On the Properties of Artificial Development and Its Use in Evolvable Hardware

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

Abstract— **The design of a new biologically inspired artificial developmental system is described in this paper. In general, developmental systems converge slower and are more computationally expensive than direct evolution. However, the performance trends of development indicate that the full benefit of development will arise with larger and more complex problems that exhibit some sort of regularity in their structure: thus, the aim is to evolve larger electronic systems through the modularity allowed by development. The hope is that the proposed artificial developmental system will exhibit adaptivity and fault tolerance in the future. The cell signalling and the system of Gene Regulatory Networks present in biological organisms are modelled in our developmental system, and tailored for tackling real world problems on electronic hardware. For the first time, a Gene Regulatory Network system is successfully shown to develop the complete circuit structure of a desired digital circuit without the help of another mechanism or any problem specific structuring. Experiments are presented that show the modular behaviour of the developmental system, as well as its ability to solve non-modular circuit problems.**

I. Introduction

Obtaining scalable systems and pushing the complexity barrier further in the evolution of hardware systems is a major topic in Evolvable Hardware (EHW). It is argued by many that the lack of scalability in EHW comes from using a direct genotype-phenotype mapping, and using biologically inspired techniques to increase the scalability of EHW systems is becoming more and more popular [1], [4], [5], [7], [17], [22], [27]. Among the various approaches, artificial developmental systems stand out as promising genotype to phenotype mapping techniques. Using developmental mechanisms to map a relatively simple genotype to complex phenotypes is the way nature works to obtain large systems. Hence, there is good incentive for development to be considered as an effective method for achieving scalability in EHW.

In this paper, we introduce a new artificial developmental system that models the biological developmental system using Gene Regulatory Networks (GRN). The artificial developmental system introduced is aimed to help evolution achieve more scalable, and more importantly fault tolerant and adaptive systems, which can tackle real life problems specifically on hardware. The increasing dependency on electronic devices in remote and critical applications requires the design of fault tolerant and adaptive systems; biological developmental systems provide the organisms with various fault tolerant and adaptive abilities. Ideally, an online artificial developmental system should also produce this kind of self recovering and adaptive artificial organisms. It was shown earlier by Miller, in his pattern formation experiments that a developmental system can provide implicit fault tolerance [19]. However, GRN based developmental systems designed for EHW so far have not been able to directly construct circuits as standalone systems; either a problem specific pre-structuring of circuit components had to be provided by the experimenter [7], or a separate algorithm had to be used to determine the connectivity of the developed circuits [27].

This paper will be focusing on describing the proposed artificial developmental system with some simple analysis of the mechanisms in the model and a set of experiments where we will develop circuits. It will be shown that the presented artificial developmental system is capable of developing stable organisms that exhibit modularity, and that it is capable of directly creating functional digital circuits without any help of an external algorithm (or problem specific pre-structuring) for the connectivity. Section II will present a short review of the related literature followed by Section III, which will introduce the proposed developmental system. Experiments and results will be presented in Section IV, and finally further discussion, conclusions, and future work will be presented in Section V.

II. Development in Evolvable Hardware

The most common use of evolution in the design of electronic systems is in a "single-cellular" fashion, where the whole phenotype for the desired system is implemented as a single cell and directly evolved. Such an approach requires the cell to be highly complex, and this creates an exponentially growing search space for evolution as the target system becomes more complex. Using nature as the inspiration and moving towards a multi-cellular developmental system, the evolution of a high complexity electronic system may become more achievable. Realizing this some researchers imitated nature by implementing development in various ways, and a number of the published developmental models turned out to be quite promising [7], [17], [18].

There are various models of artificial development in EHW, which range from systematic mechanisms that provide specific instructions to unfold the genotype to obtain a phenotype, to systems that closely model biological development. Modelling biology more closely in electronic system design could be advantageous, due to the fact that biological development is a well established system that exhibits a high level of evolvability [3]. However, mimicking biology

All authors are with The Department of Electronics
University of York, UK
{tk519,mt540,jfm7,amt}@ohm.york.ac.uk.
This work is part of a project that is funded by EPSRC - EP/E028381/1.

closely in the implementation of a developmental system can over-complicate the developmental system and render it practically unusable for Evolutionary Computation (EC) applications. Thus it may be more practical to have a simple developmental mechanism that is rather engineered for EC than copied from biology [22], [23].

A. Achieving Scalability and Fault Tolerance through Development

In nature, biological organisms use interactive networks of genes, Gene Regulatory Networks (GRN) [2], in order to map a small genotype (DNA) to a complex phenotype. The complex genotype-phenotype mapping in development allows the representation of a complex phenotype via a more compact genotype without the reduction of the search space. The genotype in a developmental system does not necessarily grow with the size of the phenotype, thus scales better on complex problems [4], [5], [7], [14]. Development has also been found to exhibit intrinsic modular behaviour while creating the phenotype [14]. Modularity is something that many researchers have previously tried to explicitly introduce to evolution and succeeded in obtaining an increase in the evolutionary performance [12], [26].

The complex mechanisms of biological development not only provide a way to create highly complex systems from simple DNA strands, but they also provide the organism with the ability to adapt nature. This makes it possible to achieve fault tolerance, which is an important aspect of development that would benefit the field of EHW [17], [19], [22].

III. An Interactive Artificial GRN System

The aim in designing a new Artificial Developmental System (ADS) is to obtain a system that can improve evolution to be more scalable and adaptive in designing and maintaining electronic systems on digital hardware. The general criteria that were considered important for the developmental system during its design were:

- **Simplicity:** In an ADS designed for developing systems on electronic hardware the computational overhead should not overshadow the performance of the system, otherwise its use for scalability experiments would be meaningless. A lot of the mechanisms in biological development are time consuming complex processes that require a large amount of computational overhead for accurate implementation. Thus, while implementing the developmental mechanisms, the complexity needs to be kept to a minimum.
- **Evolvability and Scalability:** One of the most important criteria is the evolvability of the artificial developmental system. The ADS has to be suitable to work with evolution in order to provide fruitful results for EHW. A biologically inspired artificial developmental model is likely to provide a more evolvable system than a mathematical model [1], [3].
  Scalability is a significant issue in EHW, and it was noted by some researchers that modelling biological development can help to tackle scalability problems in hardware evolution [7], [8], [21], [22]. The ability to evolve complex circuits is thereby as important as the evolution of adaptable and fault tolerant circuits in order to be able to use them for real life problems. Thus, a developmental system that can easily scale for higher complexity problems is an important criteria.
- **Interactivity and Stability:** Cell-extrinsic information plays a very important role in shaping the course of development in biology; in fact, it's the main source of information driving the multi-cellular development [16]. Development features a continuous adaptation that is affected by the environment. Without the extensive environmental information development can not work well and most often the cells in such a situation are bound to die. The environment that provides the cell-extrinsic information to the development includes; the other cells in the organism, the positional information of the cell in the organism, the outside resources that concern the cell, the outside organisms and the physical forces. This is another reason why biological development is a good candidate for adaptive and fault tolerant systems: it is capable of building and changing organisms with respect to the environment. Thus, it is valued that the designed ADS is interactive, and makes as much use of the environmental information as possible.
  Stability of the developmental system refers to its ability to continuously run when evolved and maintain a stable state when the target circuit is achieved. This is also referred to as online development. This has been tried by Miller and Roggen [18], [22], but has not been emphasised by many others; even though this is a crucial feature, which a successful developmental system should have, if an adaptable developmental system is desired. It is important that development can continuously run even when the target circuit is built perfectly, because this gives development the ability to adapt and repair the circuit in case of any failure or change in the environmental conditions [18].

A. Cell Interactions and Gene Regulation

An artificial GRN that is represented by binary rules was designed with the aforementioned properties in mind. A rule based system was chosen for its simplicity, flexibility, and easy representation and interpretation in hardware, which is still biologically plausible. The developmental system designed uses:

- **Diffusion and direct link communication:** A developmental system that has communication via both diffusion of proteins and direct linking of selected nearest neighbours is designed to provide both local and long distance means of interaction among the cells. Using both local and long distance cell signalling mechanisms should provide symmetry breaking properties, allowing the ADS to generate irregular as well as regular cell patterns.
  The diffusion process is carried out for all the proteins available in every cell, and it has been implemented

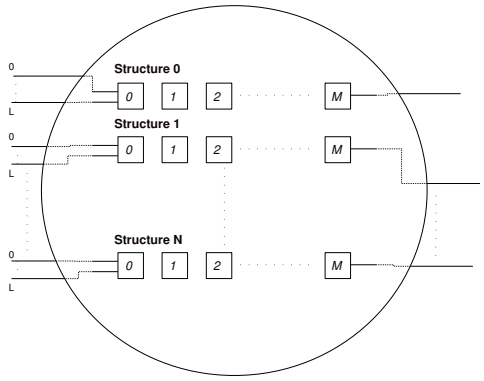

Fig. 1. The structural part of a cell: a cell can have N independent structures that can be formed of M components (gates, MUXes, CLBs, etc.). Each structure is allocated L inputs and 1 output by the cell where they can connect outside. A structure can not connect to another structure within the cell.

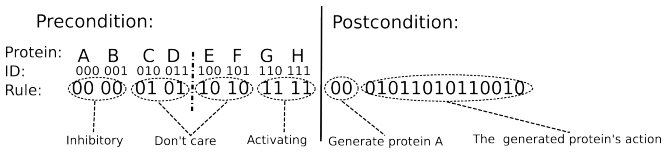

Fig. 2. An example gene formed of 32 bits is shown. The first 16 bits are reserved for the preconditional part, which specifies the rules to activate the gene. There are 8 chemicals defined in this figure; the first 4 being structuring, regulatory, sensory and plasmodesma proteins, while the last 4 are messenger molecules. Each chemical's required presence or absence is specified by a 2 bit number, which provide two don't care states. In the event of a don't care state, the presence or absence of a protein has no effect on the activation of the particular gene. The second 16 bits of the gene is reserved for the postconditional part, which provides the ID of the protein produced as a 2 bit number–this means that only the first four proteins can be produced, i.e. the messenger molecules can not be produced via the activation of a gene–which is then followed by a 14 bit number. The last 14 bits in the gene define the action of the protein produced if it has one, i.e. if it's not just a regulatory protein (further explained in Subsection III-B), if it is only a regulatory protein the last 14 bits are treated as junk.

in a simplistic way; where half of an available protein diffuses out equally to the four nearest neighbours of the source cell i.e. each neighbour obtains $\frac{1}{8}$ of the cell's protein. Protein levels are adjusted after the GRN has processed the new developmental step (including the diffusion stage) for all cells. Thus, the GRN always works with the original protein levels and the order of cell update should not bias the course of development.

- **Graded regulation:** Regulation of proteins is done at a graded level rather than boolean, thus providing a higher precision of protein diffusion, which may enable the developmental system to represent structures of higher complexity in a more compact way.
- **Multiple proteins:** The interaction of multiple proteins is allowed to affect the structural state of the cell, rather than using a single protein as a cellular state; this approach aims at providing a more compact representation.
- **Multi-Functional Cells:** Each cell in the developmental system is allowed to have a structure of $N \times M$, which means that the cells can have N physically distinct structures that are made of M components. This gives the cells the ability to do multi-tasking, since each structure has its own outputs and inputs dedicated to them, as shown in Figure 1. Although a multi cellular system should be able to achieve multi-tasking inherently, having multiple structures per cell may help with the futher partitioning of the target system. Using multi-functional cells introduces parallelism to the structural side of the developmental system, but the genes themselves are not parallelised; thus, the whole cell structure is built by a single GRN.

The proteins are defined as *n-action* proteins meaning that they can take one or more roles. In the literature of ADSs where a GRN is used there are two types of protein action implementation. The first type predefines proteins as either structuring or regulatory proteins. The second type allows all proteins to perform both structuring and regulatory functions. GRN system examples where proteins can be both regulatory and structuring are [7], [11], and examples of GRN systems where a protein has to be either regulatory or structuring are [10], [13]. In the model presented here it was decided that every protein, whatever its function, is treated as a regulatory protein as well. This seemed more natural and also a way of obtaining more complex behaviours. The types of proteins used are listed and explained in Subsection III-B.

Each gene (rule) in the GRN system described here is represented by a binary string. Similar to biology the gene string is formed of two parts: precondition and postcondition. The postconditional part of the gene specifies the resulting actions of the gene in case it is activated. The preconditional part of the gene specifies the conditions that need to be met in order to activate the gene. An example GRN within a single cell is illustrated in Figure 3.

Similar to the ADS used by Gordon [7], each gene is modeled as a rule that can be activated, inhibited or not affected by the known proteins. For a gene to be activated all the activator proteins must be present and all the inhibitor proteins must be absent. Each protein has a concentration, and it is considered to be present if its concentration is at or above the threshold level, otherwise it is considered absent. The postcondition of a gene defines the protein that is produced, and depending on the type of protein produced, the type of action it's going to take is also defined within the postcondition. Encoding of an example gene is illustrated in Figure 2.

B. Protein Types

The GRN system is composed of various proteins that are used to create rules and determine cellular actions. The proteins build the organism, create a regulatory network, and are used by the cells to communicate with other cells. The proposed GRN system has four types of proteins:

1) **Plasmodesma Protein -** Plasmodesma proteins are inspired by the *Plasmodesmata* in plants - threads of cytoplasm that breach the cell wall and connect neighbouring cells (similar to gap junction in animal

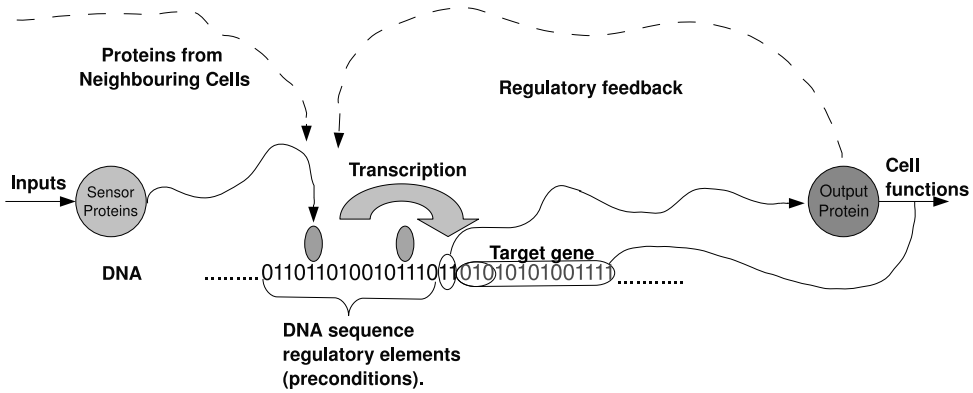

Fig. 3. The genes in a DNA sequence are activated via the correct protein activity in the GRN that cause transcription of the particular gene. In the example shown in the figure, the resulting activity of a GRN provides the correct transcription factors (the promoter proteins) to be bound at the start of a gene sequence. This then initiates the production of another protein, which can then affect the cell functionality using further information that is provided in the gene. The produced protein also creates a feedback to the GRN which regulates the transcription of further genes.

development) [16]. When a plasmodesma protein is produced, it forms a tunnel in one of the four cardinal directions (North, South, East, West), and if the neighbouring cell also has formed a tunnel in the corresponding direction, the two tunnels join together allowing the free passage of proteins between the two cells (the joint tunnels are termed plasmodesmata in this paper). After the two cells are connected by plasmodesmata (tunnels), they share the same protein distribution. If the neighbouring cell (of the cell where the plasmodesma is produced) is dead, a cell division is initiated. During the cell division plasmodesmata is created connecting the two daughter cells (one of which replaces the dead cell). Although in plants the Plasmodesmata size is known to shrink as the cells mature (thus filtering larger proteins) [16], this is omitted in our developmental system for simplicity. However, the callose deposition is simulated. The callose deposition causes the blockage of the plasmodesmata, which prevents the transport of all proteins through the plasmodesmata. In our system this is initiated by the plasmodesma protein that has callose deposition encoded in its target gene.

2) **Structuring Protein -** Structuring proteins are the type of proteins that change the physical structure of the cell, i.e. the circuit. When a structuring protein is produced, it uses the further information provided by the gene to alter the circuitry within the cell.
3) **Sensor Protein -** The sensor proteins are produced by the GRN to act as sensors around the cell, which monitor outside activity. The sensor proteins produce different kinds of messenger molecules for different types of outside activity. This enables environmental factors to affect the GRN activity, creating a more interactive developmental system.
4) **Regulatory Protein -** Although every protein is a regulatory protein, there are proteins that act only as regulatory proteins. These proteins, just like every other protein, control the GRN activity by their presence or absence, but they do not have any other purpose. Thus when a regulatory protein is produced as a product of a gene, the further information provided by the postcondition of the gene is discarded. The unused parts of the gene provide a neutral search space, which can have positive effects on evolvability. It has been shown that neutrality provides stability via preventing deleterious mutations, and helps evolution avoid local optima, thus creating a smoother search space [9], [25].

. Apart from the four proteins introduced, there are four other types of chemicals that behave similar to the proteins, however have different regulatory properties. Hence, these chemicals are referred to as "molecules". The said chemicals are called **Messenger Molecules:** The messenger molecules are produced by the sensor proteins that are affected by the outside activity. The messenger molecules can not be produced directly as a result of gene activity since their purpose is to regulate the gene activity via the environmental response, but they can still bind to activate or deactivate genes. These five types of chemicals should be sufficient to form a satisfactory mechanism for the desired GRN. The way the chemicals are structured gives the cells the ability to form a complete interactive multicellular organism. An example GRN acting at a multicellular level is shown in Figure 4.

IV. Circuit Development

The experiments presented in this paper are initial experiments using the proposed developmental system, and they are carried out as software simulations. The developmental system is evolved to develop a phenotype in the form of a Cartesian Genetic Programming (CGP) netlist [20], which solves various circuit design problems using multiplexers as the building blocks (see Table I for the type of MUXes used).

For all the experiments in this paper: the concentration of each protein is represented by an 8-bit unsigned number; thus

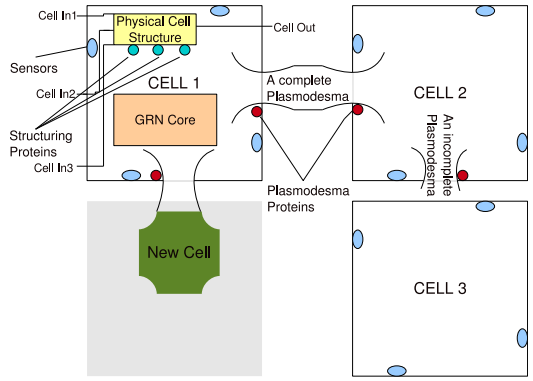

Fig. 4. In a multi-cellular environment using the 4 basic protein types a cell is able to: interact with it's environment, multiply, structure itself, and form a complex multicellular organism. The basic functions of some proteins are demonstrated in this figure; only cell 1 is drawn completely, certain components are omitted in other cells. In the example above, cells 1 and 2 both have active plasmodesma 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 plasmodesma proteins on their southern sides. Cell 1's southern neighbour is a dead cell, so the active plasmodesma protein initiates a growth process in that direction. However, cell 2's southern neighbour is an alive cell with no plasmodesma protein, thus cell 2 forms an unconnected channel on it's southern wall. The sensors drawn monitor the outside activity on 4 sides of each cell and produce messenger molecules with respect to the 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.

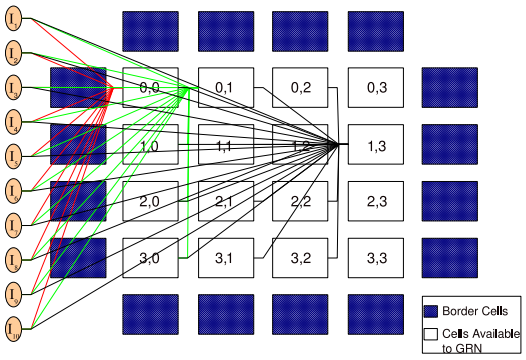

Fig. 5. An example 4×4 organism with 10 inputs is shown. The available structural connections for cells *0,0*; *0,1* and *1,3* are drawn to illustrate how the cells can connect to form a CGP circuit.

the maximum concentration is 255. The threshold level (explained in Subsection III-A) is half the maximum concentration level for all cells and proteins. The organism is initialised with one cell alive in the middle of the virtual cellular space. The components used for building the structural part of the cells are 4 different types of MUXes, shown in Table I. The phenotype of the developed organism is mapped into a CGP program for a convenient way of simulating digital circuits. For these initial experiments the sensor proteins and hence the messenger molecules are not used.

A. Mapping the Organism to a CGP Program

The maximum organism size is predefined, and the cells are ordered in columns to form an organism of size $n \times n$. As described in Figure 4, each cell builds a structural part of its own. These structural parts are represented as a constant length bitstring in the experiments described here. This bitstring is then decoded to obtain a number of CGP nodes. The CGP network evolved is feedforward, thus each node within a cell can connect to the other nodes within the cell to create a feedforward combinatorial circuit. Each cell has 3 external connections, and it is allowed to connect to the external inputs and cells from the previous columns, as described in Figure 5. The number of structural components per cell is predefined at the start of evolution. The software representation of the organism has 4 cell states defined:

1) **Dead:** initially all but one of the cells are in this state, when a cell is dead it is inactive and does not carry out any GRN activity. To activate a dead cell one of the alive cells needs to grow in to the place of a dead cell.
2) **Alive:** when a cell is alive, it carries out all the GRN and structural functions normally, and none of the other alive cells can grow over it.
3) **Border:** if a cell is in a border state it is considered dead, but none of the alive cells next to it can grow in place of it, border state was defined in our software simulations to decrease the number of exceptions.
4) **Fetus:** when a cell grows over a dead cell the state of the dead cell is changed to fetus, when in fetus state the cell is unable to carry out normal cell functions until the next developmental step when it becomes 'alive'.

B. Developing a Parity Solving Organism

In order to demonstrate how the GRN mechanism works, a 2 cell organism is evolved that develops to function as an XOR gate. The organism is formed of 5 genes using 4 proteins (1 plasmodesma, 1 structuring, 2 regulatory), and it develops to an adult organism in 3 developmental steps. It was evolved with the maximum number of cells limited to 4. Surprisingly, the evolved organism has the potential to grow to become an n-bit parity solving organism if the maximum number of cells available were increased. As shown in Figure 6, the GRN of the evolved organism keeps replicating an XOR gate for each row of cells. The growth of the XOR organism in more detail is shown in Figure 6. The example shown in Figure 6 demonstrates the proposed developmental system's ability to achieve modularity and stability.

C. Development of an Even n-bit Parity Circuit

Parity problem is easy to implement and popular amongst EHW researchers. It has been used by many to test the scalability of their systems [7], [12]. Thus, it is regarded as a suitable initial test problem for the proposed developmental system. Developmental systems are evolved that solve 5, 7, 9, 10, 11, 12 - bit even parity circuits.

In the even n-bit parity experiments, the organism is limited to 100 cells (10×10 cells) with each cell having 3 inputs, 1 output and a 2 component structure, and the number of genes per genome is defined as 50, and each gene

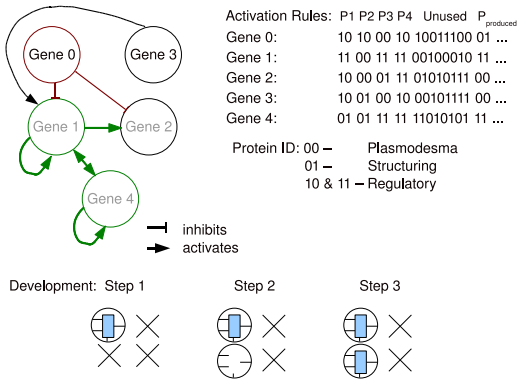

Fig. 6. An example 2 cell organism that acts as an XOR gate is shown, with the list of evolved genes, and the GRN graph for the alive cells (identical for both cells in this case). At developmental step 1, only one cell is alive in the organism. Looking at the GRN rules it can be seen that, Gene 0 and Gene 3 do not need any promoters in order to activate and the suppressor proteins initially do not exist (initially none of the proteins exist in the system). Thus Gene 0 and 3 would be active at the first stages of the GRN, while the others are inactive. Gene 3 would produce Plasmodesma Protein targeted south (this information is encoded in the postconditional bitstring, which is not shown here), which would then trigger a growth process. On developmental step 2 the second cell grows and shares its proteins with the parent cell. By step 3 the organism stabilises with two identical cells, each implementing an XOR with a MUX but only the daughter cell being connected to the output, if the development would run further, the organism would be stabilised at the stage it reaches at age 3, which is an XOR gate. This demonstrates the ability of the developmental system as a stable online developmental system, which is a desired property in order to achieve adaptivity and fault tolerance via development. The developed organism in this example also demonstrates a modular and redundant behaviour by replicating the same physical structure in both of the cells, which are desirable features for a fault tolerant and scalable system. If the organism was not limited to 2×2 cells, the daughter cell from step 2 would trigger a growth process into its southern neighbour, which would then become a chain of infinite growth events, resulting in the growth of an infinitely sized parity solving organism.

TABLE I

THE MULTIPLEXERS USED FOR THE EXPERIMENTS PRESENTED.

MUX1	$(A \ \& \ \overline{C}) \| (B \ \& \ C)$
MUX2	$(A \ \& \ \overline{C}) \| (\overline{B} \ \& \ C)$
MUX3	$(\overline{A} \ \& \ \overline{C}) \| (B \ \& \ C)$
MUX4	$(\overline{A} \ \& \ \overline{C}) \| (\overline{B} \ \& \ C)$

is represented by a 64 bit number. Thus, the evolved genome is 3200 bits long $(64[bitspergene] * 50[genes])$.

The phenotype is mapped into a CGP program, where each CGP node (formed of 3 inputs and a MUX) is represented by a 32 bit number (8 bits for each input and MUX). Thus the total size of the phenotype is 6400 bits $(32[bitspernode] * 2[gatespercell] * 100[cells])$. The number of proteins used in these experiments are 4; 3 of them being structuring, while the fourth one is a plasmodesma protein. The organism is allowed to age for 10 developmental steps and then evaluated once at age 10. Hierarchichal Bitstring Sampling (HBS) fitness function is used to check if the developed circuit functions correctly [15]. Therefore, the developmental system goes offline once it reaches maturity in these experiments. The evolutionary algorithm used is an Evolutionary Strategy (ES) (2+5) with an adaptable mutation rate in the 1-25 % range that changes with respect to the *rate of change* in the fitness. The maximum number of evolutionary generations is 2 million.

TABLE II

RESULTS OF THE EVEN PARITY EXPERIMENTS. EACH EXPERIMENT IS CARRIED OUT 30 TIMES; THE NUMBER OF SUCCESSFULLY EVOLVED CIRCUITS, TOTAL NUMBER OF GENERATIONS PER AVERAGE SUCCESSFUL RUN, AND THE STANDARD DEVIATION OF AVERAGE NUMBER OF GENERATIONS FOR THE SUCCESSFUL RUNS ARE LISTED. THE DEVELOPMENTAL SYSTEM SUCCESSFULLY DEVELOPS ALL THE TESTED SIZES OF PARITY CIRCUITS DEMONSTRATING ITS ABILITY TO DIRECTLY BUILD CIRCUITS.

Circuit Evolved	Evolutionary Mechanism	Results		
		Sol Found	Avg Gens	Std Dev
5-bit Parity	Development	30	2072	1350
7-bit Parity	Development	30	7216	7669
9-bit Parity	Development	30	22024	40706
10-bit Parity	Development	29	88222	112452
11-bit Parity	Development	30	161858	166388
12-bit Parity	Development	30	288481	378401

On almost all of the runs presented in Table II development has a perfect success rate in building parity circuits. It is a good achievement for development that it is able to directly produce functional circuit structures in a reasonably quick amount of time. The common way of developing circuits in literature has been to use pattern mapping – that removes the need to deal with routing while developing the circuit – when a circuit was evolved using a developmental system. The developmental system can then develop a pattern rather than directly the desired circuit [7], [17], [24].

On the figures presented in Table II the ADS's performance decreases as the circuit size gets bigger. There are a number of factors that could be the reason for this: for all the experiments presented in Table II, the maximum size of the organism was kept to 10×10 cells. Therefore, the increase in the average number of generations for each run may simply be the insufficient number of cells. If this is the case, increasing the number of cells in the developmental system might solve this problem. Another change in the developmental system that can increase the performance in developing parity circuits is decreasing the component-to-cell ratio, i.e. decreasing it from 2 to 1. This is because partitioning a parity circuit using a single MUX is a much easier task than partitioning it with two MUXes. Another non-trivial factor that could affect the performance of the developmental system can be the number of developmental steps that are allowed before evaluating the resulting circuit. Ten developmental steps are possibly too low for developing circuits bigger than 9 bit parity.

To investigate the actual effects of some of the predicted factors that can affect the developmental performance, a number of further experiments are undertaken with selected parity problems for different numbers of developmental steps and organism sizes.

Looking at Table III, it is interesting to see that the number of evolutionary generations required for the developmental

TABLE III

RESULTS OF 9-BIT PARITY EXPERIMENTS WITH DIFFERENT DEVELOPMENTAL STEPS. THIRTY EVOLUTIONARY RUNS WERE DONE FOR EACH EXPERIMENT LISTED.

Circuit Evolved	Cell Age	Sol Found	Avg Gens
9-bit Parity	7	30	29321
9-bit Parity	10	30	22024
9-bit Parity	15	30	17038
9-bit Parity	25	30	17153
9-bit Parity	35	30	14361
9-bit Parity	42	30	13837
9-bit Parity	50	30	13601
9-bit Parity	75	30	10780

TABLE IV

RESULTS OF THE DEVELOPMENT OF LARGER PARITY EXPERIMENTS (10-12 BIT) WITH BIGGER ORGANISM SIZES.

Circuit Evolved	Organism Size	Sol Found	Avg Gens	Std Dev
10-bit Parity	10x10	29	88222	112452
10-bit Parity	11x11	30	93594	168272
11-bit Parity	10x10	30	161858	166388
11-bit Parity	12x12	30	110454	123963
12-bit Parity	10x10	30	288481	378401
12-bit Parity	13x13	28	257757	349526
12-bit Parity	15x15	27	242716	176216

system in developing a fully functional circuit decreases as the number of developmental steps (cell age) required before the organism is considered mature is increased. This indicates that the developmental system might become more evolvable as it's developed for longer. Although the improvement on the number of evolutionary generations in Table III is not drastic with the change of developmental steps, it is almost steady with the increasing developmental steps. Further more in depth study of this behaviour may yield interesting and valuable results.

The developmental system was provided with a fixed maximum organism size regardless of the size of the tackled problem, in order to determine whether it would always use all the given resources or would be capable of determining the amount of resources it needs to use. In the experiments carried out, development is able to determine the amount of cells it needs to solve the problem, rather than using every single available cell location. Thus, the maximum number of cells provided is not always required for final organism, and the developmental system is capable of controlling the growth of the organism(even without the use of programmed cell death). For example, for all the 12-bit parity circuits all the cells were alive for the fully developed organism, whereas for the 5-bit parity circuit most of the time only 30-50 cells would be alive for the fully developed organism.

To investigate the effects of using a bigger organism on the performance of the developmental system in finding valid circuits, the experiments for developing 10, 11, and 12 bit parity problems are carried out with more resources. Increasing the organism size did lower the number of generations

TABLE V

RESULTS OF THE 2-BIT MULTIPLIER EXPERIMENTS ARE GIVEN.

Circuit Evolved	Sol Found	Avg Gens	Std Dev
2-bit Multiplier - Development	15	460310	311717

in most cases as it can be seen from Table IV. In some cases (namely the 12 bit parity), the success rate slightly dropped with the increase of organism size. The results from Table IV suggest that the larger parity problems benefit from bigger organism sizes, but when the organism size is too big the likely-hood of not finding a solution is higher as well.

D. Development of a 2-bit Multiplier

As a second demonstrator application, a 2-bit multiplier is evolved to see if the proposed developmental system is able to tackle problems that have multiple outputs. For the multiplier experiments the organism is limited to 100 cells with each cell having 4 single component structures using the MUXes mentioned earlier in Table I. The number of genes per genome is defined as 100, and each cell has 12 inputs and 4 outputs (thus each cell is structured in 4-function mode). The number of proteins used are 4; 3 of them being structuring, while the fourth is a plasmodesma protein. The organism is allowed to age for 15 developmental steps and then evaluated. The maximum number of generations for the EA was set to 1 million generations, and an ES(5+20) was used.

The developmental system is successfully evolved to develop a 2-bit multiplier 15 out of 30 times, which shows that the developmental system is capable of developing multiple output circuits. The successful direct implementation of a multiplier by the developmental system suggests that the ADS is also capable of developing asymmetric organisms; implementation of a multiplier circuit using multiplexers is not as regular and symmetric as the implementation of a parity circuit. It is an important feature of an ADS to be capable of developing asymmetrical organisms, because this means that it is not biased towards particular symmetric shapes and should be able to develop organisms with different patterns of cell types.

From the experiments presented here it can be seen that development is capable of developing circuit structures directly. The ADS is primarily aimed at developing large fault tolerant and adaptive systems that benefit from the high modularity of the development. Therefore, the ADS should also be useful for the design of complex patterns that can be interpreted as electronic systems, rather than the direct development of digital circuits using basic logic components.

V. DISCUSSION AND FUTURE WORK

A new artificial developmental system has been introduced, and first experiments show how the developmental system works. The results suggest that it is able to scale and stay stable once the desired state is reached and the task is solved. The mechanisms of the presented developmental

system are designed to be biologically defensible, but they have been kept simple enough to keep them computationally inexpensive. To the authors' knowledge, this was the first time that a GRN based developmental mechanism has been successfully used without an extra mapping mechanism or problem specific structuring for the design of logic circuits. Even with the direct development of circuits, the developmental system has a high success rate in obtaining functional circuits.

The experiments presented in this paper were all done in simulation, so the ADS had to be further simplified from some of the mechanisms that are aimed for hardware development (such as sensor proteins and messenger molecules); thus, it is hoped that the performance of the developmental system would further improve on hardware. The experiments investigate some properties of developmental system such as the importance of the number of developmental steps and the maximum organism size, which directly affect its performance.

The ADS described here is designed to be used for intrinsic hardware development, and provides an interactive and stable system for the development of systems in hardware. Thus, the main aim of the developed circuits are not to implement traditional computational circuits but hardware applications that could benefit from adaptivity and fault tolerance. It has been shown that GRN based ADSs are powerful in constructing complex patterns [6]. Thus, the ADS presented here should also prove to be successful in constructing patterns to achieve the desired systems.

In the parity circuits presented here, it was observed that in most cases the developed circuits had a very high count of repetitiveness in their components and only a small amount of the cells had different genetic and physical structures than the others. In the first example the ADS was evolved to develop a generic parity organism, which was an important feature of a developmental system to demonstrate. In future work experiments will be done to further investigate the capability of the ADS to use the genetic code of a smaller circuit (e.g 5 bit parity) to obtain a larger one (e.g. 12 bit parity). Also, the application of the ADS on hardware will be tested; it is expected to be faster and capable of running online (no predetermined developmental steps) when run on hardware. However, more importantly, the new ADS will be tested in the development of various patterns, and evolved to develop patterns that can be used to create complex electronic systems in hardware. It will then be further investigated for fault tolerance, scalability and adaptivity, and applied to real world applications.

REFERENCES

[1] Peter Bentley and Sanjeev Kumar. Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 35–43, Orlando, Florida, USA, 13-17 1999. Morgan Kaufmann.

[2] Eric Davidson. *The Regulatory Genome: Gene Regulatory Networks In Development And Evolution*. ACADEMIC PRESS, 1 edition, 2006.

[3] Richard Dawkins. *On Growth, Form and Computers*, chapter The evolution of Evolvability, pages 239–255. Elsevier Academic Press, 2003.

[4] F. Dellaert and R.D. Beer. *Toward an Evolvable Model of Development for Autonomous Agent Synthesis*. MIT Press Cambridge, 1994.

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

[6] Nicholas Flann, Hu Jing, Mayank Bansal, Vinay Patel, and Greg Podgorski. Biological development of cell patterns : Characterizing the space of cell chemistry genetic regulatory networks. In *8th European conference, ECAL*. Springer Berlin / Heidelberg, 2005.

[7] Timothy Glennie Wilson Gordon. *Exploiting Development to Enhance the Scalability of Hardware Evolution*. PhD thesis, University College London, July 2005.

[8] Pauline C. Haddow, Gunnar Tufte, and Piet van Remortel. Shrinking the genotype: L-systems for EHW? In *ICES*, pages 128–139, 2001.

[9] I. Harvey and A. Thompson. Through the labyrinth evolution finds a way: A silicon ridge. In *Proc. 1st Int.Conf.on Evolvable Systems (ICES'96)*, volume 1259 of *LNCS*, pages 406–422. Springer-Verlag, 1997.

[10] Nick Jakobi. Harnessing morphogenesis. In *International Conference on Information Processing in Cells and Tissues*, pages 29–41, 1995.

[11] Hiroaki Kitano. A simple model of neurogenesis and cell differentiation based on evolutionary large-scale chaos. *Artif. Life*, 2(1):79–99, 1995.

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

[13] S. Kumar and P. J. Bentley. Biologically plausible evolutionary development. In *In Proc. of ICES 03, the 5th International Conference on Evolvable Systems: From Biology to Hardware*, pages 57–68, 2003.

[14] S. Kumar and P.J. Bentley, editors. *On Growth, Form and Computers*. Elsevier Academic Press, 2003.

[15] Tüze Kuyucu, Martin Trefzer, Andy Greensted, Julian Miller, and Andy Tyrrell. Fitness functions for the unconstrained evolution of digital circuits. In *9th IEEE Congress on Evolutionary Computation (CEC08)*, pages 2589–2596, Hong Kong, June 2008.

[16] Ottoline Leyser and Stephen Day. *Mechanisms in Plant Development*. Blackwell, 2003.

[17] Heng Liu. *Biological Development model for the Design of Robust Digital System*. PhD thesis, University of York, September 2007.

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

[19] Julian F. Miller. Evolving a self-repairing, self-regulating, french flag organism. In *GECCO Proceedings*, 2004.

[20] Julian F. Miller and Peter Thomson. Cartesian genetic programming. In *Genetic Programming, Proceedings of EuroGP'2000*, pages 121–132. Springer-Verlag, 2000.

[21] Julian F. Miller and Peter Thomson. A developmental method for growing graphs and circuits. In *Evolvable Systems: From Biology to Hardware, 5th International Conference*, pages 93–104, 2003.

[22] Daniel Roggen. *Multi-Cellular Reconfigurable Circuits: Evolution Morphogenesis and Learning*. PhD thesis, EPFL, 2005.

[23] Gianluca Tempesti, Daniel Mange, Enrico Petraglio, Andre Stauffer, and Yann Thoma. Developmental processes in silicon: An engineering perspective. In *EH '03: Proceedings of the 2003 NASA/DoD Conference on Evolvable Hardware*, page 265, Washington, DC, USA, 2003. IEEE Computer Society.

[24] Gunnar Tufte. Discovery and investigation of inherent scalability in developmental genomes. In *8th International Conference on Evolvable Systems: From Biology to Hardware*, LNCS, pages 189–201. Springer, 2008.

[25] V.K. Vassilev and J.F. Miller. The advantages of landscape neutrality in digital circuit evolution. In *Proceedings of the 3rd International Conference on Evolvable Systems: From Biology to Hardware*, pages 252–26. Springer, 2000.

[26] J.A. Walker and J.F. Miller. Evolution and acquisition of modules in cartesian genetic programming. In *EuroGp*, volume 3003/2004, pages 187–197. Springer Berlin / Heidelberg, 2004.

[27] Song Zhan, Julian F. Miller, and Andy M. Tyrrell. A developmental gene regulation network for constructing electronic circuits. In *8th International Conference on Evolvable Systems: From Biology to Hardware*, pages 177–188. Springer-Verlag, 2008.