger molecules cannot be directly produced by the GRN but play a significant role in gene regulation [26].

E. Environmental Coupling

Being able to adapt to a changing environment, or changing inputs, requires a mechanism that is able to detect these changes and translate them into signals that, again, affect gene regulation. This ability comes for free in this case, due to the way in which a change in the cell state results in the production of different second messengers, which then affect gene regulation. Thus, the same mechanism, which enables structural feedback, also provides environmental coupling: the routing structure of the outer cells of the organism connects to the environment via external inputs, as there are no more neighbour cells to take input from. This means that the structure of the outer cells, which computes the cell state, will possibly provide different results for different input conditions. As soon as the cell state changes, the structural feedback mechanism described in section II-D comes into action and affects gene regulation.

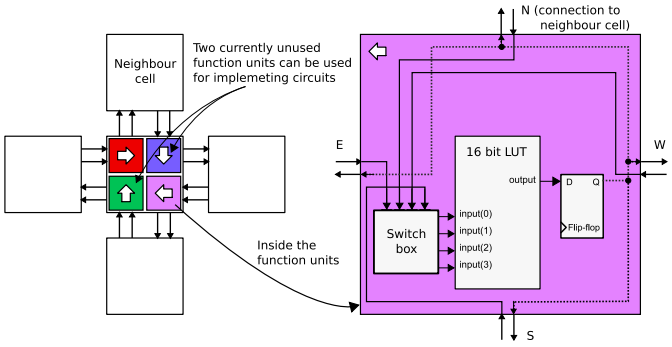

Fig. 4. *Left:* One RISA cluster is shown with the connections to its nearest neighbours. Each cluster represents one cell, resulting in 6x6 cells per chip. Each cluster features four function units that contain a 4-bit LUT, a 3-bit MUX and a flip-flop respectively. Two clusters are used as the cell's structure in the presented experiments. The results stored in the flip-flops define the two bit cell state at each developmental step. Remaining logic is used to implement a monitoring mechanism for the inner cells of the array. *Right:* the LUT of each function unit can take it's inputs from all four nearest neighbours and the cell state is provided as output to all four neighbours. The 16 configuration bits of the LUT and the 16 configuration bits of the switch box are configured by the GRN. As two of the function units are used per cell, this results in a total of 64 configuration bits per cell and 1518 configuration bits for a 36 cells organism.

III. Developmental Hardware Setup

The hardware setup used for these experiment comprises the reconfigurable integrated system array (RISA) chip and a Xilinx Spartan3E FPGA. RISA is a reconfigurable digital device, which was designed as a platform for intrinsic hardware evolution and development at the Department of Electronics, University of York. One RISA chip provides both a programmable microcontroller and configurable logic, which are inspired by the main constituents of biological cells, namely the nucleus and the cell body. Main features, which make RISA particularly suitable for (unconstrained) evolutionary and developmental experiments (examples in [14]), are the partial reconfigurability of the FPGA and the fact that it is designed in a way that it cannot be destroyed by random bit strings. A detailed description of the RISA architecture can be found in [8]. One of the future aims is to run development on the embedded processor, although it is implemented in the Spartan3E FPGA at the current stage of experiments.

A. Architecture for Development

The FPGA fabric of the RISA chip consists of 6x6 clusters, which are hardwired as described in [8]. Each cluster comprises four functional units, which provide essentially a 16 bit LUT, a 3 bit MUX and a flip-flop. In addition configurable

routing is available to route the external inputs and outputs of the cluster to the function units. Originally, one RISA chip was intended to represent one artificial cell, but as results from research in the field suggest to rather use a greater number of smaller simple cells, one RISA cluster is defined as one cell in this paper. As a consequence, the embedded processor will have to control 36 smaller, simpler cells, instead of one big, complex one.

B. Cell State and Cell Structure

In principle the developmental system can be used to configure all available resources on RISA. However, half of the resources—two function units, featuring one LUT, MUX and flip-flop each—are used for the experiments in this paper. The logic resources of the cells are allocated as shown in figure 4. The logic is configured in a way that the inputs of the LUTs can be taken from the nearest neighbours. The resulting output is then stored in the flip-flops and in turn made available to the neighbouring cells. At each developmental step, the current state of a cell is defined by the two bits that are stored in the flip-flops, hence, four cell states are possible.

IV. Measuring Properties of Development

Measuring the properties of a developmental system is not a trivial task. Unlike evolution with direct genotype to phenotype mappings, the transient component that is introduced by carrying out a number of developmental steps makes it much more difficult to define suitable fitness functions and assessments. Another transient component has to be taken into account when the developed organism carries out any kind of time dependent computation. Thus, any issues related to the evolution of the GRN itself aside, the time complexity of a developmental system aimed at creating hardware systems is at least two dimensional and the two dimensions are not separable.

In most of the research that is done into artificial development, the organisms are usually grown for $7 \ldots 30$ developmental steps. After that, development is stopped. Furthermore, there are only a few examples, where fitness at different stages of development is taken into account [18], [23]. This leads to several questions: first, how many developmental steps are actually necessary and at which step shall we demand a certain behaviour? Second, what would happen, if development was run for 50 steps, or even forever? Will it be stable in the long run? Are bistable solutions possible? Lastly, how scalable and robust are even basic solutions?

Experiments with a relatively small organism (2x2 cells) are conducted in order to obtain answers to these questions and to be able to calibrate and improve the proposed system for future experiments. The task is in all cases to find and manipulate patterns of 2-bit cell states. The examples shown are typical results of at least 10 independent runs. The DNA is always represented by 50 genes and eight proteins are used, as described in section II. Fitness calculation is straight forward in all experiments and equals the number of living cells plus the number of cells in the correct states.

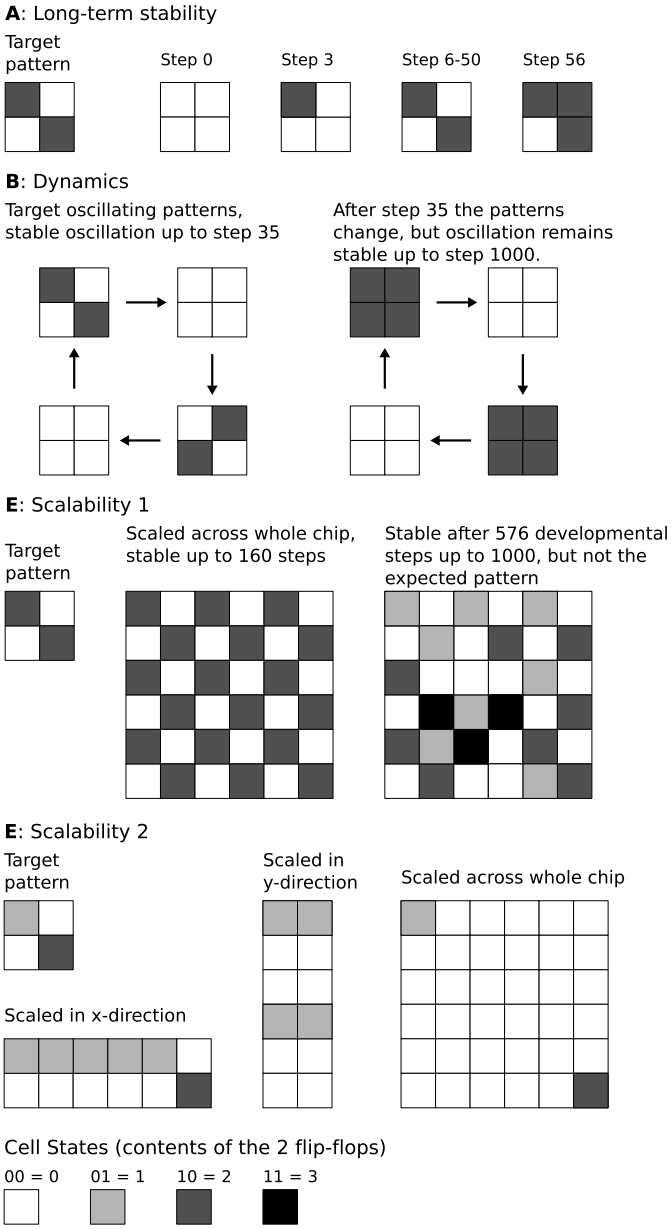

Fig. 5. Typical results for the following measured properties of the proposed development model are shown: *A*: long-term stability, *B*: dynamic behaviour, *E*: scalability. The examples shown are discussed in the text.

A. Long-term Stability

Ideally, long-term stability means that the organism grows into a desired shape or function and from a certain point remains stable forever, or at least for the longest time that is feasible to measure. Thus, the task is to generate the cell state pattern shown in figure 5-A after 5 developmental steps and then remain stable up to step 50. Fitness is accumulated after step 5. Successful solutions are then developed for 1000 developmental steps in order to investigate whether GRN has actually achieved stability.

In all experiments the pattern was found in less than 200 generations of evolution and before developmental step 6. In

most cases, the organism remained stable up to step 50, but in general only a few steps later the pattern changed into something different. In one case, the organism managed to remain stable up to step 138 and then started changing.

Thus, it is to a certain extent possible to generate stable patterns, when this property is considered in the fitness evaluation. However, in a complex dynamic system there is no guarantee beyond a certain point unless it is deterministic and fully understood. Determinism is guaranteed in the hardware system used as long as there is no random environmental input, but it is hard to prove that an evolved GRN will produce an indefinitely stable organism.

As long as we are aware of this, this is not necessarily a problem for a practical application: it just has to be considered that it will be necessary to keep developmental processes within boundaries, in which they are guaranteed to be stable.

B. Dynamics

The proposed developmental system features structural feedback mechanisms, described in section II-D. Therefore it should not only be able to maintain a stable state, but also to alternate, for instance, between two different states. Hence the task is to generate an alternating pattern, as depicted in figure 5-B(left). Pattern 1 has to be achieved after 10 developmental steps, then turn into pattern 2, then 3, then 2 and then back to 1 (following the arrows) and so on. This oscillation is required to be maintained up to step 30. Fitness is accumulated after step 10. Successful solutions are then run for 1000 developmental steps in order to observe further behaviour.

In only one out of ten runs a solution was found. The most likely reasons for the low success rate are the fact that this might be an even more challenging task than achieving stability, and that the maximum number of generations of 5000 is probably too low.

However, it is interesting to observe that in the case of the successful run the correct patterns kept alternating up to developmental step 35. After that the patterns changed into the ones shown in figure 5-B(right), but kept alternating with each developmental step up to 1000. Only nine times between step 35 and 1000 did the pattern not change for one oscillation.

Although it seems to be harder to achieve an oscillating behaviour, it appears to be more stable when development is run for a longer time. From the perspective of gene regulation this makes sense, since it is possibly easier for the GRN to exploit feedback to regulate protein production and consumption, than to completely shut down any activity and (structural) changes after a certain number of developmental steps.

C. Scalability

GRNs represent compressed, adaptive building instructions for large systems. Therefore, they should be suitable to be evolved on small problems, which they are then able to solve on a larger scale when merely provided with more resources. Therefore, experiments are undertaken where a GRN creates a 2x2 pattern using four cells. This GRN is then given all 6x6 available cells and it is observed whether the pattern repeats itself, i.e. whether some kind of scalability has been achieved.

In all experiments the target pattern has to be found after 10 developmental steps. When the evolved GRNs are run on the 6x6 cell tissue, 100 developmental steps are carried out. Almost all runs are successful in finding the 2x2 target pattern, but in general the patterns do not scale well. Examples are shown in figure 5-E. Example E1: the simple checkerboard pattern does scale across the whole organism, but does not remain stable in the long run. Example E2: the advanced checkerboard pattern does scale, but in an unexpected way. It features some kind of scalability when extended in the x direction, but not when extended in the y direction or when grown across the whole tissue (6x6 cells).

In the experiments undertaken, scalability is expected to be achieved as the cell tissue is implemented as a toroid: cell signalling and growth take place in a toroidal fashion, hence, there are no boundaries and the pattern appears 'infinite' to the GRN. Unfortunately, the hardware platform used imposes slight differences in the connectivity of the inner cells and the border cells, which breaks the symmetry.

D. Initial Position of the Zygote

The initial position of the zygote should make no difference in the appearance of the final organism. In the case of a pattern, it might be expected that it is shifted in accordance with the relative new position of the zygote, but not to look entirely different. This property could not yet be tested, due to the inhomogeneous cell tissue available (sections IV-C and III). However, simple patterns like the checkerboard pattern and the more complex checkerboard pattern (figure 5-E) can be successfully obtained independent of the initial position of the zygote, when the GRN is re-evolved for different starting positions. In all experiments, the organism is initially grown from one single zygote located in the middle of the organism (6x6 cells) or at the upper left (2x2 cells) of the tissue.

V. Patterns, Floorplans and Circuits

A valid way to achieve scalable circuits is through pattern generation in combination with predefined building blocks and routing schemes of the structural part of the cells. Examples can be found in [7], [20]. In this respect the ability to evolve patterns is valuable for practical circuit applications. Furthermore, published research results suggest that pattern formation tasks are suitable to be tackled with GRNs [5], [23].

A. Patterns

Comparing this work more closely with others, the ability of the proposed model to generate a set of different patterns with different symmetries is investigated. The selected patterns are shown in figure 6 and are inspired by patterns that are used in related papers [5], [20], [23]. The set contains patterns of different symmetries and periodicity. If the GRN is able to generate these different types of patterns, it will be shown that it is actually making use of all available cell

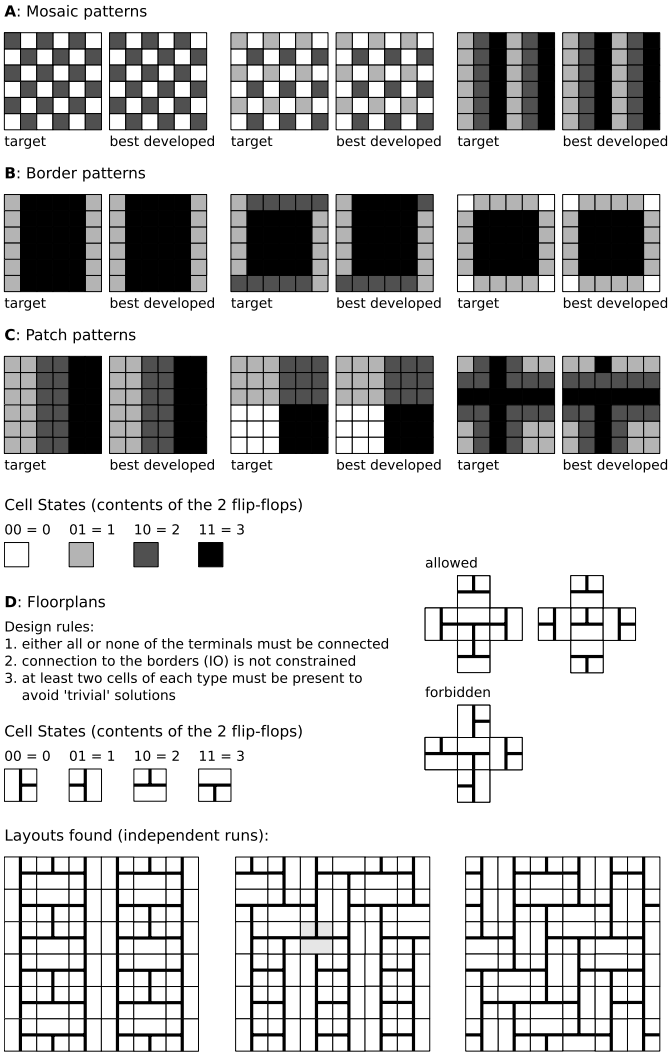

Fig. 6. The performance of the proposed developmental model is tested on the generation of the shown patterns. Classification of these patterns is done according to [5]. Similar or same kinds of patterns are widely used to test the performance of GRNs [18], [23]. It is also illustrated that the GRN can produce layouts that can be different for each run, but are satisfying the same set of design rules. The task is to find layouts of components with three terminals.

signalling mechanisms (different symmetries), can operate in an unbiased fashion and features some kind of modularity (periodic patterns). Fitness calculation is straight forward in all cases and the score equals the number of living cells plus the number of cells in the correct states.

As can be seen from figure 6, the GRN produces perfect matches to all *mosaic patterns* and almost all *border patterns* and *patch patterns*. It is observed that in those two patterns, where it comes only close to the solution, symmetry breaks are present (figure 6-B-middle and figure 6-C-right). Despite the fact that the GRN was able to handle these symmetry breaks to a certain extent, a perfect match is not achieved within 500,000 generations and 10 developmental steps.

B. Layouts

The idea of a layout aims at more practical applications: if the cell states represent different building blocks that can only be interconnected in accordance with design rules, the task for the GRN would be to create valid circuit layout. This task is also interesting insofar, that there is not necessarily only one single solution. This could be an advantage when alternatives have to be found in the case of faulty components.

In this example, each cell represents a component with three terminals, as shown in figure 6-D. The orientation of the component is defined by the cell states. Unlike in the case of the patterns, where predefined cell states have to be matched, the fitness of the layouts is calculated according to how well they satisfy the set of design rules listed in figure 6-D-left.

As can be seen from figure 6-D, different runs of evolution and development result in different layouts that satisfy the design rules. It is observed that the layouts found are featuring both a high degree of symmetry and modularity as well as symmetry breaks. The layout in figure 6-D-middle features one flaw: one of the terminals is floating (greyed cell). This represents a local optimum, since adjusting this cell would initiate a sequence of cell state changes.

C. Simple Circuits

The example given in this section is intended to give an illustration of how a simple pattern could be mapped to a circuit in a real system, rather than to claim the achievement of a robust, scalable, automatic circuit generator by means of development. Using a pattern as a circuit layout instead of directly creating a circuit is a quite simple idea, but its simplicity might make it more feasible for practical applications.

As described in section II-C, only half of the RISA cell's hardware resources are used to determine the cell state and feedback to the GRN. The other half, i.e. two LUTs and two multiplexers (MUXes), could be interconnected in a predefined fashion as suggested in [7]. Evolution of circuits on the RISA platform would then also become a pattern formation problem, which has been shown to be more successfully found by GRNs than both structure and routing. A simple scalable pattern as shown in figure 5-E2(x-scaled) could then describe a logic circuit template on a more abstract layer.

VI. Discussion

A novel model for intrinsic artificial development is introduced in this paper. It is shown that the proposed developmental system exhibits basic desired properties of artificial developmental systems, namely long-term stability, dynamic behaviour and scalability. It is observed that even organisms, which appear to be stable for a long time, can still change at a later stage. This implies that practical applications will have to be designed in a way to cope with long-term behaviour of development, e.g. by stopping development or restarting it. The model presented is able to produce stable dynamic behaviour in the form of an oscillating pattern. The results suggest that due to the dynamic, self-regulatory nature of GRNs, it might generally be easier to achieve long-term stable

oscillations than infinitely static states. Stable oscillations could also be advantageous to maintain the ability to react to environmental changes, as the GRN is not in a shut-down state. Basic scalability could be observed in the case of simple (small) patterns. In due course, a larger hardware system will be available and enable the investigation of larger, more complex patterns.

Furthermore, it is possible to grow organisms that form perfect matches to patterns of different symmetries and periodicity: *mosaic patterns*, *border patterns* and *patch patterns*. Only in cases where symmetries are broken do the developed patterns feature flaws. It is shown that the GRN is able to develop circuit layouts based on a set of rules, rather than predefined target patterns. As there is not only one possible solution, different runs result in different layouts that satisfy the design rules. It is observed that the layouts found are featuring both a high degree of symmetry and modularity as well as symmetry breaks. The results for the layouts suggest that the GRN might be able to develop alternative layouts in the case of faulty cells. Thus, the capability of the system to recover from faults and perturbations will be investigated in future experiments.

Both presented cell signalling mechanisms (*Plasmodesmata*, *diffusion*), emergent growth as well as structural feedback (*sensor proteins*) are necessary to achieve the patterns and layouts shown. Hence, these features of the developmental model are successfully used by the GRN in the presented experiments.

For the first time, a developmental approach of this kind has been investigated when run for more than 30-50 developmental steps. However, our future aim is to always keep development running in hardware, since stopping it at a certain stage would require additional mechanisms to control development. The results from this paper will be used to refine and calibrate the introduced intrinsic developmental model. Further experiments will examine scalability and environment coupling, and the behaviour of the system when going through different developmental stages will be investigated. The latter is the case in biological systems and might even be crucial to achieve scalability.

REFERENCES

[1] K. Clegg, S. Stepney, and T. Clarke, "Using feedback to regulate gene expression in a developmental control architecture," in *Proceedings of Genetic and Evolutionary Computation Conference (GECCO) 2007*. New York, NY, USA: ACM, 2007, pp. 966–973.

[2] F. Dellaert and R. Beer, *Toward an Evolvable Model of Development for Autonomous Agent Synthesis*. MIT Press Cambridge, 1994. [Online]. Available: citeseer.ist.psu.edu/dellaert94toward.html

[3] A. Devert, N. Bredeche, and M. Schoenauer, "Robust multi-cellular developmental design," in *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2007, pp. 982–989.

[4] P. Eggenberger, "Evolving morphologies of simulated 3d organisms based on differential gene expression," in *Proceedings of 4th European Conference on Artificial Life*, 1997, pp. 205–213.

[5] N. Flann, J. Hu, M. Bansal, V. Patel, and G. Podgorski, "Biological development of cell patterns: Characterizing the space of cell chemistry genetic regulatory networks," in *Proceedings of Advances in Artificial Life, 8th European Conference, ECAL 2005*, ser. Lecture Notes in Computer Science, vol. 3630. Canterbury, UK: Springer, September 2005, pp. 57–66.

[6] K. Fleischer and A. H. Barr, "A simulation testbed for the study of multicellular development: The multiple mechanisms of morphogenesis," in *Third Artificial Life Workshop*, Santa Fe, New Mexico, USA, June 1993, pp. 389–416.

[7] T. G. W. Gordon, "Exploiting development to enhance the scalability of hardware evolution," Ph.D. dissertation, University College London, July 2005.

[8] A. Greensted and A. Tyrrell, "RISA: A hardware platform for evolutionary design," in *Proceedings of 2007 IEEE Workshop on Evolvable and Adaptive Hardware*, April 2007.

[9] P. C. Haddow, G. Tufte, and P. van Remortel, "Shrinking the genotype: L-systems for EHW?" in *ICES*, 2001, pp. 128–139.

[10] G. S. Hornby, "Generative representations for evolving families of designs," in *Genetic and Evolutionary Computation GECCO-2003, volume 2724 of LNCS*. Springer-Verlag, 2003, pp. 1678–1689.

[11] J. F. Knabe, C. L. Nehaniv, and M. J. Schilstra, "Regulation of Gene Regulation - Smooth Binding with Dynamic Affinity affects Evolvability," in *IEEE Congress on Evolutionary Computation (CEC 2008). Proc WCCI 2008*. IEEE Press, 2008, pp. 890–896.

[12] J. R. Koza, *Genetic programming II: automatic discovery of reusable programs*. Cambridge, MA, USA: MIT Press, 1994.

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

[14] T. Kuyucu, M. A. Trefzer, A. J. Greensted, J. F. Miller, and A. M. Tyrrell, "Fitness functions for the unconstrained evolution of digital circuits," in *Proceedings of the 9th IEEE World Congress on Computational Intelligence (WCCI 2008)*, Hong Kong, June 2008, pp. 2589–2596.

[15] O. Leyser and S. Day, *Mechanisms in Plant Development*. Blackwell, 2003.

[16] A. Lindenmayer, "Mathematical models for cellular interactions in development. i. filaments with one-sided inputs." *Journal of Theoretical Biology*, pp. 280–299, 1968.

[17] H. Liu, J. F. Miller, and A. M. Tyrrell, "Intrinsic evolvable hardware implementation of a robust biological development model for digital systems," in *EH '05: Proceedings of the 2005 NASA/DoD Conference on Evolvable Hardware*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 87–92.

[18] J. F. Miller, "Evolving developmental programs for adaptation, morphogenesis, and self-repair," in *7th European Conference on Artificial Life*. Springer LNAI, 2003, pp. 256–265.

[19] M. Mitchell, J. P. Crutchfield, and P. T. Hraber, "Evolving cellular automata to perform computations: mechanisms and impediments," in *Proceedings of the Oji international seminar on Complex systems : from complex dynamical systems to sciences of artificial reality*. New York, NY, USA: Elsevier North-Holland, Inc., 1994, pp. 361–391.

[20] D. Roggen, "Multi-cellular reconfigurable circuits: Evolution morphogenesis and learning," Ph.D. dissertation, EPFL, 2005. [Online]. Available: citeseer.ist.psu.edu/roggen05multicellular.html

[21] A. P. Shanthi, P. Muruganandam, and R. Parthasarathi, "Enhancing the development based evolution of digital circuits," in *NASA/DoD Conference on Evolvable Hardware*, NASA/DoD. IEEE, June 2004, pp. 91–94.

[22] E. Stomeo, T. Kalganova, and C. Lambert, "Generalized disjunction decomposition for the evolution of programmable logic array structures," in *AHS '06: Proceedings of the first NASA/ESA conference on Adaptive Hardware and Systems*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 179–185.

[23] G. Tufte, "Discovery and investigation of inherent scalability in developmental genomes," in *Evolvable Systems: From Biology to Hardware, 8th International Conference, ICES 2008, Prague, Czech Republic, September 21-24, 2008. Proceedings*, ser. Lecture Notes in Computer Science, G. Hornby, L. Sekanina, and P. C. Haddow, Eds., vol. 5216. Springer, 2008, pp. 189–200.

[24] J. A. Walker and J. F. Miller, "Predicting prime numbers using cartesian genetic programming," in *Proceedings of 10th European Conference on Genetic Programming*, vol. 4445/2007. LNCS, 2007, pp. 205–216.

[25] J. Walker and J. Miller, "Evolution and acquisition of modules in cartesian genetic programming," in *EuroGp*, vol. 3003/2004. Springer Berlin / Heidelberg, 2004, pp. 187–197.

[26] L. Wolpert, R. Beddington, T. Jessel, P. Lawrence, E. Meyerowitz, and J. Smith, *Principles of Development*, 2nd ed., L. Wolpert, Ed. Oxford University Press Inc., New York, 2002.