Published in Proceedings of ICES98, Lausanne, Switzerland, September, 1998
Lecture Notes in Computer Science 1478, Springer-Verlag, 1998, pp.206-217

MUXTREE Revisited: Embryonics as a Reconfiguration Strategy in Fault-Tolerant Processor Arrays

César Ortega-Sánchez[1] and Andrew Tyrrell

Department of Electronics
University of York
York, YO10 5DD, UK
{cesar, amt}@ohm.york.ac.uk

Abstract: Embryonics' proposal is to construct arrays of processing elements with self-diagnosis and self-reconfiguration abilities able to tolerate the presence of failing cells in the same fashion as natural cellular systems do. Self-healing mechanisms found in nature and the implicit redundancy of cellular architectures constitute the foundations of embryonic systems' fault tolerance properties. It will be shown in this paper how by incorporating the biological concepts of chromosome and gene, the complexity of the MUXTREE embryonic architecture can be simplified, in comparison with the previous version. It is argued that by assuming a broader meaning for the concept of evolution it possible to classify embryonic arrays and other adaptable systems as evolvable.

1 Introduction

When a system (natural or man-made) reaches a certain level of complexity, it becomes very difficult to grasp all of its underlying dynamics, and therefore, it becomes less controllable and less reliable [1]. However, the needs of the modern individual are fulfilled using extremely complex systems. What would our society be without computers, satellites, medicines, mega-software and free market? Complex systems are the foundation of our life-style but they have become very difficult to design. Therefore, it is necessary to look for new methodologies and strategies to deal with complex systems. One approach is the refinement of traditional design techniques, but the techniques themselves are becoming too complex to be considered error-free. Evidently, we have to look somewhere else for the answers [2].

Nature offers to us some remarkable examples of how to deal with complexity and its associated unreliability. For example, the human body is one of the most complex systems ever known. Local failures are common, but the overall function of our organism is highly reliable because of the self-diagnosis and self-healing mechanisms that work ceaselessly throughout our bodies. These mechanisms are the result of millions of years

[1] This work has been partially supported by the Mexican Government under grants CONACYT-111183 and IIE-9611310226

Published in Proceedings of ICES98, Lausanne, Switzerland, September, 1998
Lecture Notes in Computer Science 1478, Springer-Verlag, 1998, pp.206-217

of our gene's evolution. Evolving instead of designing seems to be an attractive alternative when dealing with complexity [3].

During the past few years the work done on evolvable systems has generated some remarkable results [4,5]. Genetic algorithms, neural networks, artificial brains, genetic engineering and evolvable hardware are just a few examples of this novel approach. What is in evolution that is so attractive for hundreds of engineers and scientists? The answer can be found in the characteristics that evolved systems possess, for example, adaptation, auto-regulation and learning.

Embryonics is the design of novel reconfigurable hardware inspired by mechanisms found in nature. We show in this paper how Embryonics can be used to incorporate fault tolerance characteristics, e.g. automatic diagnosis and reconfiguration, into processor arrays by using self-healing like mechanisms.

The paper is organised as follows. Section 2 offers a brief survey on the subjects of evolution and evolvable systems. The Phylogeny-Ontogeny-Epigenesis (POE) model is introduced as a way of classifying evolvable systems. In section 3 a short introduction to processor arrays and their associated fault tolerance techniques is given. The main characteristics of the Embryonics project are explained in section 4. In section 5 a particular embryonic architecture called MUXTREE is analysed and possible improvements are proposed. An example of how to implement a circuit using embryonic arrays is given in section 6. Conclusion are given in section 7.

2 Evolution and Evolvable Systems

To try to find a universal definition for evolution seems to be a fruitless search. The Oxford Dictionary of Current English defines evolution as "origination of species by development from earlier forms; gradual development of phenomenon, organism, etc." [6]. The first definition restricts evolution to natural (or artificial) selection; the second, uses the term as a synonym for development. Evolution, in a broad sense, is much more than that.

Anyone working on the field could give a personal definition for evolution. The term has different meanings for biologists, engineers, geneticists, software developers and priests. When a term is so difficult (or easy) to define, one way of explaining it is describing the properties of the objects possessing such attribute. Although it is not possible to achieve a general agreement on the definition of evolution, it is clear that evolvable systems are: adaptive, fault-tolerant, dynamic, complex in structure and behaviour, dependant on initial conditions (chaotic), capable of learning and made out of interacting simpler units. The list is not exhaustive but, incomplete as it is, it manifests a feature characteristic of evolution and evolvable systems: Change.

We perceive evolution as the continuous change of state in adaptive complex systems. Planetary systems, species and Economics although different in essence, all share the common attribute of perpetual change. Evolution is a sensory experience. It is us who, in an attempt to understand and predict the behaviour of such phenomena, try to find some coherent transition from one state to the next one. Change is the only objective attribute of dynamic complex systems, the rest is subjective interpretation.

Published in Proceedings of ICES98, Lausanne, Switzerland, September, 1998
Lecture Notes in Computer Science 1478, Springer-Verlag, 1998, pp.206-217

It is possible now to state that evolution, in a broader sense, is the spontaneous and purposeful change of state in a dynamic complex system as a response to changes in the environment. Purpose is the subjective quality that we, as observers of the system, attribute to evolvable systems.

2.1 The POE model

Sánchez et al. [7], proposed the Phylogeny-Ontogeny-Epigenesis model (POE model) as a framework to represent the three levels of organisation that can be distinguished in living beings.

Phylogeny embraces the evolution of species when genes are passed from one generation to the next. This is the kind of evolution usually associated with natural selection, where infrequent errors occurring during the copy of genes (mutations) originate new traits on the species. This traits, in some cases, allow the species to better adapt themselves to changes in the environment. Phylogeny presents evolution as species adaptation.

Ontogeny refers to the evolution (development, change of state) of multicellular organisms during their embryonic phase. When reproduction takes place, the new individuals are formed out of a single cell (the fertilised egg). During the following weeks after conception, the mother cell and all its offsprings copy themselves continuously following the "instructions" stored in their DNA (the genome). During this reproductive process cells differentiate to shape the different tissues, organs and limbs that characterise a complete healthy individual of a particular species. Ontogeny is evolution as embryonic development.

Epigenesis is the evolution of individuals' ability to respond to changes in the environment through learning. After birth every individual must adapt to the environment (society, geography, historical context) on which he/she has to live. To achieve adaptation every individual is born with a set of systems defined by the genome (nervous system, immune system and endocrine system), which suffer modifications when interacting with the outside world. Hence, epigenesis is evolution as learning. The POE model can be represented as three orthogonal axis, as shown in figure 1.

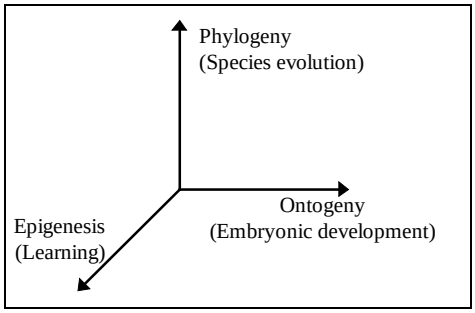

Fig. 1 The POE model for evolvable systems

Published in Proceedings of ICES98, Lausanne, Switzerland, September, 1998
Lecture Notes in Computer Science 1478, Springer-Verlag, 1998, pp.206-217

Human-made evolvable systems can be classified using the POE model. The work on genetic algorithms and evolvable hardware can be placed on the phylogeny axis. Research on artificial intelligence and learning systems can be situated in the epigenesis axis. Works on self-reproducing cellular automata such as the Embryonics project would find a place in the ontogeny axis.

3 Processor Arrays

Processor arrays are a paradigm of computer architecture where multiple identical processing elements (cells) are interconnected in a regular pattern so that the processing power of the whole system (ideally) grows linearly with the number of processors. Communications are strictly local, i.e. each processor only communicates with its nearest neighbours. Examples of this kind of systems are:

Cellular Automata (CA).- In CAs the state of a cell is defined by the state of its nearest neighbours. There is a finite (usually small) number of possible states for a cell. The transition from one state to the next at discrete time steps in all the cells is what determines the array's behaviour and applications. CAs have been used for studying self-reproduction [8], and modelling of dynamic complex systems [9,10].

Systolic Arrays.- In systolic arrays the processing elements are designed to match a particular algorithm. Systolic means that synchronous pipelined computations take place along all dimensions of the array and result in very high computational throughput. The main applications for such computing machines are found in the fields of signal and image processing, pattern recognition, matrix arithmetic and graph algorithms [11].

Wavefront Arrays.- Wavefront arrays combine the systolic pipelining principle with the dataflow computing concept. They are data-driven and do not use a global clock, therefore, the requirement for correct timing in the systolic array is replaced by a requirement for correct sequencing in the wavefront array [12]. The applications of this architecture are practically the same as for systolic arrays.

Parallel structures, like those mentioned above, are good candidates for being implemented in silicon because of their regularity and relative simplicity in the processing unit. In recent years the idea of parallel computers on a chip has become feasible thanks to the advances in VLSI and WSI technologies. Nevertheless, production yields of VLSI circuits is far from being optimum; therefore, reconfiguration techniques have been explored for the past few years in order to provide VLSI processor arrays with fault tolerance.

3.1 Fault Tolerance Techniques in Processor Arrays

All fault tolerance techniques for hardware systems rely on the use of spare components to substitute failing elements. In the past, the cost associated with this redundancy has prevented the widespread use of fault-tolerant hardware. However, in the case of VLSI processor arrays, redundancy comes for free because not all available cells in the array are used on every application.

Published in Proceedings of ICES98, Lausanne, Switzerland, September, 1998
Lecture Notes in Computer Science 1478, Springer-Verlag, 1998, pp.206-217

Fault tolerance in processor arrays implies the mapping of a logical array into a physical non-faulty array, i.e. every logical cell must have a correspondent physical cell [13]. When faults arise, a mechanism must be provided for reconfiguring the physical array such that the logical array can still be represented by the remaining non-faulty cells. All reconfiguring mechanisms are based on one of two types of redundancy: Time redundancy or hardware redundancy [14].

In time redundancy the tasks performed by faulty cells are distributed among its neighbours. In this scheme the application must allow a degradation in performance. When reconfiguration occurs, processors dedicate some time performing its own tasks and some performing faulty cells functions. Interconnection is a key issue. Besides, the algorithm being executed must be flexible enough so as to allow a simple and flexible division of tasks.

In hardware redundancy physical spare cells and links are used to replace the faulty ones. Therefore, reconfiguring algorithms must optimise the use of spares. In the ideal case a processor array with N spares must be able to tolerate N faulty cells; but, in practice, limitations on the interconnection capabilities of each cell prevents this goal from being achieved.

Most of hardware redundancy reconfiguration techniques rely on complex algorithms to re-assign physical resources to the elements of the logical array. In most cases these algorithms are executed by a central processor which also performs diagnosis functions and co-ordinates the reconfiguration of the physical array [15]. This approach has demonstrated to be effective, but its centralised nature makes it prone to collapse if the processor in charge of the fault tolerance functions fails.

An alternative approach is to distribute the diagnosis and reconfiguration mechanisms among all the cells in the array. In this way no central agent is necessary and the time response of the system improves. This mechanism resembles that found in natural cellular systems.

4 Embryology + Electronics = Embryonics

Embryonics was firstly proposed by Mange and Marchal [16,17] as a new family of field programmable gate arrays (FPGAs) inspired by the mechanism sustaining the development of multicellular organisms in nature.

4.1 The Central Dogma

The DNA is a ribbon of 2 billion characters which encode the ensemble of the genetic inheritance of the individual and, at the same time, the instructions for the construction and the operation of the complete organism. In this sense DNA can be both information and physical medium.

In any living being, every one of its constituent cells performs the same basic operation regardless of the particular function it is involved with, i.e. each cell interprets the DNA strand allocated in its nucleus to produce the proteins needed for the survival of

the organism. Proteins are particular sequences of amino acids; such sequences are stored in the DNA as successions of nucleotide triplets (codons).

Protein synthesis imply two mechanisms: transcription and translation of the DNA. During transcription, the sequence stored in the DNA is copied by the enzyme RNA polymerase into messenger RNA (mRNA). During translation, mRNA is bond to ribosomes inside the cell where transfer RNA (tRNA) carrying amino acids are attached to the mRNA. The ribosome catalyses the bond between amino acids to build a molecule of the corresponding protein. When a cell reproduces, the offspring get a copy of its mother's DNA so that the complete process can be ceaselessly repeated. The flow of information from DNA to protein and from DNA of the parent to DNA of the offspring is known as the central dogma [18].

Although the DNA is identical in all the cells, only part of the strand is interpreted. Which part or parts of the DNA are interpreted will depend on the physical location of the cell with respect to its neighbours [19,20]. Figure 2 represents the way DNA's information is organised. Arrows indicate the direction in which complexity increases, e.g. a set of nucleotides forms a codon, etc.

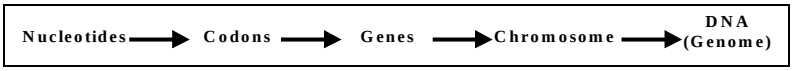

Fig. 2 Structure of DNA

The aim of Embryonics is to transport these basic structure to the 2-dimensional world of cellular arrays using specifically designed FPGAs as building blocks. An equivalent representation of the architecture of field-programmable processor arrays is shown in figure 3.

Fig. 3 Structure of a field-programmable processor array

4.2 Embryonics Architecture

Similarly to natural cellular systems, every one of the embryonic array's cells performs the same basic operation independently of the particular logic function it is involved with, i.e. each cell interprets one of the configuration registers allocated in its memory to perform the logic operations needed for the correct implementation of the system's specification. Which configuration register is selected will depend on the co-ordinates of the cell determined by those of its neighbours. Embryonic cellular arrays share the following properties with their biological counterparts [21]: Multicellular organisation (the logic blocks of an FPGA), cellular differentiation (every cell has a unique set of co-ordinates) and cellular division (every cell is configured by only one configuration register). Figure 4 shows the architecture of a generic embryonic system.

Published in Proceedings of ICES98, Lausanne, Switzerland, September, 1998
Lecture Notes in Computer Science 1478, Springer-Verlag, 1998, pp.206-217

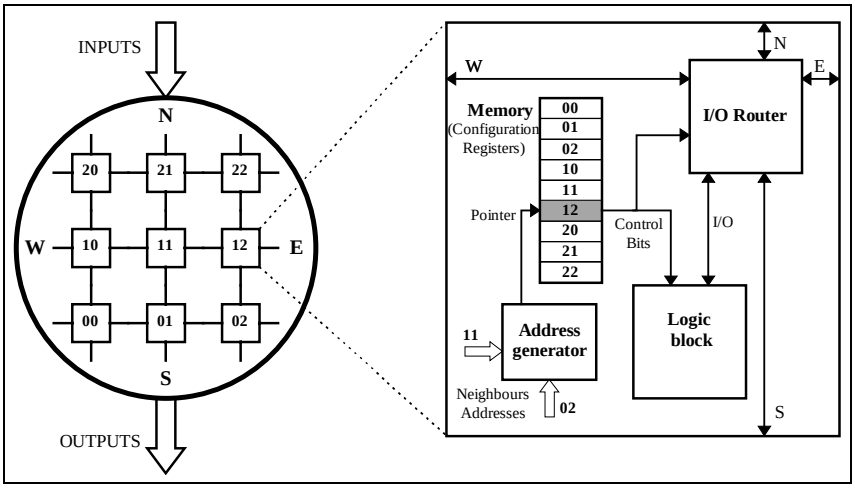

Fig. 4 Basic Components of an Embryonic System

The address generator assigns to each cell an individual set of co-ordinates, which depend exclusively on the co-ordinates of the nearest south and west neighbours. The logic block is controlled by a configuration register which is selected by the corresponding co-ordinates. When a fault is self-detected by a cell, it becomes transparent for the calculation of co-ordinates allowing another cell to take its co-ordinates and therefore its function. Digital data are transmitted from one cell to its neighbours through a North-East-West-South (NEWS) connection. The I/O router block allows the spread of information over all the array. This block is controlled by one section of the corresponding configuration register.

The architecture shown in figure 4 presents the following advantages:

- It is highly regular, which simplifies its implementation on silicon.
- The actual function of the logic block is independent from the function of the remaining blocks. This modularity can be exploited to produce a family of embryonic FPGAs, each member offering either a particular logic function, e.g. a binary selection function [21] or a microprogrammed architecture [22].
- Provided the architecture of the logic block is kept simple, it would be possible to implement built-in self test (BIST) logic to provide self-diagnosis without excessively incrementing the silicon area [21,23].

5 The MUXTREE Revisited

5.1 General Architecture

MUXTREE designates a particular implementation of embryonic FPGA where the processing element of the basic cell (logic block in figure 4), is a selector or multiplexer [21,24]. Multiplexers have the particular characteristic of being able to implement any

node from an ordered binary decision diagram (OBDD), which in turn can represent any combinatorial or sequential logic function [25,26]. Therefore, the resulting architecture is ideal to be implemented as an FPGA. Nevertheless, the MUXTREE architecture, as shown in figure 4, has the following drawbacks:

- Each cell must be able to store the configuration registers of all the cells in the array. For example, in a 32x32 array, each cell must have 1,024 configuration registers, from which only one will be used. This level of redundancy prevents a practical and reliable implementation of the system.
- Depending on the reconfiguration technique chosen (row elimination, column elimination, neighbour substitution, etc.), only one of the co-ordinates is re-calculated when a fail is detected, therefore, it is possible to simplify the address generator block (see Fig. 6).

To overcome the first problem we propose a new architecture for the memory subsystem inspired by the concept of chromosome. The DNA strand is not a single double helix as the general concept might suggest, but it is divided into a number of sub-units called chromosomes.

In the same way, the genome or configuration program for an embryonic FPGA can be divided into smaller parts in order to concentrate the information closer to where it will be needed. For example, if the reconfiguration strategy followed is row replacement, then it would be convenient to divide the genome by columns because a failing cell can only be replaced by another on the same column. Figure 5 shows an example of the memory subsystem simplified by the chromosome analogy. Numbers in bold are the co-ordinates selecting the configuration register on each cell.

This chromosome approach reduces the number of configuration registers per cell to

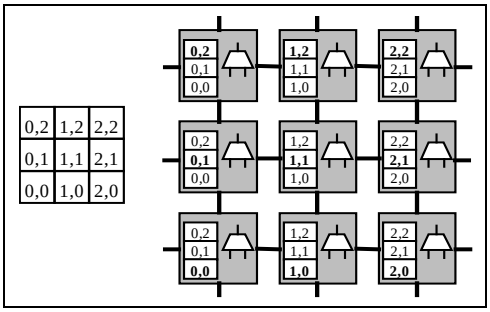

Fig. 5 Genome memory and its distribution in an embryonic array

the number needed in the original design divided by the number of columns. In the case of square arrays every cell contains only the square root of the number of registers needed in the original design, which represents substantial savings on silicon area.

Another characteristic of the previous version of the MUXTREE is that when a cell is self-detected faulty, it needed to broadcast a non-OK signal to all the cells in the same row and column so that they can recalculate their new co-ordinates and consequently, select a new configuration register. But close inspection of figure 5 reveals that when reconfiguration takes place the column co-ordinate remains constant. Therefore, it is possible to simplify the reconfiguration mechanism by calculating row co-ordinates only

and propagating non-OK signals exclusively to neighbours in the same row. Figure 6 illustrates these mechanisms.

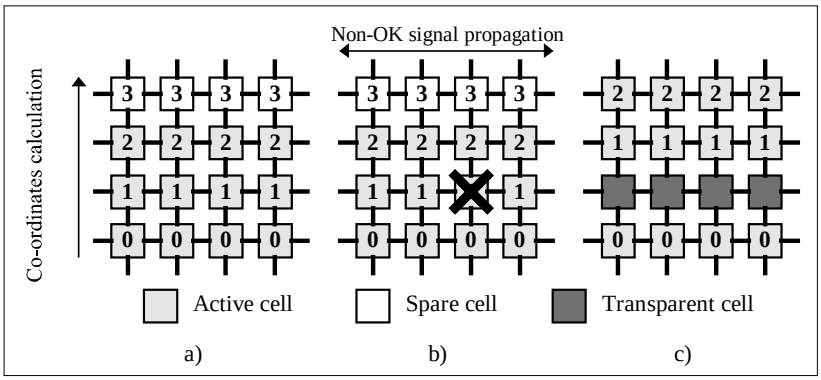

Fig. 6 Embryonic array
a) Co-ordinates calculation b) Fault detection c) Reconfiguration mechanism

One advantage of the proposed scheme is that it becomes possible to re-load independent chromosomes (configuration registers of a particular column), without affecting the others. This characteristic could be used to evolve embryonic arrays using genetic algorithms. The chromosomes in a genome would be the population exposed to crossover and mutation. Routing of circuits could be achieved following this approach.

5.2 Synthesis Method

The design cycle for implementing a system based on embryonic arrays is as follows:

1. Express the function to be implemented as a set of Boolean equations.
2. Obtain the OBDD for each one of the equations.
3. Replace each node of the OBDDs with a 2-1 multiplexer, using the node's variable as selection input and the outputs of the node as inputs to the multiplexer.
4. Map the network of multiplexers into an array of embryonic cells, assigning one multiplexer to each cell. This has been experienced to be the most difficult step due to the finite connectivity of the cells.
5. Obtain a configuration register for each cell in the array. The set of all configuration registers conforms the genome of the application.

After power-up all the cells in the array calculate their co-ordinates. At the same time external circuitry downloads the genome into the array column by column. The memory block was designed in such a way that chromosomes are downloaded to all the cells in the corresponding column at the same time, making this process faster.

Published in Proceedings of ICES98, Lausanne, Switzerland, September, 1998
Lecture Notes in Computer Science 1478, Springer-Verlag, 1998, pp.206-217

6 Example

To illustrate embryonic arrays' properties the design of a programmable frequency divider is presented next. Figure 7 shows the circuit's block diagram. It is composed by a 3-bit selector which latches either the division factor **n,** or the next state of a 3-bit down-counter, according to the output of a zero detector. In this way, a 1-cycle wide pulse will be generated every **n** cycles of **F**. The output of the circuit is taken from the output of the zero detector. It will be high during one cycle of **F** when the down-counter reaches the 000 state.

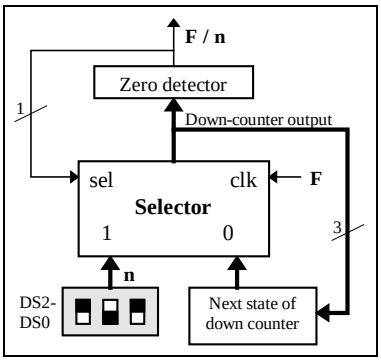

Fig. 7 Programmable frequency divider

Figure 8 shows the hardware implementation of the circuit shown in figure 7. Every multiplexer corresponds to a node in the corresponding OBDD. **A**,**B**,**C** are the outputs of the 3-bit down counter, **C** being the most significant bit. Multiplexers 1, 2 and 3 implement the selector block. These multiplexers update their outputs on the rising edge of **F**. **DS2**, **DS1** and **DS0** are used to set the value of **n**.

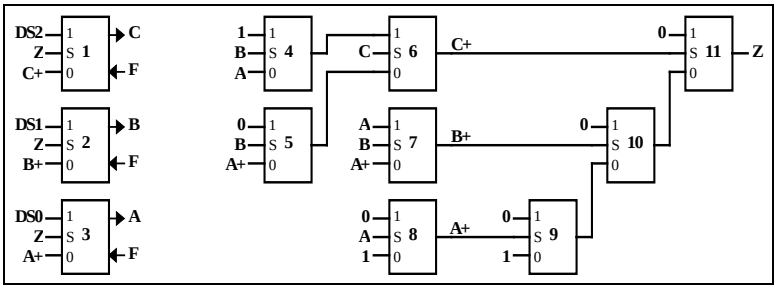

Fig. 8 Hardware implementation of circuit in figure 7.

Figure 9 shows how the circuits in figure 8 were mapped into an embryonic array. The numbers on each cell correspond with the numbers assigned to the multiplexers. Cells labelled **S** are spare cells, two rows for this example. Cells labelled **R** are routing cells. Routing cells are needed because every cell has direct connections only with its cardinal neighbours.

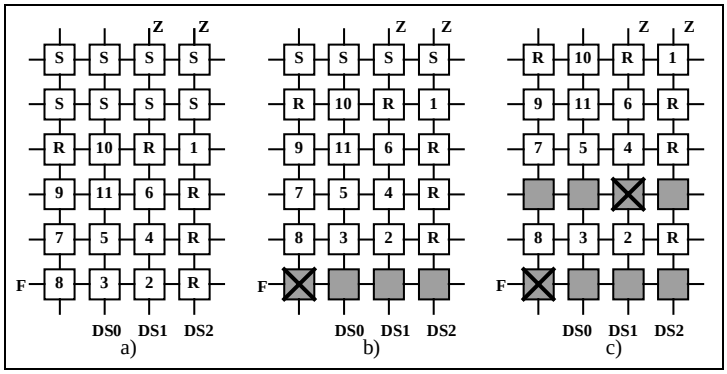

Fig. 9 Frequency divider implemented in embryonic array.
a) Without fails, b) with one faulty cell, c) with two faulty cells

Figure 10 shows the simulation results obtained for the frequency divider. Labels correspond with those of figure 8. **OK4** and **OK8** simulate fails in cells 4 and 8 respectively. Notice that when **OK** signals go to logic 0, there is a small time interval on which the output of the circuit remains constant, just before returning to normal behaviour. This is due to the reconfiguration process being carried out.

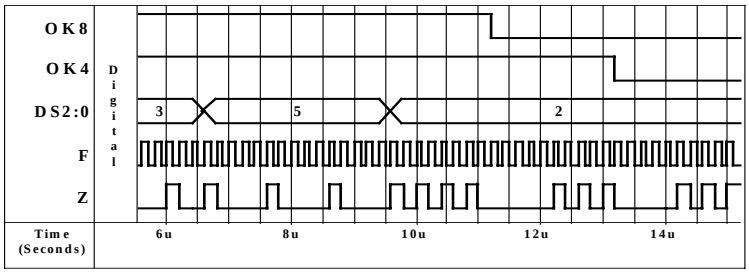

Fig. 10 Functional simulation of frequency divider

7 Conclusions

Evolution is perceived by the change of state in a dynamic complex system. In this broader sense, we can apply the attribute “evolvable” to numerous man-made systems whose configuration or response is influenced by the environment. In this way genetic algorithms, neural networks, artificial intelligence projects and Embryonics can all be classified as evolvable systems.

Embryonic arrays have demonstrated to be an attractive alternative for the design of biologically-inspired field programmable processor arrays with fault tolerance properties. We still are in the early stages of the Embryonics project, but the results obtained so far are encouraging to keep investigating on the application of biological concepts to the design of fault-tolerant engineering systems.

Published in Proceedings of ICES98, Lausanne, Switzerland, September, 1998
Lecture Notes in Computer Science 1478, Springer-Verlag, 1998, pp.206-217

References

1. Paul G.: Beyond Humanity: CyberEvolution and future minds, Charles River Media, 1996
2. Avizienis A.: “Toward Systematic Design of Fault-Tolerant Systems”, IEEE Computer, April, 1997, Computer Society Press, pp. 51-58
3. Kelly K.: Out of Control: The new Biology of machines, Fourth State-London, 1994
4. Sánchez E. et al.: (Eds.), Towards Evolvable Hardware: The evolutionary engineering approach, LNCS 1062, Springer-Verlag, 1996
5. Higuchi T. et al. (eds.): Evolvable Systems: From Biology to Hardware, LNCS, Springer-Verlag, 1997
6. The Oxford Dictionary of Current English, Oxford University Press, 1990
7. Sánchez E. et al.: “Phylogeny, Ontogeny and Epigenesis: three sources of biological inspiration for softening hardware”, in Higuchi T. et al. (eds.), Evolvable Systems: From Biology to Hardware, Springer-Verlag, 1997
8. Langton C.: “Self-reproduction in Cellular Automata”, Physica 10D, 1984, pp.135-144
9. Burks C.: “Towards Modelling DNA as Automata”, Physica 10D, 1984, pp.157-167
10. Grassberger P.: “Chaos and Diffusion in Deterministic Cellular Automata”, Physica 10D, 1984, pp. 145-156
11. Fortes J. et al.: “Systolic Arrays- From Concept to Implementation”, IEEE Computer, July, 1987, pp.12-17
12. Kung S. et al.: “Wavefront Array Processors- Concept to Implementation”, IEEE Computer, July, 1987, pp. 18-33
13. Grosspietsch K.: “Fault Tolerance in Highly Parallel Hardware Systems”, IEEE Micro, Feb. 1994, pp.60-68
14. Chean M. et al.: “A Taxonomy of Reconfiguration Techniques for Fault-Tolerant Processor Arrays”, Computer, January, 1990, pp. 55-69
15. Fortes J. et al.: “Gracefully Degradable Processor Arrays”, Trans. on Computers, Vol.34-11, November, 1985, pp.1033-1043
16. Mange D. et al.: “Embryonics: A new family of coarse-grained FPGA with self-repair and self-reproduction properties”, in Sanchez E. (Ed.), Towards Evolvable Hardware, LNCS 1062, Springer-Verlag, 1996, pp.197-220
17. Marchal P.: “Embryonics: The birth of synthetic life”, in Sanchez E. (Ed.), Towards Evolvable Hardware, LNCS 1062, Springer-Verlag, 1996, pp.166-196
18. Murrell J.C. and Roberts L.M. (Eds.): Understanding Genetic Engineering, Ellis Horwood, Great Britain, 1989
19. Nüsslein-Volhard C.: “Gradients that Organize Embryo Development”, Scientific American, August, 1996, pp.38-43
20. Gerhart J. and Kirschner M., Cells, Embryos and Evolution, Blackwell Science, 1997
21. Mange D. and Tomassini M. (Eds.), Bio-Inspired Computing Machines, Presses Polytechniques et Universitaires Romandes, Switzerland, 1998
22. Mange D., Madon D., Stauffer A. and Tempesti G.: “Von Neumann Revisited: A Turing Machine with Self-Repair and Self-Reproduction Properties”, Robotics and Autonomous Systems, Vol.22-1, 1997, pp.35-38
23. Lala P.: Fault Tolerance and Fault Testable Hardware Design, Prentice-Hall, 1985
24. Ortega C. and Tyrrell A.: “Design of a Basic Cell to Construct Embryonic Arrays”, IEE Procs. on Computers and Digital Techniques, May, 1998
25. Akers S.: “Binary Decision Diagrams”, IEEE Trans. on Computers, Vol.27-6, June 1978
26. Liaw H. et al.: “On the OBDD-Representation of General Boolean Functions”, IEEE Trans. on Computers, Vol.41-6, June, 1992, pp.661-664