# An Intrinsic Robust Transient Fault-Tolerant Developmental Model for Digital Systems

Heng Liu[*], Julian F. Miller[*], Andy M. Tyrrell[*]

[*] University of York, Department of Electronics, Bio-inspired Architectures Lab, Heslington, York YO10 5DD, UK
{hl142, jfm, amt}@ohm.york.ac.uk

**Abstract.** A biologically inspired developmental model targeted at hardware implementation (off-shelf FPGA) is proposed which exhibits extremely robust transient fault-tolerant capability: in the software simulation of the experimental application. In a 6x6 cell French Flag, some individuals were discovered using evolution that have the ability to "recover" themselves from almost any kinds of transient faults, even in the worst case of only one "live" cell remaining. All cells in this model have identical genotype (physical structures), and only differ in internal states.

## 1 Introduction

Living multi-cellular biological organisms exhibit several intrinsic characteristics electronic engineers earnestly long for, such as growth, self-repair and fault-tolerance. Multicellular living organisms achieved these traits through millions of years of evolution by means of cells which have identical genotypes. All of them come from a single special cell (zygote): this process is named development. The entire process of development is controlled by the interaction of cells rather than by a centralized process.

### 1.1 Background of Development Principles

The development of an embryo is determined by genes, which control where, when and how many proteins are synthesized [1]. Complex interactions between various proteins and between proteins and genes within cells and hence interactions between cells are set up by activities of genes. It is these interactions that control how the embryo develops.

Development involves cell division, the emergence of pattern, change in form, cell differentiation and growth. The model proposed in this paper contains only two of these aspects, cell division and differentiation. Since in hardware, no new resources can be created as cells are pre-formatted and their number can not be increased, "growth" is used in this report to refer to "cell division".

Cell differentiation emerges as a result of differences in gene activities which lead to the synthesis of different proteins. As development is progressive, the fate of cells becomes determined at different times. Inductive interactions by means of chemicals or proteins between cells can make cells different from each other and the response to these inductive signals depends on the state of this cell. Patterning can involve the interpretation of positional information and lateral inhibition.

## 1.2 Related Projects

This work is based upon Dr. J. Miller's previous research [2]. In his paper a method for evolving a developmental program inside a cell to create multi-cellular organisms of arbitrary size is described and its characteristics are analyzed. He found that the artificial organism is able to organize itself into well defined patterns of differentiated cell types, such as the French Flag. Emergent properties, such as self-repair and adaptation were observed in the solutions found using an evolutionary strategy: one of the matured organism was subjected to a harsh damage, and it can reconstruct itself in several steps of growth (with some scarring); another organism can respond appropriately to global environmental signals which lead to the metamorphosis of the whole organism.

## 1.3 Conventional Fault-tolerant Techniques

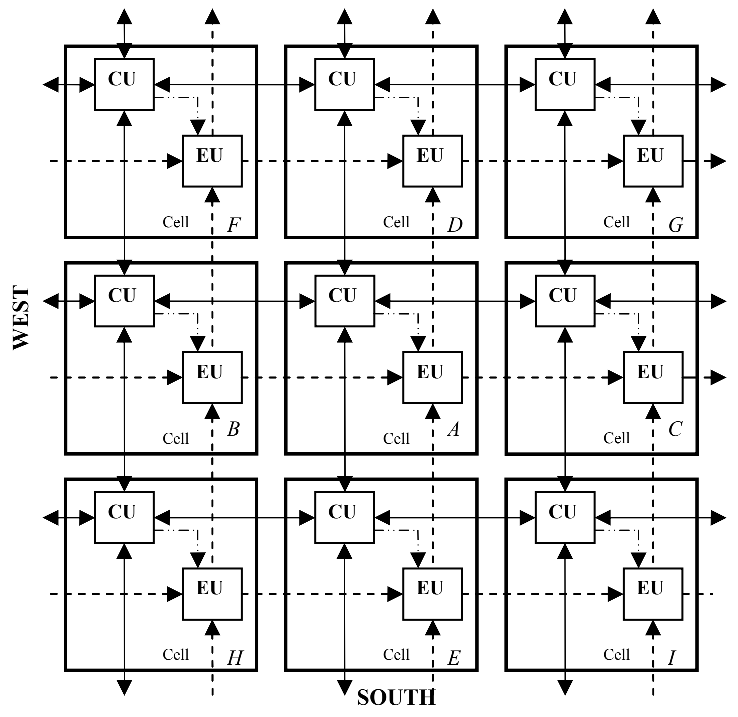
Built-in redundancies and error handling capabilities are the most widely used conventional fault-tolerant technologies. Redundancies can be employed either spatially or temporally. Spatial (area) redundancy can be applied using Dual Modular Redundancy or Triple Modular Redundancy, both of which are based on the majority vote of individual modules. In temporal (time) redundancy techniques, after an error output is detected, it is recomputed in an attempt to recover from the transient fault. Although time redundancy in general requires fewer resources than area redundancy, it demands error handling capability which will incontrovertibly increase the complexity of the system and its design cost. What's more, it is difficult to design such an error handling circuit which stores adequate information for recovery so that it can discover most transient faults.

Transient faults account for most system failures [6]: this is why this paper will concentrate on transient fault-tolerance.

## 1.4 Main Objective

The final objective of our work is to propose a general FPGA-based evolutionary developmental model, which has the emergent properties of adaptation and fault-recovery. At least one real-world application will be implemented on the FPGA using this model. The results of this application and the intrinsic characteristics of this model will be analyzed.

In the first stage of this work (which is described here), a software simulation to verify its applicability and feasibility is produced, and the intended application also was to construct cell patterns that represent the French Flag.



LEGENDS:

| | | | |
|---|---|---|---|
| CU | Control Unit | EU | Execution Unit |
| -··-··▶ | EU Function Selection | ◀──▶ | States Signals |
| ━━━ | Cell border | ----▶ | Executing Signals |

**Fig. 1.** Inter-connection of Cells

Section 2 discusses the details of the model proposed. The new evolutionary algorithm employed in this work is described in section 3. Experiments carried out and the results are presented in section 4. Conclusion and future work are presented in the last section.

## 2 Development Cellular Model

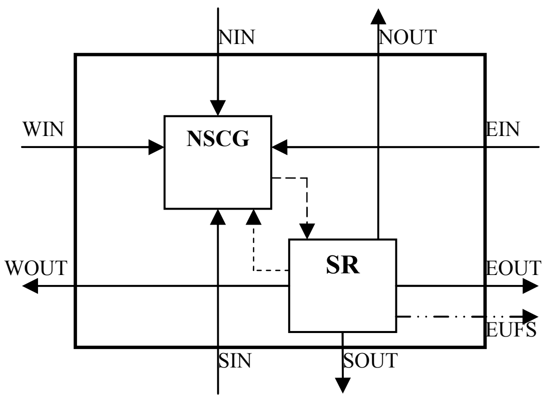
The model currently used in the software-simulation is presented in this section.

### 2.1 Cell Structure and Inter-Cell Connections

As shown in Fig. 1, a cell is composed of two main components: Control Unit (CU) and Execution Unit (EU).

Each Control Unit has a States Register (SR in Fig. 2), which stores the internal states of the cell, including the cell type and chemicals. Each CU connects to its 4 immediate neighbors and a Next States & Chemical Generator (NSCG in Fig. 2) determines its own next state/chemicals according to the current states and chemicals of the neighbors, the state and its own chemical (Fig. 3).

At present only a combinational application is considered, hence the Execution Units are purely combinational circuits. Each EU's inputs come from its immediate west or south neighbor, which is determined by this cell's state (saved in SR in the CU). Every EU also propagates its output (Executing Signals) to its immediate north and east neighbors.



LEGENDS:

| | | | |
|---|---|---|---|
| NSCG | Next States & Chemical Generator | SR | Cell States & Chemical Register |
| ——▶ | States & Chemical Signals from neighbors | - - - -▶ | States & Chemical Signals of this cell |
| –··–··–▶ | Intra-cell Signals | EUFS | EU Function Selection |

**Fig. 2.** Control Unit Structure

Both the structure of EU and CU are determined by evolution, so the genotype encodes the EU and CU internal structures. The representation of the inter-structure of EU and CU will be described in "2.2 Genotype Representation".

Since the French flag is made up from the state of all the cells, there are no EUs, only a CUs network. However, in the future, with other applications the addition of EUs will be incorporated.

## 2.2 Genotype Representation

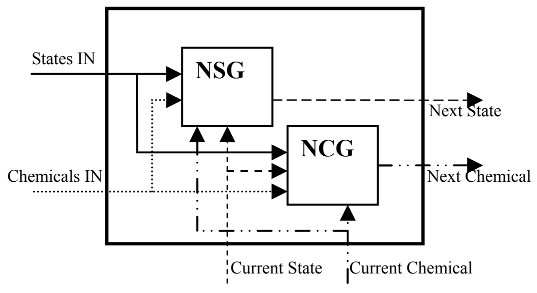
Cartesian Genetic Programming [3] was used as in [2]. However in order to make full use of modern CPU's 32 bits wide data bus to speed up the simulation procedure, high level functions were chosen, such as ADD, AND, OR, XOR, RIGHT SHIFT (all of which are 32 bits wide operators).

One instance of these functions with 3 inputs (at present) is called a molecule. A digital circuit is encoded as *n* rows, *m* columns molecules: each molecule's input can only connect to molecules in its left columns, no connection within the same column is allowed. These regulations will ensure the feed-forward nature of the circuit.

The molecules in the left most column can only receive data from the inputs to the system. A molecule's output can connect to inputs of molecules no further than *p* columns away on the right. So *p* is termed "Back Level Depth".

As 32-bit is not practical to be implemented in hardware, a smaller data width is chosen. *w* is used as the data width for a given circuit.

A digital circuit is encoded as *n* x *m* molecules, with data width *w*, and back level depth *p*. From now on, this is abbreviated as "the setting for a gene is *n*x*m@w-p*" (In this work, one gene encodes one circuit).



LEGENDS:

| | | | |
|---|---|---|---|
| NSG | Next States Generator | NCG | Next Chemical Generator |
| ——▶ | States Signals from neighbors | ⋯⋯▶ | Chemical Signals from neighbors |
| –··–··–▶ | Chemical of this cell | - - - -▶ | State of this cell |

**Fig. 3.** NSCG Structure

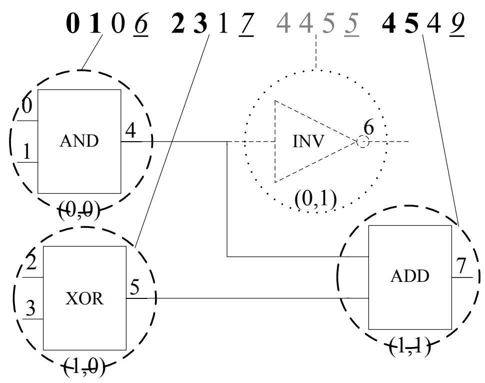
A sample circuit with gene setting *2*x*2@8-1* is shown in Fig. 4. This circuit which is composed of a maximum of 4 molecules has 4 inputs and one output.

## 2.3 Cell Growth

Given a genotype, the inner-structure of the cells is determined and a zygote which is located at the centre of an ‘artificial environment’ with *x* rows and *y* columns of cells can be initiated and ‘duplicate’ itself. The inputs to the cells on the border of this environment are fixed to 0. As shown below, without chemicals no cells can live. So initially, some chemicals must be injected at the position of the zygote.

The growth procedure is as follows:

1. Initialize chemical and the zygote;
2. Chemical diffusion; (details in section 2.4 Chemical Diffusion)
3. All cells update their state simultaneously:
    4. If no chemical at a position or all the cell’s 4 neighbors and itself are dead, then this cell’s internal program will not be executed;
    5. Otherwise, executing its program, which will generate its next time chemical and state based on current states and chemicals;
        6. If next state generated is alive, then overwrite chemical at this position with its own generated chemical;
        7. Otherwise, do not touch the chemical at this position;
8. Unless stopping criterion reached go to 2.



Comments on the left figure: The underlined integers in italics represent functions: INV, AND, XOR and ADD are function 5, 6, 7, 9 respectively. The other 3 integers in each group encode the inputs to this molecule. This circuit has a maximum of 6 inputs, connecting to the two molecules in column 1 (left two), although in this sample circuit only 4 inputs are exactly utilized. The output is 7, so the INV gate (encoded by the 4 grey digits) is useless. Bold integers are exactly used inputs to the molecules (remaining digits are just ignored)

**Fig. 4.** A sample digital circuit

## 2.4 Chemical Diffusion

Previous experiments [2] suggest that chemicals are indispensable in order to achieve a robust solution. Chemical diffusion is the key issue which makes it such a significant component of this model: cells have a means to send long-distance messages.

The chemical diffusion rule used here is almost the same as in [2], except that there are only 4 immediate neighbors, not 8. So the rule is as follows:

$$(C_{ij})_{t+1} = \frac{1}{2}(C_{ij})_t + \frac{1}{8}\sum_{k,l\in N}(C_{kl})_t \ . \quad (1)$$

Let $N$ denote all the 4 immediate neighbors of position $(i, j)$ with neighboring position $(k, l)$, the chemical at this position at the new time step is given by (1).

## 3 Evolutionary Algorithm

Inspired by D. Levi's HereBoy [5], a new EA as follows was constructed:

1. Randomly initiate $n$ individuals;
2. Evaluate all of them;
3. Mutate every individual once to generate $n$ offspring; (Mutation rate $p_m$ is fitness related, described below)
4. Evaluate all offspring;
5. If an offspring is better than its parent, replace its parent with it.
6. Else if an offspring is worse than its parent, the offspring has a probability $p_r$ to replace its parent;
7. Else if an offspring's fitness is the same as its parent, the offspring has a constant probability $p_e$ (which is determined as an input parameter beforehand) to replace its parent;
8. Unless stopping criterion reached return to 3.

Adaptive mutation rate has been found to be efficient for hardware evolution [4], thus adaptive mutation rate $p_m$ was employed. It is defined as:

$$p_m = \begin{cases} p_{\min} & (p_c < p_{\min}) \\ p_c & (p_{\min} < p_c < p_{\max}) \\ p_{\max} & (p_c > p_{\max}) \end{cases} \ . \quad (2)$$

$p_c$ is calculated based on the individual's current fitness $f$ and the maximum fitness $f_{max}$, given as:

$$p_c = p'_m(1 - \frac{f}{f_{\max}}) \ . \quad (3)$$

$p_{min}$ , $p_{max}$ and $p'_m$ are parameters defined before evolution. In general, $p_{min}$ multiplied by the number of molecules should be greater than 1 and $p_{max}$ should be equal to or less than 0.5.

The probability $p_r$ of a worse offspring replacing its parent is given by:

$$p_r = p'_r(1 - \frac{f}{f_{\max}}) \ . \quad (4)$$

$p'_r$ is another input parameter. Clearly $p_r$ decreases as the fitness of individual approaches the maximum fitness.

The fitness for a given genotype is the number of cells which exhibit correct states compared with a French flag at growth step *s* or the total correct cells number from step *s1* to *s2*.

## 4 Experiment and Results

A 6x6-cell sized French flag was the intended final pattern.

### 4.1 Parameters

2 bits were used to represent the states of cells. 0 represented a stem (dead) cell without any colors (gray in pictures); 1, 2, 3 were blue, white and red cells respectively.

One 8-bit wide chemical was used. The gene setting for sub-circuit NCG and NSG are *1*x*20@8-20* and *1*x20*@2-20* respectively.

For a sub-circuit, *mols* was defined as the total available number of molecules, for example, *mols* of NCG: *mols* = 1 x 20 = *20*. $p_{min}$, $p_{max}$, $p'_m$ and $p'_r$ were defined as *1/mols*, *0.5*, *0.5* and *1xE-5* respectively.

Initially, all the cells were stem cells except the middle one at (4, 4) which was a red cell (the zygote). The chemicals were all 0 except for at the position of the zygote it was 255 (the maximum chemical available).

The fitness for a perfect individual was the total number of cells in correct colors from growth step 6 and step 9.

Population size is 10. The evolution strategy would not stop unless it found a perfect solution. Once it ceased, it would restart itself to go on searching for another solution (until it was instructed to terminate).

### 4.2 Results

Some extremely robust individuals were found. The growth process of one of them is shown in Fig. 5.

In each image, left part shows the states of the organism, while on the right chemicals are displayed. Top left image in Fig. 5. is the initial states (step 0). Remaining images, column-wise then row-wise, are pictures taken at step 1 to 11 respectively.

It is obvious that step 8 is identical with step 10, and step 9 and step 11 are the same (in regard to both states and chemicals). So after step 8, the organism stabilized itself in a French flag pattern.

Wiping all blue cells to stem cells (after step 9), this organism could recover flawlessly in 5 steps; Changing all blue cells to white, it could recover correctly in 2 steps; Changing all blue cells to red cells, again it could recover accurately in just 2 steps.

As the case stands, altering any whole color area into another one, this organism can recover perfectly in at most 10 steps.

Even when only one live cell left, for example one blue cell at (1, 1), and wiping all chemicals except for the position (1, 1), it could totally recover in 13 steps (As show in Appendix).

In fact, no condition was ever found that this organism could not recover completely.

So once this artificial organism reached a French flag pattern, it was stabilized even when transient faults are inserted into the states and/or chemicals of cells.

## 5 Conclusion and Future Work

The CUs networks in the model proposed has been able to generate structures capable of self-repair, exhibiting extremely high transient fault-tolerant capability. This fault-tolerant ability is not designed consciously, but emerges under the evolution pressures.

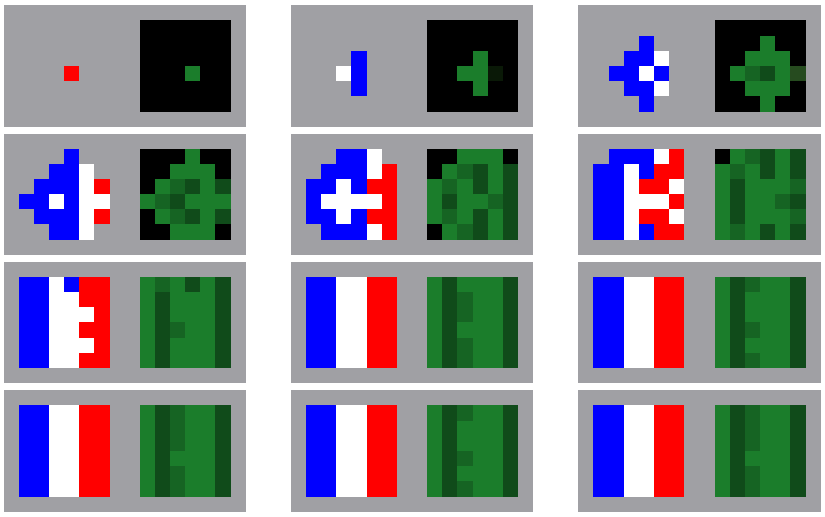


**Fig. 5.** Growth of one of the best individuals found so far

This intrinsic trait could be used to store some crucial information in a real-world system to control external executing modules, such as to store weights in a neural network system.

Larger size flags to find a robust individual similar to the one already found will be evolved. If this is possible, it is reasonable to consider the current model as scaleable.

At present, only the 2-lowest bits of chemicals are used in NSG, ignoring the top 6 bits and new generated chemicals will overwrite previous ones. Effort will be put into the manipulation mechanism of chemicals, such as introducing another kind of chemical behaving as energy and some anisotropic chemicals. It is appreciated if a circuit capable of a particular function is able to grow and mature in a restrained area by its own, rather than just occupying all available resources. This will also be investigated.

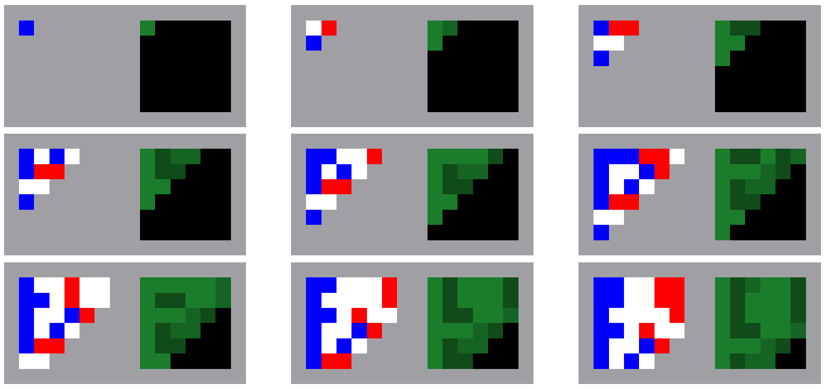
Finally, this model will be enhanced to be FPGA-friendly and a useful application will be implemented on hardware.

## References

1. Lewis Wolpert: Principles of Development 2nd (2002), Chapter 1, Oxford University Press, c2002.
2. Julian F. Miller, "Evolving developmental programs for adaptation, morphogenesis, and self-repair" (2003), Seventh European Conference on Artificial Life, Lecture Notes in Artificial Life, Vol. 2801, pp. 256-265
3. J. Miller and P. Thomson: Cartesian genetic programming. Lecture Notes in Computer Science, Vol. 1802, pp. 121-132, Poli, R., Banzhaf, W., Langdon, W.B., Miller, J. F., Nordin, P., Fogarty, T.C., (Eds.)
4. R. Krohling, Y. Zhou and A. Tyrrell: Evolving FPGA-based robot controller using an evolutionary algorithm, 1st International Conference on Artificial Immune Systems, Canterbury (Sep 2003)
5. D. Levi, "HereBoy: A Fast Evolutionary Algorithm", Proceedings of the 2nd NASA/DoD Evolvable Hardware Workshop, IEEE Computer Society, Los Alamitos, Ca, July 2000.
6. H. Ball and F. Hardy, "Effects and detection of intermittent failures in digital systems" 1969 FJCC, AFIPS Conf. Proc., vol. 35, pp.329-335

## Appendix: Recovery Process of a robust individual

The robust individual found can even recover from only one cell after it matures (at step 8).



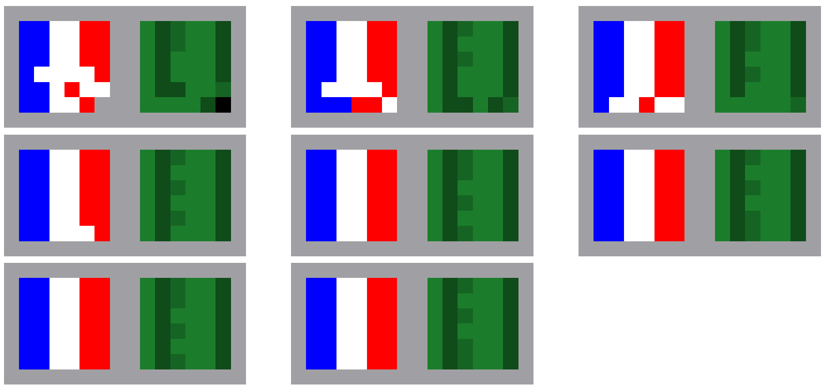


**Fig. 6.** Top left image is at step 9, the state of the organism after transient damages (modification of cell states and chemicals). At step 22, it recover French flag pattern, and stabilized again (can be seen from step 22 – step 25)