# Survivability of Embryonic Memories: Analysis and Design Principles

Lucian Prodan Mihai Udrescu Mircea Vladutiu
Advanced Computing Systems and Architectures Laboratory
"Politehnica" University, Computer Department
2 V. Parvan Blvd., 300223 Timisoara, Romania
www.acsa.utt.ro
+40-722-664779 +40-723-154989 +40-256-403258
lprodan@cs.utt.ro mudrescu@cs.utt.ro mvlad@cs.utt.ro

## Abstract

*This paper proposes an original approach to the reliability analysis for Embryonics [4], by introducing the accuracy threshold measure, borrowed from fault-tolerant quantum computing theory, as one of the main parameters for our qualitative assessment. The validity of this technique is proven by comparison with the classical reliability results; furthermore, it brings new perspectives on designing reliable embryonic memory structures at both the molecular and the cellular levels. Appropriate design principles are provided on both information encoding (concatenated codes) and storage (fault tolerant memory structures).*

## 1 Introduction

Computer evolution in present days faces a fine paradox, arising from its dual nature: some applications require speed above anything else, while others require highest possible reliability. While each of the requirements has to deliver its best, unfortunately, computers can only *partially* fulfill them, therefore justifying a quest for different and better-suited designs. Inspiration is constantly sought in both their hardware and software designs. Since their beginning, computers were protagonists of the quest for performance. Once the benefits of computing power and technological advances enabled the coming of the space exploration era, a shifting in performance priorities was encouraged. Demands for brute computing force (which appears to have reached somewhat sufficient levels today) began to lose ground to dependability requirements.

As stated by Avižienis *et al.*, dependability can be defined as "*the ability of a system to avoid service failures that are more frequent or more severe than is acceptable*" [1]. It is therefore a synthetic term that involves a list of attributes including reliability, fault tolerance, availability, and others. In the real world, a dependable system would have to operate normally over long periods of time before experiencing any fail (reliability, availability) and even in the presence of faults (fault tolerance). The term "*acceptable*" has an essential meaning within the dependability's definition, setting the upper limits of the damage that can be supported by the system while still remaining functional. This paper will elaborate upon computing the threshold beyond which the damaging effects become no longer acceptable, and, by consequence, no longer recoverable from. All of the above being considered, dependable systems are crucial for applications that prohibit or limit human interventions, such as long term exposure to aggressive (even hostile) or unknown environments. The best examples are long term operating machines, as required by managing deep-underwater/nuclear activities and outer space exploration.

The quest of designing digital systems that offer superior dependability can draw benefits from two distinct sources. The first one has been around since the dawn of times: Nature. Its living elements continuously demonstrate a variety of solutions for achieving robustness in an error-prone, macro-scale environment. There are numerous similarities and differences between artificial, digital computing systems and natural, living beings; although such a thorough analysis is beyond the scope of this paper, it is likely that the field of digital computing could exploit some of the mechanisms implemented by Nature and adapt them to the electronic environment. The result is the new field of bio-inspired computing, a representative example being the Embryonics project [3][16].

Another source of inspiration may be constituted by novel computing paradigms, whose research has already considered dependability-raising techniques. Opposed to natural computing, which takes place in an error-prone, macro-scale environment, the emerging field of quantum computing relies on successful calculus in an error-prone, micro-scale environment. Since errors are (as of yet) intrinsic to quantum systems, a number of techniques were established in order to overcome their damaging effects and deliver consistent results. However, though a variety of

Proceedings of the 2005 NASA/DoD Conference of Evolution Hardware (EH'05)
0-7695-2399-4/05 $ 20.00 **IEEE**

methodologies for estimating dependability parameters have been proposed, they usually remain localized and rarely reach other architectures.

### 1.1 Paper outline

The basics of quantum computing together with the accuracy threshold estimation technique will be revisited in Section 2. After an introduction to the concepts and the architecture of the Embryonics project, Section 3 will detail the process of adapting this technique and present arguments on its relevance to the bio-inspired computing (and, in particular, to Embryonics). Qualitative results will be compared to those provided by the more classical reliability estimations. The suitability of the concatenated coding technique (also inspired from quantum computing) to memory structures in Embryonics will also be discussed, together with the appropriate design principles. Section 4 argues, in qualitative terms (accuracy threshold), over design decisions such as Hamming and concatenated coding. Finally, Section 5 will present our conclusions and thoughts for future work.

## 2 Fault-tolerant quantum computing

Quantum computation uses coherent atomic-scale dynamics [13] and therefore takes place in a microscopic environment. The information storage unit in quantum computing is the quantum bit or *qubit*, which is presented here in *bra-ket* notation [13]. Any qubit $|\psi\rangle = a_0|0\rangle + a_1|1\rangle$ is a normalized vector in a bi-dimensional Hilbert space, with $|0\rangle$ and $|1\rangle$ as the orthonormal basis, and can be seen as a simultaneous superposition of both classical binary values. Parameters $a_0$, $a_1 \in \mathbb{C}$ are called *quantum amplitudes* and represent the square root of the associated measurement probabilities for the superposed states $|0\rangle$ and $|1\rangle$ respectively, with $\|a_0\|^2 + \|a_1\|^2 = 1$. The qubits can be organized in linear structures called *quantum registers*, encoding a superposition of all possible states of a corresponding classical register [6]: $|\psi_r\rangle = \sum_{i=0}^{2^n-1} a_i |i\rangle$; $\sum_{i=0}^{2^n-1} \|a_i\|^2 = 1,\ a_i \in \mathbb{C},\ i \in \mathbb{N}$.

An essential promise of quantum computing is solving in polynomial time problems that have only exponential known classical solutions. However, the transition from classical computing to this new paradigm is far from immediate and beyond the scope of this paper. The new computational environment requires a new set of problems to be solved first [14][15] before any benefits are to be drawn. Dealing with dependability issues constitutes a priority in quantum computing because of its inner faulty nature. Faults are native to the quantum environment since a quantum state cannot be fully isolated from the environment, which continuously attempts to measure it. The state, in turn, decays to one of the basis (classical) states, a phenomenon called *decoherence*. Although they can be classified into 3 categories, namely bit flips, phase shifts and small amplitude errors, faults can all be reduced to bit flips [6][8]. Errors affecting the quantum computing are considered to be uncorrelated, neither in space, nor in time [8], therefore sharing the same model with soft fails induced in digital devices by aggressive radiations.

In order to make accurate quantum computation possible in an error-injecting environment, recovery procedures are required; redundant coding presents a choice of strategies for achieving fault tolerance. However, the recovery process is by itself computational, and therefore vulnerable to errors. In order to ensure a level of fault tolerance that would make the prospect of a quantum computation device feasible, the following questions have to be raised: what is the accuracy threshold that still warrants valid computation? Or, what is the upper bound of the fault frequency that would still allow a successful recovery? These questions were already answered in the quantum context [8][17]; we will however revisit the proposed qualitative assessment since we believe a similar reasoning may also be applied to the Embryonics project [4] and to fault-tolerant digital systems in general.

If the redundant, error-correcting coding allows the correction of $t$ faults, then an uncorrectable fault means that at least $t$+1 faults occur before the recovery process can be finalized. If the probability of a fault affecting the macro-molecular information is $\xi$, then an uncorrectable error will happen with a probability of the order $\xi^{t+1}$ [8][17]. Ideally, choosing a reasonably high value for $t$ can make the probability of an unrecoverable fault as small as desired; however, the complexity of the code shows a steep rise with the value of $t$, with a polynomial function of the form $t^b$, eventually leading to the situation when correcting the data takes so long that the appearance of an unrecoverable event becomes most likely. Then, the block error probability (*BEP*) of $t$+1 errors accumulating in a codeword before the recovery is complete (thus producing an unrecoverable event) will have the form [8]:

$$BEP(t) \sim \left(t^b \xi\right)^{t+1} \quad \textbf{(1)}$$

Minimizing the *BEP* function after parameter $t$ yields:

$$\frac{dBEP(t)}{dt} = 0 \quad \textbf{(2)}$$

which results in:

$$\ln t + \frac{1}{b}\ln \xi + 1 + \frac{1}{t} = 0 \ \Leftrightarrow\ te^{1/t} = e^{-1}\xi^{-1/b} \quad \textbf{(3)}$$

Solving Equation 3 and assuming that $t$ is large [8] gives:

$$t \sim e^{-1}\xi^{-1/b} \quad \textbf{(4)}$$

Substituting this result into Equation 1, the minimum block error probability *MBEP* then becomes of the form [8]:

$$MBEP(\xi) \sim \exp\left(-e^{-1}b\xi^{-1/b}\right) \quad \textbf{(5)}$$

The result for $MBEP(\xi)$ is important with respect to estimating the required accuracy for a reliable computation. If we consider $T$ as the time interval without any unrecoverable error occurring, then:

$$T(\xi) \sim MBEP(\xi) \;\Rightarrow\; T(\xi) \sim \exp\left(\xi^{-1/b}\right) \quad \textbf{(6)}$$

From this equation, $\xi$ can then be extracted under the form:

$$\xi \sim (\ln T)^{-b} \quad \textbf{(7)}$$

For the situation when no codes are used at all, the accuracy decreases as the computation becomes longer and therefore gives:

$$\xi_{NoCodes} \sim T^{-1} \quad \textbf{(8)}$$

Equation 7 provides a qualitative assessment of the computational accuracy threshold with error protecting codes that is clearly superior to the case when no codes are used at all (Equation 8). Due to the lack of standardization when dependability measures are concerned [1], establishing precise values is difficult. However, this qualitative assessment delivers the necessary criteria for a dependability comparison between two functionally identical systems, before and after applying fault tolerance measures.

## 3 Naturally-inspired dependability and the Embryonics project

Natural computation occurs at a macroscopic scale, the environment being subject to dynamic changes, which affect living beings by inducing a variety of wounds and illnesses (faults). Though the natural computation is also error-prone, successful healing and recovery are quite common: in a majority of cases, natural systems continue to carry on their vital functions while their overall functionality levels do not drop abruptly.

With the exception of unicellular organisms (bacteria), multicellular organisms share some key features [4]:

- *multicellular organization* divides the organism into a finite number of *cells*, each accessing the same genetic program;
- *cellular division and differentiation* allow any cell to generate daughter cell(s) with certain features through execution of part(s) of the genome.

A consequence of these features is that each cell is "universal", as it contains the whole of the organism's genetic material, the genome. This enables very flexible redundancy strategies, the living organisms being capable of self-repair (healing) or self-replication (cloning). These two properties, based on a multicellular tissue, are essentially unique to the living world.

The capacity of healing is what gives natural systems their robustness. Fault tolerance is hierarchical, being present at several different levels: redundancy and self-repairing features can be found at molecular level (the DNA contains redundancies and can repair a variety of faults [10]), at the cellular level (cells do replace each other in case of faults, faulty ones die and are eliminated while new ones are grown) and even at higher levels (brain hemispheres, for instance, are known to be able to transfer some functionalities in case of damage). The success of Nature's solutions is proven by the uncountable variety of living beings, with a life span up to several hundreds (for the animal regnum) or even thousands (for the vegetal regnum) of years; furthermore, considering the amounts of time spent for evolving them, they are as close to perfection as possible. This alone makes for a strong argument supporting bio-inspiration in digital computing, an idea enounced in the 1950s by John von Neumann, who may also be considered the pioneer of reliable systems [5].

The Embryonics (from *embryonic electronics*) project made its debut as long-term research aimed at exploring the potential of biologically-inspired mechanisms adapted into digital devices [4]. Rather than achieving a specific goal, the purpose is building novel, massively parallel, computational systems, that implement the key features shared by all multicellular organisms and would also borrow the remarkable robustness present in biological entities.

As a bio-inspired digital platform, the key features shared by all multicellular organisms are present in Embryonics through a quasi-biological hierarchy based on four levels of organization, as shown in Figure 1 [3][4]. The Embryonics project targets applications in which the failure frequency must be very low to be "acceptable".

The upmost level in Embryonics, similar to what is found in nature, is the population level. One step down inside the hierarchy the focus zooms to the population's components. This is the organismic level, and corresponds to individual entities in a variety of functionalities and sizes. Each entity may, however, be further decomposed into smaller, simpler parts, called cells, and then, molecules. A biological organism corresponds in the world of digital systems to a complete computer, a biological cell is equivalent to a processor, and the smallest part in biology, the molecule, may be seen as the smallest, programmable element in digital electronics.

The hierarchical architecture in Embryonics enables the implementation of a multi-level self-repairing strategy. All molecules are structurally identical and constitute a layer of reconfigurable logic, thus providing support for universal

computation. Any change in the functionality takes place by altering the binary configuration, thus allowing for a flexible redundancy strategy involving spares: every cell is a rectangular array of molecules, involving both active and spare columns.

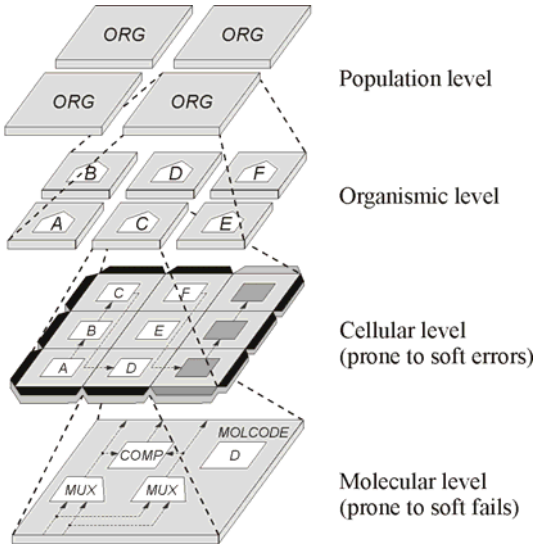


**Figure 1. Structural hierarchy in Embryonics [9]**

Whenever a faulty molecule is detected, a reconfiguration process is triggered at this level, the result being the faulty molecule's bypassing, together with the closest spare molecule becoming active and taking over its role. The reconfiguration process is shown in Figure 2; inside a simple cell consisting of 3x3 molecules, molecule *E* is detected as being faulty and is replaced by its closest spare molecule from its right (molecule *H*) through signal re-routing. The reconfiguration process at the molecular level can protect the cell's normal behavior as long as spare molecules are available for repair. The situation when no more spares are available for repair results in disabling (or "killing") the entire cell and triggering the reconfiguration process at the higher, cellular level. Such a situation occurs, for instance, when molecule *H* (which has now become active) also suffers an error, shown in Figure 3.

Organisms are also rectangular structures, where cells coexist as both active and spare columns. Figure 4 shows an organism containing 6 active cells and 2 spare cells. Let us suppose that an unrepairable error occurs inside cell C, as a result of internal damage produced by errors that occurred as described by Figure 2 and Figure 3. In this situation, the cell will "die" and the reconfiguration process at the cellular level will transfer the affected column's functionality by activating an available spare column.

A central purpose of the Embryonics' bio-inspired architecture is to ensure that the basic bricks are suitable for building extremely dependable machines. Because design flexibility also requires the existence of memory structures, a new operating mode was added at the molecular level: each molecule may be used either as programmable logic (when in logic mode), or as a storage element with data shifting features (when in memory mode). In order to detect the presence of faults and to provide an architecturally efficient compromise, both off-line and on-line self-testing strategies were used initially for the logic mode [16].

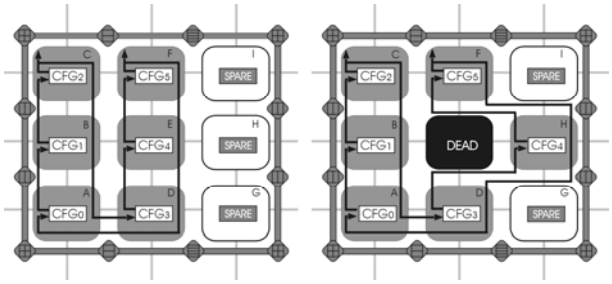


**Figure 2. Reconfiguration at the molecular level [4][16]**

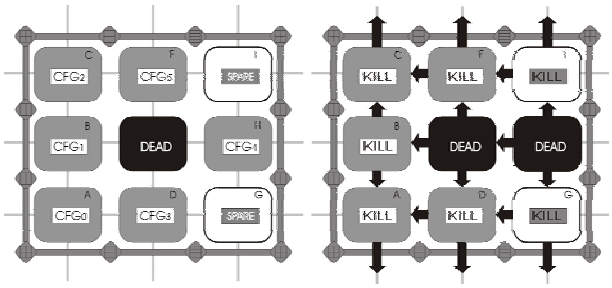


**Figure 3. Molecular reconfiguration failure. The cell "dies", triggering the reconfiguration at the cellular level [4][16]**

However, the added flexibility [11][12] could not be protected by employing the self-repairing mechanism, used in case of the logic mode; a strategy based on redundant coding was chosen in order to ensure the integrity of storage data [9]. Therefore a fault tolerant memory structure (which we called a *macro-molecule*) based on Hamming-type codes would require the existence of additional memory structures, together with corresponding logic. Typically, a complete cell (see Figure 5) would include 3 categories of molecules: logic molecules (operating in logic mode and used for combinational logic implementation), storage molecules (operating in memory mode and used for micro-programmed machine implementation) and spare molecules (used as provisions for the reconfiguration mechanisms) [9].

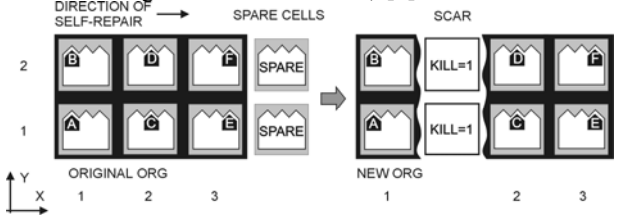


**Figure 4. Reconfiguration at the cellular level [4]**

Previous research efforts have covered reliability analysis in case of Embryonic cells made of logic-only

operating molecules [7]. The addition of the new, fault-tolerant memory structures called macro-molecules changes the Embryonics architecture and therefore reflects upon its reliability. At this point, it may be said that the quantum and bio-inspired computing (through Embryonics) do share some common ground: both target applications in which reliable computation has to be performed in harsh, error-injecting environments. Since error modeling is similar, techniques already established in quantum computing might be worth being considered for Embryonics.

### 3.1 The computation accuracy threshold

As long as a macro-molecule is concerned, *T* represents the time frame required for an error to be corrected, the worst case being a fault occurrence placed furthest from its corresponding data output port [9][12]. Such a situation occurs when the flipped data bit is positioned as the first bit from a bottom row molecule, the shifting path until it may be put into evidence and corrected being of length $F \cdot M$, where *F* is the storage dimension of the memory molecule and *M* is the vertical dimension of the macro-molecule (or the number of rows); thus $T = F \cdot M$.

Of course, when no techniques ensuring fault tolerance are implemented, *T* is proportional with the size of the data:

$$\xi \sim T^{-1} \Leftrightarrow \xi \sim \left[FM\left(N-s\right)\right]^{-1} \quad \textbf{(9)}$$

As for parameter *b* (see Equation 1), it depends on the size of the code as an expression of the gain in complexity with its dimension. In our case the size of the dataword to be protected results as $t = N - s$ bits, where *N* represents the horizontal dimension of the macro-molecule (or the total number of columns) and s represents the number of spare columns. A single error correcting Hamming code requires a number of *k* additional control (or check) bits, where *k* represents the smallest integer that satisfies the following equation:

$$k = \left\lceil \log_2\left(1 + k + N - s\right)\right\rceil \quad \textbf{(10)}$$

Therefore, the total size of the codeword, including the redundant bits results as $t + k = N - s + \left\lceil \log_2\left(1 + k + N - s\right)\right\rceil$, with the Hamming matrix being of dimensions $k \times 2^k$. As a result, any fault detection or correction process needs at most a certain number of computational steps that is given by the dimensions of the Hamming matrix, which is of the order $t \cdot \log_2\left(t\right)$. Parameter *b* can be estimated as the power of *t* that approximates best the number of necessary detection/correction steps, leading to the following equation:

$$t^b \sim c \cdot t \cdot \log_2\left(t\right) \quad \textbf{(11)}$$

where *c* is a constant. Because there are several algorithms performing the detection/correction process, we will choose the value covering the worst case scenario; following Equations 10 and 11 this value results as $b = 2$, the macro-molecular accuracy in case of integrated fault tolerance measures being:

$$\xi \sim \left[\ln\left(FM\right)\right]^{-1/2} \quad \textbf{(12)}$$

Parameter *N* does not appear directly in Equation 12 since its influence is quantified by the gain in the code's complexity defined by parameter *b* (*N* signifies the number of data bits that are to be protected, which in turn imposes the number of redundant code bits and the total length of the codeword). Equations 9 and 12 show how the macro-molecular accuracy scales for situations with and without error correction techniques. Plots for the accuracy trends are given in Figure 6, showing superior scaling when using error correcting codes as opposed to when no codes are used at all.

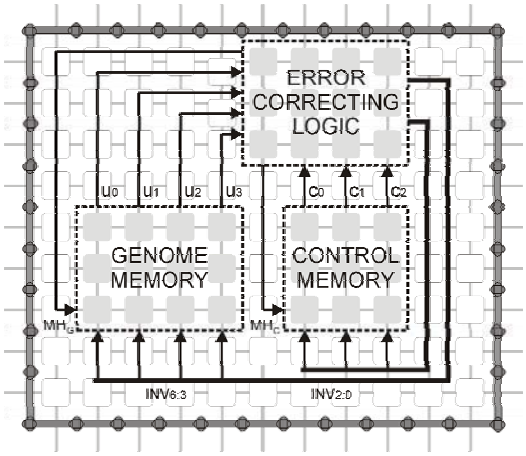


**Figure 5. A typical cell includes all 3 categories of molecules: logic, memory, and spares**

For a macro-molecule with no data error protection mechanisms, the graph from Figure 6 (top) shows an accuracy decrease when the overall storage capacity increases. This is consistent with the fact that the probability of an ocurring error is directly proportional with the area of the macro-molecule. The situation changes when error-correcting codes (or *ECC*s) are used. If each row can recover from a single error, then the accuracy dependencies show an increased efficiency with the increase of storage area; however, the graph from Figure 6 (bottom) does not contain parameters involved in an exhaustive manner, which probably leads to the final results for the macro-molecular reliability being less optimistic but, at the same time, superior to the case when no fault tolerant measures are taken into account. The qualitative results produced by the accuracy threshold estimation technique should be similar to those provided by the more classical reliability analyses.

### 3.2 Reliability analysis of a complete cell

When regarded at molecular scale, an entire cell consists of two categories of molecules, their functionality being dictated by the mode they operate in: logic or memory. There are no restrictions over the proportions in which molecules may operate in a certain mode, being possible for a cell to be made either of molecules operating in logic mode only, molecules operating in any of the memory modes, or any mixture between logic and memory modes.

Therefore, estimating the reliability of a cell is not a trivial task, since it depends on the reliability of its components [2], which operate quite differently. Furthermore, due to their different strategies in case of occurring faults, any reliability analysis has to be carried out separately for logic molecules and memory molecules. On one hand, a faulty logic molecule will be eliminated through reconfiguration, a spare one being activated in order to take its place, whereas a fault detected inside a macro-molecule does not trigger any structural reconfiguration measures at this level. The reconfiguration at the higher (cellular) level is triggered by a failure of the reconfiguration mechanism at the lower (molecular) level [4][16]. We will not discuss the details of the failure of the reconfiguration at the molecular level in case of logic molecules

A non-recoverable situation inside a macro-molecule could be addressed by two scenarios:

– since at least 2 bits worth of data will be damaged, the result is a macro-molecule that contains altered data, but the cell retains a certain level of overall functionality;
– the *KILL* process is initiated, resulting in the death of the entire cell. This is the scenario we chose to implement in Embryonics.

Whether one scenario is a better choice that the other may constitute a subject of debate, since it is difficult to say if having a functional (but, at the same time, crippled) cell has any advantages over not having that cell at all. Nature itself encounters a similar problem, since cellular mutation does not necessarily render the organism non-viable; however, altered cellular information often leads to damaging effects and illnesses, such as cancer.

#### 3.2.1 Reliability of an ensemble of logic molecules

The reliability analysis of embryonic structures made entirely by logic molecules has been previously addressed [7]. We will, however, reconsider such an analysis as the molecular internal architecture has been changed with the addition of the memory operating mode [9][12]. Let us consider that the logic molecules make up a rectangular structure of $M^*$ lines and $N^*$ columns, of which $S^*$ are spares.

Such a logic structure was analyzed as being based on the *k*-out-of-*m* reliability model, that is, the proper function of the system as a whole is ensured as long as at least *k* units out of a total of *m* are operating normally [7]. In our case, considering that any detected fault inside a molecule triggers a reconfiguration strategy that leads to the "death" of the respective molecule, this means that no more than $S^*$ errors (or faulty molecules) can be tolerated in a single row. Therefore the reliability of a single row becomes of the form:

$$R_{Row}(t) = \sum_{i=N^*-S^*}^{N^*} C_i^{N^*} e^{-i\lambda^* t} \left(1 - e^{-\lambda^* t}\right)^{N^*-i} \quad \textbf{(13)}$$

Because the logic ensemble is built of $M^*$ rows, its overall reliability results as:

$$R_{LogicEnsemble}(t) = \left[R_{Row}(t)\right]^{M^*} \quad \textbf{(14)}$$

#### 3.2.2 Reliability of a macro-molecule

We consider a non-fault tolerant macro-molecule of *M* lines and *N* columns (of which *S* are spares), each molecule storing *F* bits worth of data. Considering that $\lambda$ is the failure rate for a single flip-flop, the reliability of the entire macro-molecule is then given by:

$$R_{MMol}(t) = e^{-\lambda FM(N-S)t} \quad \textbf{(15)}$$

Parameters $M^*$, $N^*$ and $S^*$ are generally different than parameters *M*, *N* and *S* since they characterize completely different entities. Furthermore, the failure rate $\lambda$ considered for the elementary memory unit (the flip-flop) may prove to be different than the failure rate $\lambda^*$ used in case of a logic molecule (which typically employs other resources than a memory one), in which situation flip-flops may be used under different operating conditions or not be used at all.

The addition of single fault tolerance capabilities to a macro-molecule with a number of *t* columns leads to the employment of *k* additional columns (defined by Equation 10) required by the Control Memory. At each clock cycle, genome data are read through access ports and new control bits are computed (see Equation 16) and compared to those stored by the Control Memory. Figure 5 presents the block structure of a single fault-tolerant macro-molecule; the macro-molecule storing genetic data has 4 columns, thus requiring a number of *k=3* additional columns used for storing the redundant information.

The implementation of the Hamming code requires a decision being made with respect to the structure of the control memory, which could either make up for a second macro-molecule or could leave each control memory column operate independently.

The first situation unifies the control memory into a single structure that requires a detailed analysis of a macro-molecule's data path specifics [9] in order to operate correctly. Control data is computed according to Equation

16, where $c_{2:0}$ are the redundant bits output by the Control Memory and required for error correction:

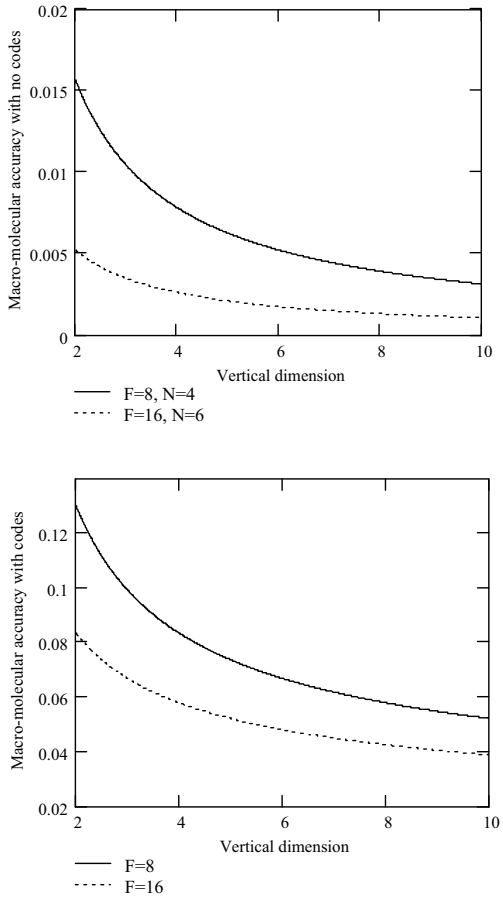


**Figure 6. Macro-molecular accuracy variation without and with codes**

$$\left.\begin{cases} c_0 = u_0 \oplus u_2 \oplus u_3 \\ c_1 = u_0 \oplus u_1 \oplus u_2 \\ c_2 = u_1 \oplus u_2 \oplus u_3 \end{cases}\right|_{t} \quad \textbf{(16)}$$

Data is continuously shifted inside both the Genome and the Control Memory [9], the process being shown intuitively in Figure 7. It is important the two memories maintain synchrony even after data permutation, in order to preserve the consistency of Equation 16. At time *t* the Error Correcting Logic (or *ECL*) reads the word $u_0u_1u_2u_3c_0c_1c_2$ (Equation 16), which will shift into $u_3u_0u_1u_2c_2c_0c_1$ at time *t*+1. If the data macro-molecule has a vertical dimension of *M* lines (each molecule storing *F* data bits), then at time $t+1+F(M-1)$ (which is necessary for the data to travel from the bottom to the output ports situated at the top of the macro-molecule [9][12]) the *ECL* will read $u_3u_0u_1u_2c_2c_0c_1$. Computing the control data for this new configuration is done by Equation 17, the identity with Equation 16 confirming the two macro-molecules (data and control) remain in synchrony for a (7,3) Hamming code.

$$\left.\begin{cases} c_2 = u_3 \oplus u_1 \oplus u_2 \\ c_0 = u_3 \oplus u_0 \oplus u_2 \\ c_1 = u_0 \oplus u_1 \oplus u_2 \end{cases}\right|_{t+1+F(M-1)} \quad \textbf{(17)}$$

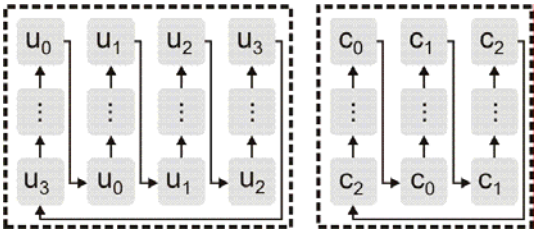


**Figure 7. Data shifting inside Genome and Control Memory for a (7,3) Hamming code implementation**

The situation changes when a (15,4) Hamming code is employed, the Genome and Control Memory being shown in Figure 8. Control data is now computed according to Equation 18:

$$\left.\begin{cases} c_0 = u_0 \oplus u_3 \oplus u_4 \oplus u_6 \oplus u_8 \oplus u_9 \oplus u_{10} \\ c_1 = u_0 \oplus u_1 \oplus u_3 \oplus u_5 \oplus u_6 \oplus u_7 \oplus u_8 \\ c_2 = u_1 \oplus u_2 \oplus u_4 \oplus u_6 \oplus u_7 \oplus u_8 \oplus u_9 \\ c_3 = u_2 \oplus u_3 \oplus u_5 \oplus u_7 \oplus u_8 \oplus u_9 \oplus u_{10} \end{cases}\right|_{t} \quad \textbf{(18)}$$

At time *t* the *ECL* reads the word $u_0u_1u_2u_3u_4u_5u_6u_7u_8u_9u_{10}c_0c_1c_2c_3$, which will shift into $u_{10}u_0u_1u_2u_3u_4u_5u_6u_7u_8u_9c_3c_0c_1c_2$ at time *t*+1. When this configuration reaches the *ECL*, the new control data will be computed (Equation 19), indicating the synchrony between the Genome and Control macro-molecules was lost during data shifting. Therefore, implementing a (15,4) Hamming code by using a Control macro-molecule shows that the solution to the synchronicity issue is not straightforward. However, it may be possible to find the right configuration for the Hamming matrix for preserving the synchrony, which has to be carefully considered for the general case.

$$\left.\begin{cases} c_3 = u_{10} \oplus u_2 \oplus u_3 \oplus u_5 \oplus u_7 \oplus u_8 \oplus u_9 \\ c_0 = u_{10} \oplus u_0 \oplus u_2 \oplus u_4 \oplus u_5 \oplus u_6 \oplus u_7 \\ c_1 = u_0 \oplus u_1 \oplus u_3 \oplus u_5 \oplus u_6 \oplus u_7 \oplus u_8 \\ c_2 = u_1 \oplus u_2 \oplus u_4 \oplus u_6 \oplus u_7 \oplus u_8 \oplus u_9 \end{cases}\right|_{t+1+F(M-1)} \quad \textbf{(19)}$$

After having investigated the possibility of implementing the Control Memory as a standalone macro-molecule, another possibility would be using independent columns to provide the control data. However, the problem of maintaining the synchrony between the genome and the control memory structures remains: the relative position of

data bits changes as they are shifted inside the genome macro-molecule, thus violating Equation 18. A possible solution would be to have separate genome macro-molecules, with each providing one data bit to the *ECL* unit. For a (7,3) Hamming code, this would mean the existence of 4 genome macro-molecules (with identical dimensions), as suggested in Figure 9.

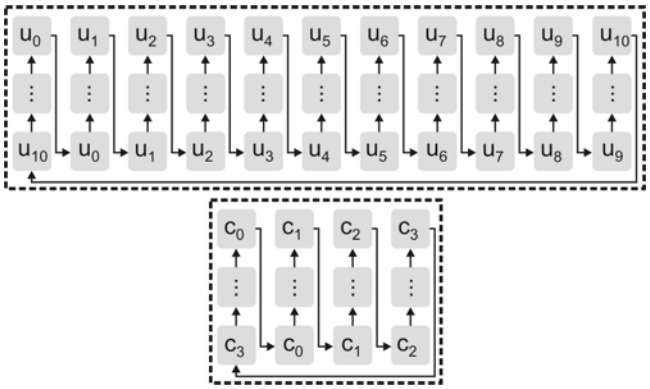


**Figure 8. Data shifting inside Genome and Control Memory for a (15,4) Hamming code implementation**

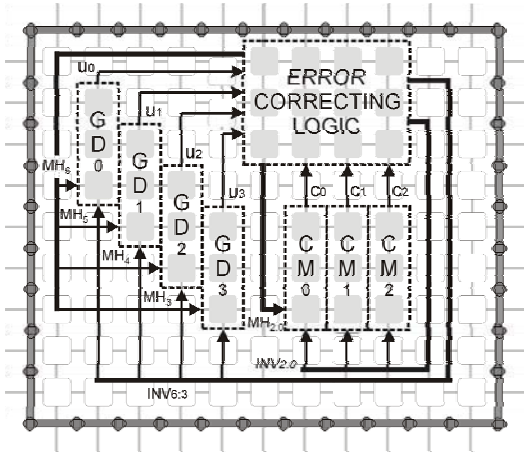


**Figure 9. Implementation of a fault-tolerant macro-molecule with independent control memory columns**

Estimating the reliability function for the storage structure involved in a fault-tolerant macro-molecule must take into consideration the fact that errors may accumulate during the shifting process. Furthermore, data has to be shifted *M*-1 lines in order to be read and, if required, corrected, a process which is also subject to error occurrence. The reliability function will therefore be of the form of the likelihood describing the situation when at most one error has occurred at the data access ports (consisting of *N* genome and *k* control bits):

$$R_{MMol}(t) \sim Prob\{\text{no fails}\} + Prob\{\text{single fail}\} \quad \textbf{(20)}$$

The first term of Equation 20 refers to the fact that no errors occured at time *t* at the north border molecules, where data is read at the *N*+*k* output ports:

$$Prob\{\text{no fails}\} = e^{-\lambda(N+k)t} \quad \textbf{(21)}$$

The second term quantifies the likelihood of experiencing a single fault at time *t* in a row, in any of the *M*-1 remaining rows:

$$Prob\{\text{single fails}\} = C_1^{M-1} C_1^{N+k} e^{-\lambda(N+k-1)t}\left(1-e^{-\lambda t}\right) \quad \textbf{(22)}$$

Then the reliability of a fault-tolerant macro-molecule results as:

$$R_{MMol}(t) \sim e^{-\lambda(N+k)t} + \\ +(M-1)(N+k)e^{-\lambda(N+k-1)t}\left(1-e^{-\lambda t}\right) \quad \textbf{(23)}$$

Plots based on Equations 15 and 23 are shown in Figure 10 for several macro-molecular dimensions. A qualitative comparison between the two situations, when there is no fault-tolerance present (top) and when single faults are tolerated (bottom), points to a significant reliability improvement brought by the addition of single fault-tolerance strategy. Although establishing precise measurements is difficult, the reliability plots are similar to those given through accuracy threshold estimation, thus supporting the relevance of this technique for Embryonics.

### 3.2.3 Reliability at the cellular level

Any cell within the Embryonics project is made of molecules operating either in logic mode or in any of the memory modes. A full reliability analysis at the cellular level requires estimating the individual reliabilities of the two component structures, macro-molecules and logic ensemble, which are given by Equations 14 and 23, respectively. All component structures are required to perform properly in order to ensure the normal operations of the cell; therefore the cell can be considered as a series system in which each subsystem (be it macro-molecule or logic ensemble) has to function if the system as a whole is to function [2]. Therefore the cellular reliability function may be derived as the product of the reliability functions of its component subsystems as follows:

$$R_{Cell}(t) = R_{LogicEnsemble}(t) \cdot \prod_{i=1}^{n} \left(R_{MMol}\right)_i (t) \quad \textbf{(24)}$$

where *n* is the number of macro-molecules present in the cell.

## 4 From multiple-level self-repairing to multiple-level coding

The accuracy threshold $\xi$, from the quantum computing context and the failure rate $\lambda$, from the bio-inspired computing context, are not dissimilar: while $\lambda$ gives the error probability, $\xi$ gives the upper bound for the error probability so as the computation is still valid. Therefore,

we have:

$$\max(\lambda) \sim \xi \quad \textbf{(25)}$$

As long as the error rate $\lambda$ remains below the accuracy threshold, valid computations can be recovered from the damaging effects of incurring errors. However, these estimations only cover the time frame between an error occurrence and the end of the recovery process, that is the period between data damage and data restoration. While a reasonable accuracy can be obtained by using error correcting codes, the occurrence of errors becomes more likely as the length of the computation increases [8]. Since machines based on the Embryonics platform are intended to operate over long periods of time (therefore involving long computations), this primarily affects the memory structures, since protective measures are already implemented within the logic ones [3].

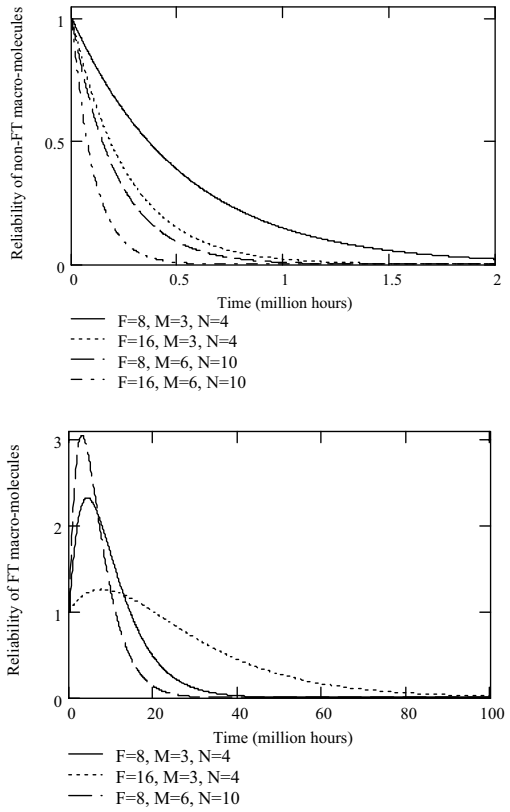


**Figure 10. Macro-molecular reliability without and with fault-tolerance implemented**

The fault tolerant quantum computation length limit can be overcome by employing concatenated codes [8]; when viewed at a higher resolution, each qubit is encoded by a block of qubits. Embryonics offers a hierarchical architecture, each level corresponding to a higher resolution view. With information encoded at each level, Embryonics seems natively endowed for implementing concatenated codes, a first idea being presented in [11]. Moreover, the successful importing of the accuracy threshold technique creates incentives for an also successful adaptation of concatenated coding in Embryonics.

Instead of storing binary words worth of data, fault-tolerant macro-molecules can store bits that would in turn assemble to provide data for the next hierarchical level an encoded binary digit. At the cellular level, genetic information may also be protected using the same or similar Hamming codes as implemented at the molecular level. If such is the case, and we accept the error rate at the macro-molecular level as being $\varepsilon$, then an unrecoverable error will occur with a probability of $\varepsilon^2$. A concatenated code [8] in which each bit at the cellular level is encoded by 7 bits at the molecular level stored by fault-tolerant macro-molecules will give the probability of an unrecoverable error as $\varepsilon^{2^2} = \varepsilon^4$ (assuming errors are of stochastic nature and uncorrelated). This is where error coding and concatenation can work together against error influences: while error coding lowers the probability of an unrecoverable error, concatenation brings the possibility of making it arbitrarily small by adding sufficient levels of concatenation.

Let us consider the following scenario: at the molecular level, genetic information is stored by fault-tolerant macro-molecules as a (7,3) Hamming code [2]. Essentially, 4 bits worth of genetic data are encoded into a 7-bit codeword, which makes up the elementary piece of information at this level. At the cellular level, these 7 bits are considered as a single "bit" of actual data; by applying the same Hamming encoding, 4 such "bits" require a number of 3 additional control "bits", the resulting codeword being able to recover from an error affecting a single such "bit". An unrecoverable situation occurs when a double error affects a codeword at the cellular level. However, this can only happen if two sub-blocks fail simultaneously, which, in turn, means that each of the two (7,3) Hamming codewords have to experience a double error. Such a concatenated code offers superior protection, the unrecoverable situation occuring with a probability of $\varepsilon^4$. The situation is depicted in Figure 11, where higher level "bits" are encoded as follows: a "1" at cellular level can be encoded at the molecular level by any Hamming word with an odd number of 1s, while a "0" can be encoded by any Hamming word with an even number of 1s. In general, if *N* such levels of concatenations are used, the probability of an unrecoverable error decreases to $\varepsilon^{2^N}$, which becomes negligible if $\varepsilon$ is reasonably small.

From an engineering standpoint, concatenated coding involves significant hardware overhead, with one bit at the cellular level being encoded as 7 bits at the molecular level. An "efficient" platform should theoretically employ only minimal resources. However, Nature, as the best engineer,

chose redundancy for any of its creations, which constitutes hard evidence that redundancy should not be considered as equivalent to resource wasting. If high redundancy seems a problem for the moment, perhaps the real culprit lies under current technological limitations.

## 5 Conclusions

This paper is supported by an analysis revealing similarities between two emerging fields in modern computing, namely quantum and bio-inpired fault-tolerant computing. Driven by the same goal, surviving in an aggressive and frequent faults inducing environment, both fields share the same fault model. Moreover, we proved that the qualitative reliability measures and fault-tolerance techniques from quantum computing can be adapted to embryonic memories, intended to operate reliably under cosmic ray influences. The accuracy threshold is a valid indicator that, although producing qualitative figures similar to classical reliability analyses, provides a useful perspective regarding design principles used for attaining multiple-level fault-tolerance.

As shown in the core of the paper, the (7,3) Hamming code has the property of maintaining the consistency of its equations through the data shifting proces. Future work will investigate the possibility of employing more extensive (and therefore, more efficient) coding schemes.

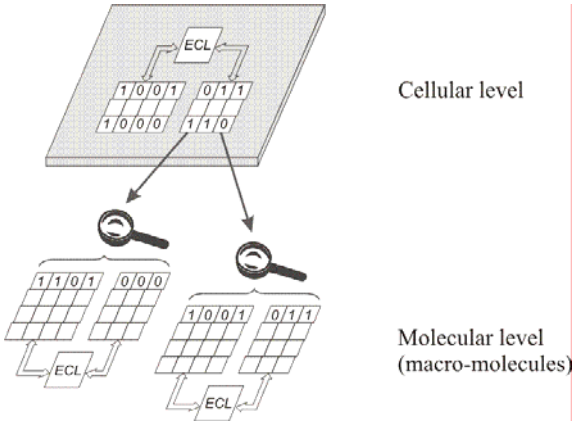


**Figure 11. Two-level concatenated coding in Embryonics**

## References


[1] Avižienis, A., Laprie, J.C., Randell, B., Landwehr, C. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, 1, 1 (Jan-Mar 2004), 11-33.

[2]Lala, P.K. *Fault Tolerance and Fault Testable Hardware Design*. Prentice Hall, 1985.

[3]Mange, D., Sipper, M., Stauffer, A., Tempesti, G. Toward Robust Integrated Circuits: The Embryonics Approach. In *Proc. IEEE*, vol. 88, No. 4, April 2000, pp. 516-541.

[4] Mange, D. and Tomassini, M. eds. *Bio-Inspired Computing Machines: Towards Novel Computational Architectures.* Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland, 1998.

[5] Neumann, J. Von. Probabilistic Logic and the Synthesis of Reliable Organisms from Unreliable Components. In C.E. Shannon, J. McCarthy (eds.) *Automata Studies*, *Annals of Mathematical Studies 34*, Princeton University Press, 1956, 43-98.

[6] Nielsen, M.A., Chuang, I.L. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[7] Ortega, C., Tyrrell, A. Reliability Analysis in Self-Repairing Embryonic Systems. *Proc. 1st NASA/DoD Workshop on Evolvable Hardware*, Pasadena CA, 1999, 120-128.

[8] Preskill, J. Fault Tolerant Quantum Computation. In H.K. Lo, S. Popescu and T.P. Spiller, eds. *Introduction to Quantum Computation*, World Scientific Publishing Co., 1998.

[9] Prodan, L., Udrescu, M., Vladutiu, M. Self-Repairing Embryonic Memory Arrays. *Proc. IEEE NASA/DoD Conference on Evolvable Hardware*, Seattle WA, 2004, 130-137.

[10] Prodan, L., Tempesti, G., Mange, D., and Stauffer, A. Embryonics: Electronic Stem Cells. *Proc. Artificial Life VIII*, The MIT Press, Cambridge MA, 2003, 101-105.

[11] Prodan, L., Tempesti, G., Mange, D., and Stauffer, A. Embryonics: Artificial Cells Driven by Artificial DNA. *Proc. 4th International Conference on Evolvable Systems (ICES2001)*, Tokyo, Japan, LNCS vol. 2210, Springer, Berlin, 2001, 100-111.

[12] Prodan, L., Tempesti, G., Mange, D., and Stauffer, A. Biology Meets Electronics: The Path to a Bio-Inspired FPGA. In *Proc. 3rd International Conference on Evolvable Systems (ICES2000),* Edinburgh, Scotland, LNCS 1801, Springer, Berlin, 2000, 187-196.

[13] Spector, L. *Automatic Quantum Computer Programming: A Genetic Programming Approach*. Kluwer Academic Publishers, Boston MA, 2004.

[14] Udrescu, M., Prodan, L., Vladutiu, M. Using HDLs for describing quantum circuits: a framework for efficient quantum algorithm simulation**.** *Proc. 1st ACM Conference on Computing Frontiers*, Ischia, Italy, 2004, 96-110.

[15] Udrescu, M., Prodan, L., Vladutiu, M. The Bubble Bit Technique as Improvement of HDL-Based Quantum Circuits Simulation. *Proc. IEEE 38th Annual Simulation Symposium*, San Diego CA, April 2005, 217-224.

[16] Tempesti, G. *A Self-Repairing Multiplexer-Based FPGA Inspired by Biological Processes*. Ph.D. Thesis No. 1827, Logic Systems Laboratory, The Swiss Federal Institute of Technology, Lausanne, 1998.

[17] Zalka, C. Threshold Estimate for Fault Tolerant Quantum Computation. *arXiv:quant-ph/9612028*, v2, 28 Jul. 1997.