Biologically Inspired Reconfigurable Hardware for Dependable Applications

Cesar Ortega and Andy Tyrrell
Department of Electronics
University of York
York, YO1 5DD, UK
E-mail: (cesar, amt)@ohm.york.ac.uk

1. Introduction

Practical experience has demonstrated that the goal of building fault-free systems, although attractive, is impossible to achieve. Hardware deteriorates with time, and software has become so complex that design faults are difficult, if not impossible, to avoid. Hence, the more viable alternative for applications requiring high levels of dependability is to implement systems capable of tolerating faults, i.e. Fault-Tolerant Systems.

The complexity of problems solved by computers is increasing with time, so it is with the architectures of computers themselves. Therefore, the risk of failure is greater now than when computers and the problems they solved were simple. Complexity implies unreliability. Hence, 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.

Nature offers us some remarkable examples of how to deal with complexity and its associated unreliability. The human body possibly is one of the most complex systems ever known. Failures are not rare, but the overall function is highly reliable because of self-diagnosis and self-healing mechanisms that work ceaselessly throughout our bodies. To borrow the main principles that sustain this mechanisms and applying them to the design of electronic systems, could result in a new approach for the design of fault-tolerant systems [2].

Of particular interest for our research are the fault tolerance attributes of massively parallel processing networks or processor arrays, like cellular automata or artificial neural networks. In this approach the "knowledge" is distributed throughout multiple processing elements, therefore, if one or a relatively small subset of the processors fail, the overall functionality can be maintained [4] [7]. For our purposes a processor arrays will be considered as a special case of cellular automata, therefore we will use the terms processor and cell interchangeably.

Self-healing mechanisms found in nature and the implicit redundancy in processor arrays constitute the inspiration for the presented work. Embryonics' proposal is to construct processor arrays with self-diagnosis and self-reconfiguration abilities, able to tolerate the presence of failing cells, in the same fashion as natural cellular systems do. The embryonics concept was initially presented Mange and Marchal in [9].

2. The Present Approach to Fault Tolerance in Processor Arrays

Fault tolerance in processor arrays implies the mapping of a logical array into a physical non-faulty array, i.e., every logical cell in the logical array, must have a correspondent physical cell in the physical array. 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 [5].

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. Again, 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. Our research is focused on this type of redundancy.

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 [6] [12] [8]. 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 algorithms among all the cells in the array. In this way no central agent is necessary and consequently the reliability of the system increases. This mechanism resembles that found in natural cellular systems.

Our proposal is to embed some of the distinctive characteristics of biological cellular systems into the design of processor arrays.

3. The Biological Approach to Fault Tolerance

A human being consists of approximately 60 trillion ($60x10^{12}$) cells. At each instant, in each of these 60 trillion cells, the genome, a ribbon of 2 billion characters, is decoded to produce the proteins needed for the survival of the organism. This genome contains the ensemble of the genetic inheritance of the individual and, at the same time, the instructions for both the construction and the operation of the organism. The parallel execution of 60 trillion genomes in as many cells occurs ceaselessly from the conception to the death of the individual. Faults are rare and, in the majority of cases, successfully detected and repaired [9]. Which part of the DNA is interpreted will depend on the physical location of the cell with respect to its neighbours.

Embryonics is inspired by the basic processes of molecular biology. By adopting the features of biological cellular organisation, and by transposing them to the two-dimensional world of cellular arrays, it can be shown that properties unique to the living world, such as self-reproduction and self-repair, can also be applied to integrated circuits. Figure 1 shows the basic architecture of an embryonic system:

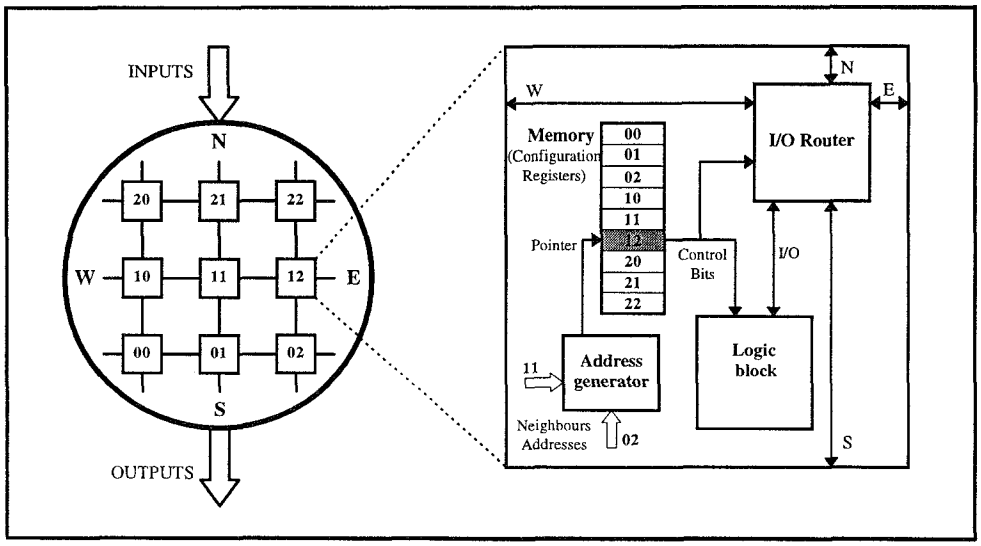

Fig. 1 Basic Components of an Embryonic System

This architecture presents the following advantages:

- It is highly regular, which simplifies its implementation on silicon.
- The actual function of the logic block can be changed or substituted by others without affecting the function of the remaining blocks. This modularity could be exploited to produce a family of Embryonic FPGAs, each member offering a particular logic function, e.g. ALU, MUX, DEC, etc.
- The simplicity of blocks' architecture allows the implementation of built-in self test (BIST) logic to provide self-diagnosis without excessively incrementing the silicon area. A full description of the self-diagnosis system can be found in [13]. If one of the cells fails, it becomes transparent for the calculation of co-ordinates allowing another cell to take its co-ordinates and therefore its function. The block corresponding to error-detection and error-handling is not shown in figure 1.

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.

We designed the reconfiguration logic so that when a cell auto-diagnoses faulty, the row for the corresponding cell is eliminated and replaced by a spare row. Co-ordinates for the cells above the eliminated row are recalculated and new configuration registers are selected accordingly. This strategy is far from being optimal

with respect to the use of spare resources, but the logic needed to implement it is simple and enough to probe the fault tolerance properties of embryonic arrays. A detailed description of the cell can be found in [10].

4. Example

To probe the embryonics thesis we designed a simple application. Because we are more concerned with reconfiguration issues rather than with the processing capabilities of the array, we chose a multiplexer to be the functional part of the basic cell. Besides, the multiplexer has the particular advantage of being able to implement any node from a binary decision diagram (BDD), which in turn can represent any combinatorial or sequential logic function [1]. Therefore, the architecture presented is ideal to be implemented as a field programmable gate array (FPGA) [13].

Our goal was to design a programmable frequency divider. Figure 2 shows the circuit as composed by a 3-bit selector which latches, according to the output of a zero detector, either the division factor **n** or the next state of a 3-bit down-counter. In this way, a 1-cycle wide pulse will be generated every **n** cycles of **F**.

The circuit's output is taken from the output of the zero detector. It will be 1 during one cycle of **F** when the down-counter reaches the 000 state. Figure 3 shows the binary decision diagrams for the combinatorial down-counter and the zero detector. A,B,C are the outputs of the 3-bit down counter, being C the most significant bit.

The diagrams must be evaluated from top to bottom. If the variable on each node takes the value 0, then the branch with a broken line is selected. Otherwise, if the variable takes the value 1, then the branch represented with a solid line is followed. The final nodes on a BDD represent the output of the function for a particular set of input values. A good tutorial on BDDs can be found in [3] and [11].

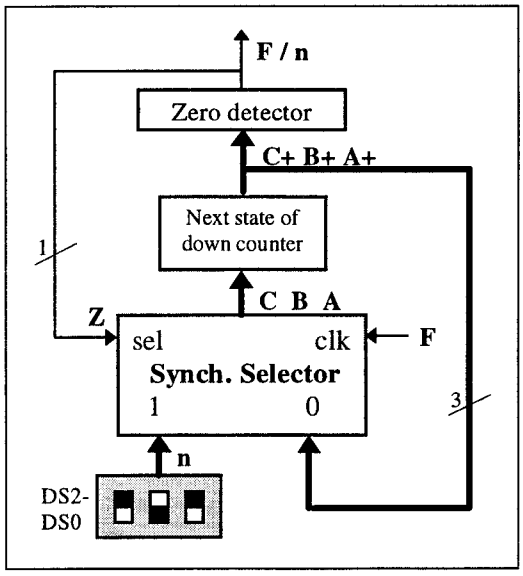

Fig. 2 Programmable frequency divider

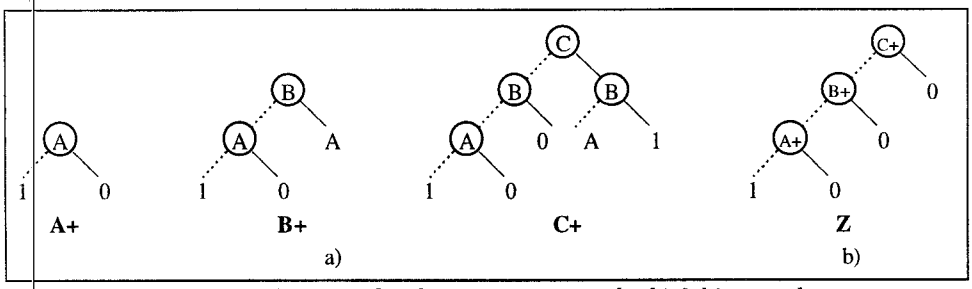

Fig. 3 BDDs for a) 3-bit down-counter; and b) 3-bit zero detector

Figure 4 shows the hardware implementation of the BDDs shown in figure 3. Note that the BDD for **A+** is repeated in the diagrams for **B+** and **C+**, therefore the final circuit can be simplified. Multiplexers 1, 2 and 3 implement the selector block in figure 2. These multiplexers operate in synchronous mode, i.e., their outputs will be updated on the rising edge of **F**. **DS2**, **DS1** and **DS0** are used to set the value of **n**.

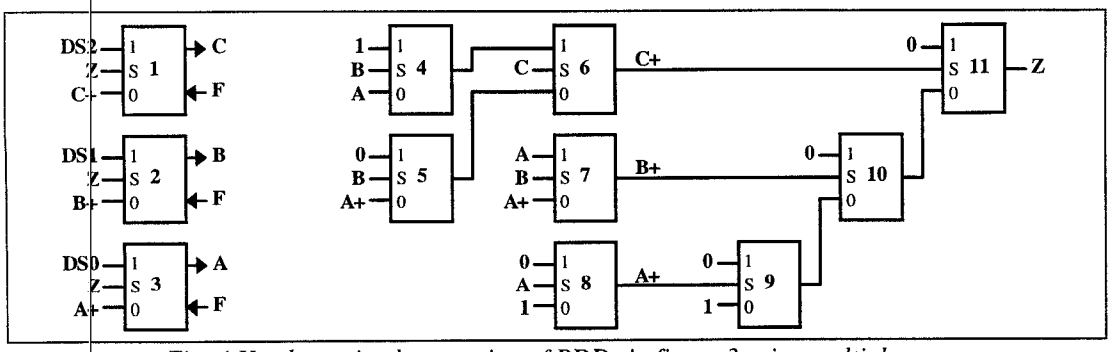

Fig. 4 Hardware implementation of BDDs in figure 3 using multiplexers.

The circuits in figure 4 were accommodated by hand into an embryonic array. Figure 5a shows the final distribution of multiplexers. The numbers on each cell correspond with the numbers assigned to the multiplexers in figure 4. 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 to its cardinal neighbours.

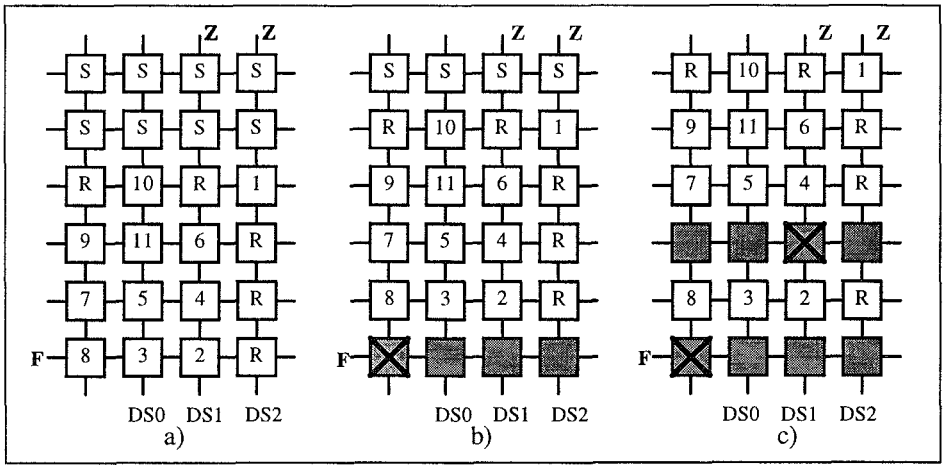

Fig. 5 Frequency divider implemented in embryonic array.
a) Without fails, b) with one faulty cell, c) with two faulty cells

Figures 5b and 5c show the reconfiguration process when one and two cells become faulty, respectively. Remember that during the reconfiguration process the cells' co-ordinates are re-calculated so that every cell selects the configuration register of the cell it is substituting.

5. Conclusions

A novel paradigm for designing fault-tolerant processor arrays inspired by biological processes has been presented. These ideas can be applied to the design of fault-tolerant FPGAs, or for improvement of yield in the fabrication of WSI systems. Embryonics is a nascent discipline, therefore much research must be done to investigate in depth the fault-tolerant properties of these systems. Our approach is coherent with a recent uproar about the application of biological concepts to the solution of engineering problems, e.g., evolutionary computing, evolvable hardware, and genetic algorithms.

References

[1] Akers S., "Binary Decision Diagrams", IEEE Trans. on Computers, Vol.27-6, June 1978

[2] Avizienis A., "Toward Systematic Design of Fault-Tolerant Systems", Computer, April, 1997, pp. 51-58

[3] Bryant R., "Graph-based Algorithms for Boolean Function Manipulation", IEEE Trans. on Computers, Vol.35-8, Aug.1986, pp.677-691

[4] Chean M. et al., "The Full Use of Suitable Spares (FUSS) Approach to Hardware Reconfiguration for Fault-Tolerant Processor Arrays, IEEE Trans. on Computers, Vol.39-4, April 1990

[5] Chean M. et al., "A Taxonomy of Reconfiguration Techniques for Fault-Tolerant Processor Arrays", Computer, January, 1990, pp. 55-69

[6] Dutt S. et al., "Node-covering, Error-correcting Codes and Multiprocessors with Very High Average Fault Tolerance", IEEE Trans. on Computers, Vol. 46-9, Sept. 1997, pp. 997-1014

[7] Dyer M., "Toward Synthesising Artificial Neural Networks that Exhibit Cooperative Intelligent Behaviour: Some open issues in Artificial Life", in Langton C. (ed.), Artificial Life: an Overview, MIT Press, 1995

[8] Fortes J. et al., "Gracefully Degradable Processor Arrays", Trans. on Computers, Vol.34-11, Nov.1985, pp.1033-1043

[9] 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

[10] Ortega C. and Tyrrell A., "Fault-tolerant Systems: The way Biology does it!", Short contribution in Euromicro 97 Conference, Budapest, Sept. 1997

[11] Rauzy A., "A brief introduction to Binary Decision Diagrams", Journal Européen des Systèmes Automatisés, Hermes, Vol. 30-8, 1996, pp. 1033-1050

[12] Sun K. et al., "Fault-Tolerant Array Processors Using Single-Track Switches", IEEE Trans. on Computers, Vol. 38-4, April, 1989, pp. 501-513

[13] Tempesti G., "A Robust Multiplexer-based FPGA Inspired by Biological Systems", Special Issue of JSA on Dependable Parallel Computer Systems, Feb. 1997

