Multiple-Level Concatenated Coding in Embryonics: A Dependability Analysis

Lucian Prodan Mihai Udrescu Mircea Vladutiu
Advanced Computing Systems and Architectures Laboratory
"Politehnica" University, Computer Department
2 V.Parvan Blvd, 300223 Timisoara TM, 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
Computing machines require the highest possible dependability in order to provide accurate functionality in aggressive, critical environments. For this purpose, the Embryonics (for *embryonic electronics*) project explores Nature's structural redundancy mechanisms in digital electronics. It offers a hierarchically reconfigurable framework [4][5][18], whose effectiveness was assessed only for some particular cases [8]. Following the introduction of specialized memory structures [10][13], this paper proposes a more thorough reliability analysis, inspired by fault-tolerant quantum computing theory. After adopting the accuracy threshold measure as the main parameter for our qualitative evaluation, the concepts and implementation details about concatenated coding are presented. This technique, also inspired from reliable quantum computing, seems particularly well suited for the multiple-level architecture in Embryonics and allows preserving arbitrary long fault-tolerant computation.

Categories and Subject Descriptors
B.8.1–Reliability, Testing, and Fault-Tolerance

General Terms
Algorithms, Reliability, Theory

Keywords
Embryonics, reliability, accuracy threshold, concatenated coding

1. INTRODUCTION
The technological era of our present days relies on computers as some of its finest representatives. They offer a variety of balance between sheer computational power and ability of running their programs both accurately and relentlessly. While there is no argument over the necessity of computer evolution, it is the choice of directions that raises difficulties, because of its dual purpose: some applications require speed above anything else, others require highest possible reliability. It is a largely acknowledged fact in engineering that enhancing certain parameters usually affects others, unfortunately leading to a common situation in which computers find themselves unable to fully fulfill any of their tasks. The scientific rush for new inspiration in both their hardware and software designs is therefore well justified. Since their beginning, computers were protagonists of the quest for performance; the resulting benefits decisively led to both a scientific and industrial blossoming, the pinnacle being the coming of the space exploration era. At this stage, the critical mass accumulated in computing started to encourage a shifting in performance priorities from brute computing force (which seems to have reached somewhat sufficient levels today) towards the advent of computers offering superior dependability.

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 that is acceptable*" [1]. It is therefore a synthetic term that involves a list of attributes including reliability, fault tolerance, availability, and others. In real world, a dependable system would have to operate normally over long periods of time before experiencing any fail (reliability, availability) and to recover quickly from errors (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. Dependable systems are crucial for applications that prohibit or limit human interventions, such as long-term exposure to aggressive (even hostile) environments.

The quest of building digital systems that offer superior dependability can draw benefits from at least two distinct sources. The first one is the oldest and most complex computing system, which 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 living beings; although such a thorough analysis is beyond the scope of this paper, Nature has the upper hand at least when it comes to design periods: if

GECCO'05, June 25–29, 2005, Washington, DC, USA.

engineers have a limited time for providing ever better computer designs, Nature uses time frames that are impossible to attain. It is likely, therefore, that the field of digital computing could benefit by exploiting some of the mechanisms implemented by Nature and adapting them to the electronic environment; a representative attempt is the Embryonics project [4][5].

A second source of inspiration may be constituted by novel computing paradigms, whose research already considered dependability-raising techniques. Although pertaining to the category of artificial systems, quantum computing represents an emerging field in which successful calculus takes place in an error-prone, micro-scale environment. Since frequent errors are (as of yet) intrinsic to quantum systems, a number of techniques were established in order to recover from their damaging effects.

However, though a variety of methodologies for estimating dependability parameters have been proposed, they usually remain localized to their originating field and rarely reach other architectures. This paper proposes a unifying view by drawing inspiration from *both* quantum and bio-inspired computing. We argue upon the benefits drawn by importing a methodology of estimating the computing accuracy threshold from quantum computing to the Embryonics project. As a member of the evolvable hardware family, the Embryonics platform is also subject to evolution of its concepts. An updated reliability analysis introduces the concatenated coding as a supplemental dependability-raising technique.

2. Dependability in Quantum Computing

Quantum computation uses atomic-scale dynamics [15] 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 [15]. Any qubit ψ is a normalized vector in an $\mathcal{H}^2$ Hilbert space, with $|0\rangle$ and $|1\rangle$ as the orthonormal basis being equivalent to the classical, binary states: $|\psi\rangle = a_0|0\rangle + a_1|1\rangle$. Parameters a_0, $a_1 \in \mathbb{C}$, called *quantum amplitudes*, 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 [7].

An essential promise of quantum computing is solving in polynomial time problems that are otherwise known (in classical computing) to have exponential solutions only. We will not go into details, as 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 [16][17] before any benefits are to be drawn. Dealing with dependability issues constitutes a priority in quantum computing because of its innate erroneous nature. Faults are native to the quantum environment, since a quantum state cannot be fully isolated from the environment; the environment continuously attempts to measure it, which 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), all faults can be reduced to bit flips [7][9]. The errors affecting quantum computing processes are considered to be uncorrelated, neither in space, nor in time [9]. These error characteristics are also common in classical computing, where soft fails are induced in digital devices by aggressive radiations [2][10]. As a prerequisite for building dependable computing systems, dealing with these errors may successfully act as a liaison between the fields of quantum and classical computing: the latter may benefit by adapting readily available fault tolerant techniques from the former, such as the accuracy threshold as the basic reliability measure.

As quantum computing takes place in an error-injecting environment, the frequency of errors imposes recovery procedures (through redundant coding) for accurate computation. However, the recovery process is by itself vulnerable to errors: as information is restored through the use of additional, redundant information, new errors may occur and affect data *during* the very recovery process. In order to ensure a sufficient level of fault tolerance, 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 error frequency that would still allow a successful recovery? These questions were answered in the quantum context [9][19]; we will however revisit the proposed qualitative assessment since we believe a similar reasoning may also be applied to bio-inspired computing systems (Embryonics) and fault-tolerant digital systems in general.

If the redundant coding allows the correction of t errors, then an unrecoverable error occurs if at least t+1 errors occur before the recovery process ends. Therefore, if the probability of an error is ξ, then an unrecoverable error occurs with a probability of the order ξ^{t+1} [9][19]. Apparently, choosing a reasonably high value for t can make this probability as small as desired; however, the complexity of the code rises steeply 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 an unrecoverable event occurrence becomes most likely. The block error probability (*BEP*) of t+1 errors accumulating in a codeword before the recovery is complete will then have the form [9]:

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

Minimizing the *BEP* function after parameter t yields:

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

which results in:

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

Solving Equation 3 and assuming that t is large [9] 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:

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

The result for $MBEP(\xi)$ is important with respect to estimating

the required accuracy for a reliable computation. If T is 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, ξ 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], providing precise values is difficult. However, criteria for a dependability comparison between two functionally identical systems, before and after applying fault tolerance measures, are established.

3. EMBRYONICS

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 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 [11]), at the cellular level (cells can replace each other when required) 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 rich variety of living beings, and, 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 [6].

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 [5]. 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 Embryonics architecture consists of a quasi-biological hierarchy based on four levels of organization (Figure 1) [4][5]. The targeted applications are those in which the failure frequency must be very low to be "acceptable".

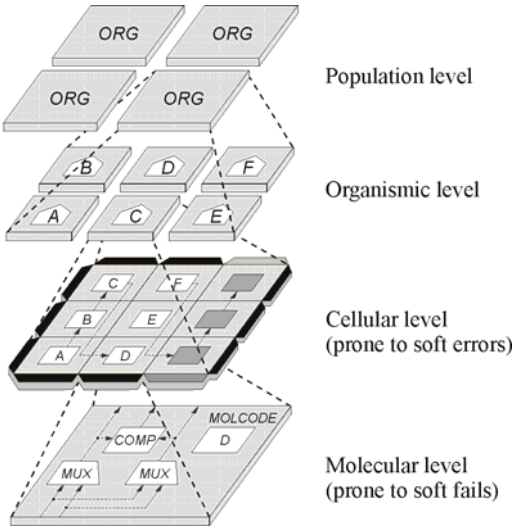


Figure 1. Structural hierarchy in Embryonics [10]

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. According to Embryonics, 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 [4].

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, allowing for a flexible redundancy strategy: each cell is a rectangular array of molecules, involving both active and spare columns. Whenever a faulty molecule is detected, a reconfiguration process is triggered at this level: the closest spare molecule becomes active and takes over its role, while the faulty molecule is bypassed. The reconfiguration process is shown in Figure 2; inside a simple cell consisting of 3x3 molecules, molecule *E* is detected as being faulty and replaced by its closest spare neighbor from its right (molecule *H*) through signal re-routing. The reconfiguration at the molecular level protects the cell's normal behavior as long as spare molecules are available for repair. When these become unavailable, the entire cell is disabled (or "killed"), thus triggering the reconfiguration process at the higher, cellular level [18].

Organisms are also rectangular structures, where cells coexist as both active and spare columns Let us consider the organism shown in Figure 3 (left), which contains 6 active cells and 2 spare cells. After an error affected molecule *E* and triggered the reconfiguration process (presented in Figure 2), cell *C* suffers another error inside molecule *H*. This is a non-repairable error (there are no more spare molecules left for reconfiguration); 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 [18]. The result of the reconfiguration at the cellular level is presented in Figure 3 (right).

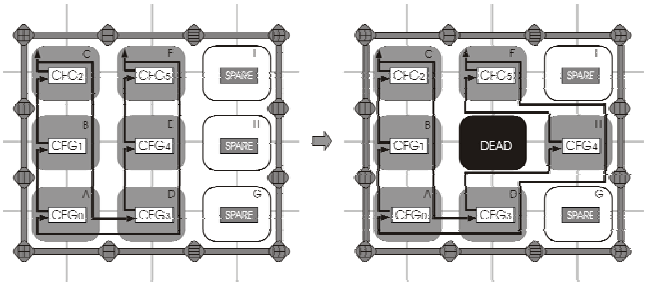


Figure 2. Reconfiguration at the molecular level [18]

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 (which we call *macro-molecules*), 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 for what was initially the logic mode [13].

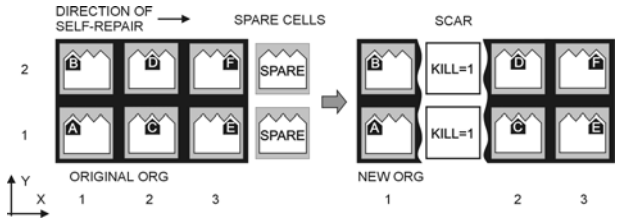


Figure 3. Reconfiguration at the cellular level [13]

However, the added flexibility could not be protected by employing the same 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 [10]. Therefore a fault tolerant memory structure based on Hamming-type codes would require the existence of additional memory structures, together with corresponding logic. Typically, a complete cell (see Figure 4) includes 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) [10].

Previous research efforts have covered reliability analyses in case of Embryonic cells made of logic operating molecules only [8]. The addition of the new, fault-tolerant macro-molecules changes the Embryonics architecture and reflects upon its reliability. We will provide such an analysis in order to introduce the accuracy threshold qualitative assessment and argue upon the concepts and implementation of concatenated coding in Embryonics.

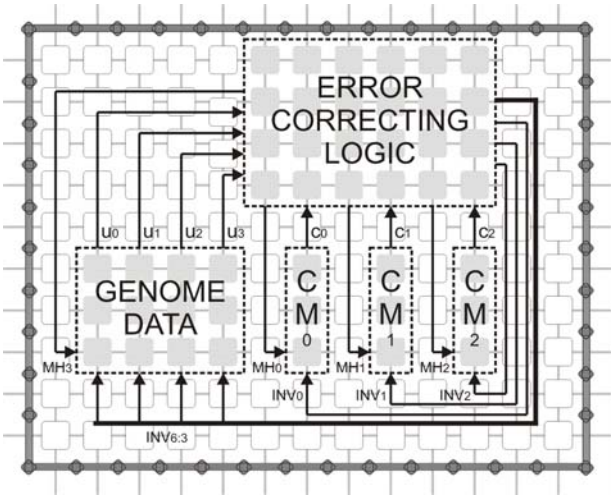


Figure 4. A typical cell includes 3 types of molecules: logic (white), memory (gray), and spares (user transparent) [18]

3.1 Estimating 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 [10]. 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 data word 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 check bits, where *k* represents the smallest integer that satisfies the following equation:

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

Therefore, the total size of the codeword, including the redundant bits results as $t + k = N - s + \lceil \log_2\left(1 + k + N - s\right) \rceil$, with the Hamming matrix being of dimensions $k \times 2^k$. As a result, any fault detection/correction process needs at most a 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:

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

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:

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

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 5, showing superior scaling when using error-correcting codes as opposed to when no codes are used at all.

For a macro-molecule with no data error protection mechanisms, the graph from Figure 5 (top) shows an accuracy decrease when the overall storage capacity increases. This is consistent with the fact that the probability of an occurring error is directly proportional with the area of the macro-molecule. The situation changes when ECC codes are used: if each row can recover from a single error, the accuracy dependencies show an increased efficiency; because parameters are not involved in an exhaustive manner in the graph from Figure 5 (bottom), the final results for the macro-molecular reliability will probably result as less optimistic but, at the same time, superior to the case when no fault tolerant measures are taken into account (Figure 5, top).

3.2 Reliability Analysis of a Complete Cell

When regarded at molecular scale, an entire cell consists of two parts. First, there is the cellular membrane [18], which is implemented by a small automaton that has no functional role (that is, does not participate actively to any logical machine implementation), its only purpose being that of specifying the borders of a cell. The second, and most important, part of a cell consists of its molecules, their functionality being dictated by the mode they operate in. 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.

Estimating the reliability of a cell is therefore not a trivial task, since it depends on the reliability of its components, which may operate differently. Furthermore, any reliability analysis has to be carried out separately for logic molecules and memory molecules, due to their different strategies in case of incurring faults. 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.

3.2.1 Reliability of a Macro-Molecule

We will start our analysis by considering a general macro-molecule consisting of a memory array of M lines and N columns (of which S are spares) of molecules, each storing F bits worth of data and with no fault tolerance in place. Considering that λ is the failure rate for a single flip-flop, the reliability of the entire macro-molecule is then given by Equation 13:

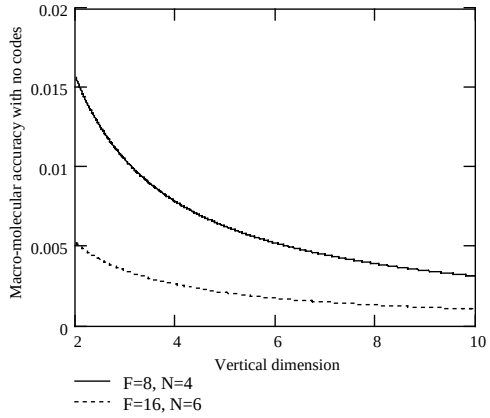
$$R_{MMol}(t) = e^{-\lambda FM(N-S)t} \quad \textbf{(13)}$$

On the other hand, adding single fault tolerance capabilities to this macro-molecule leads to the employment of k additional columns or arrays of Mx1 memory molecules, required by storing redundant data. Then the reliability function for a macro-molecular row can be redefined as follows:

$$\begin{aligned} R_{Row}(t) &= Prob\{no\ FF\ fails\}(t) + Prob\{\text{single } FF\ fail\}(t) \\ R_{Row}(t) &= e^{-\lambda(N-S+k)t} + \\ &\quad + C^{1}_{N-S+k}\left(1-e^{-\lambda t}\right)e^{-\lambda(N-S+k-1)t} \end{aligned} \quad \textbf{(14)}$$

which gives the overall reliability function:

$$R_{MMol}(t) = \left[R^{F}_{Row}(t)\right]^{M} = \left[R_{Row}(t)\right]^{FM} \quad \textbf{(15)}$$



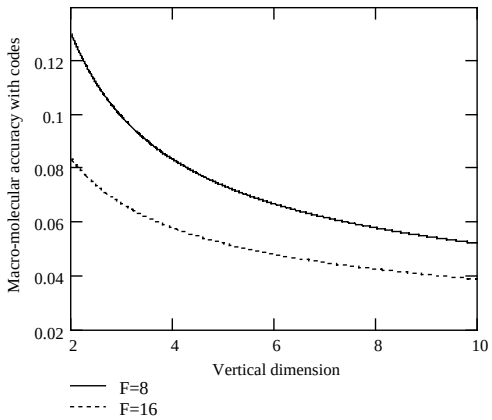


Figure 5. Macro-molecular accuracy variation without and with codes

3.2.2 Reliability of an Ensemble of Logic Molecules

The reliability analysis of embryonic structures made entirely by logic molecules has been previously addressed [8]. We will, however, reconsider such an analysis as the molecular internal

architecture has been changed with the addition of the memory operating mode [10]. Let us consider that the logic molecules make up a rectangular structure of M^* lines and N^* columns, of which S^* are spares. Parameters M^*, N^* and S^* are generally different than parameters M, N and S considered in subsection 3.2.1 since they characterize completely different entities. Furthermore, the failure rate λ considered for the elementary memory unit (the flip-flop) may prove to be different than the failure rate λ^* used in case of a logic molecule (which typically employs other resources than memory), in which situation flip-flops may either be used under different operating conditions or not be used at all.

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 still operating normally [8]. 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:

$$R_{Row}(t) = \sum_{i=N^*-S^*}^{N^*} C_{N^*}^{i} e^{-i\lambda^* t} \left(1 - e^{-\lambda^* t}\right)^{N^*-i} \quad \textbf{(16)}$$

Because the logic ensemble is built of M^* rows, its overall reliability can now be estimated as:

$$R_{LogicEnsemble}(t) = \left[R_{Row}(t)\right]^{M^*} \quad \textbf{(17)}$$

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 15 and 17, 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 [3]. Therefore the cellular reliability function may be derived as the product of the reliability functions of its component subsystems as follows:

$$R_{Cell}(t) = R_{LogicEnsemble}(t) \prod_{i=1}^{n} \left(R_{MMol}\right)_i (t) \quad \textbf{(18)}$$

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

3.2.4 Reliability at the Organismic Level

When more faults affect the internal structure of a cell than spare resources are available for repairing, the cell becomes faulty and needs replacement within the organism. In this situation a reconfiguration process that will mark and eliminate the entire column of cells (including the faulty cell) is activated. Ongoing cellular processes from the marked column will be taken over by a spare column by shifting them to the right. In order to illustrate the reconfiguration process, Figure 3 (left) presents a cellular structure affected by faults at this (cellular) level, through the *C* cell. Since any fault detected at the cellular level triggers a column-elimination strategy, Figure 3 (right) shows the organism's layout after the reconfiguration. Because cell *C* was faulty, this means the entire column (which includes cells *C* and *D*) will be disabled, its role being transferred to the closest spare column to the right, which will become active.

The above considerations justify the reliability function of an organism as also being based on the *k*-out-of-*m* model, where the successful operation of the organism is ensured by the proper function of at least *k* columns out of a total of *m*. If the organism consists of M_c lines and N_c columns (including S_c spares), its reliability is given by the fact that, at any moment, at least N_c-S_c columns are operational:

$$R_{Org}(t) = \sum_{i=N_c-S_c}^{N_c} C_{N_c}^{i} R_{Column}^{i} (1 - R_{Column}^{i})^{N_c - i} \quad \textbf{(19)}$$

Since a column is fully operational if all M_c component cells are functional, the reliability function for a column results as:

$$R_{Column}(t) = R_{Cell}^{M_c}(t) \quad \textbf{(20)}$$

4. PUTTING IT ALL TOGETHER

At a first glance, the boundaries between bio-inspired computing and quantum computing seem to discourage unveiling any common ground between the two fields. Though technology may be essentially different, they both share the same error model and employ techniques for achieving fault tolerance from classic computing. Moreover, the accuracy threshold ξ in the quantum computing context and the failure rate λ in the bio-inspired computing context are not dissimilar: while λ gives the error probability, ξ gives the upper bound for the error probability so as the computation is still valid. Therefore, we have:

$$\max(\lambda) \sim \xi \quad \textbf{(21)}$$

As long as the error rate λ is 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 [9]. 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 in Embryonics, since its logic structures already have protective measures implemented [4].

5. FROM MULTIPLE-LEVEL SELF-REPAIR TO MULTIPLE-LEVEL CODES

The fault tolerant quantum computation length limit can be overcome by employing concatenated codes [9]; when viewed at a higher resolution, each qubit is encoded by a block of qubits. Such a hierarchical encoding appears to be particularly well suited for the Embryonics project since its architecture offers an intrinsic hierarchy, one level corresponding to a higher resolution view of the next superior level. With information being encoded at each level, Embryonics seems natively endowed for implementing concatenated codes, the principles being presented in Figure 6; a first idea of information coding in Embryonics for error detection purposes was presented in [12].

Instead of storing binary words worth of data, fault-tolerant macro-molecules can store binary words that would in turn assemble to provide data for the next hierarchical level as an encoded binary digit. At the cellular level, genetic information may also be protected using 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 ε, then an unrecoverable error will occur with a probability of ε^2. A concatenated code in which each bit at the cellular level is encoded by 7 bits at the molecular level stored by fault-tolerant macro-molecules [9] 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.

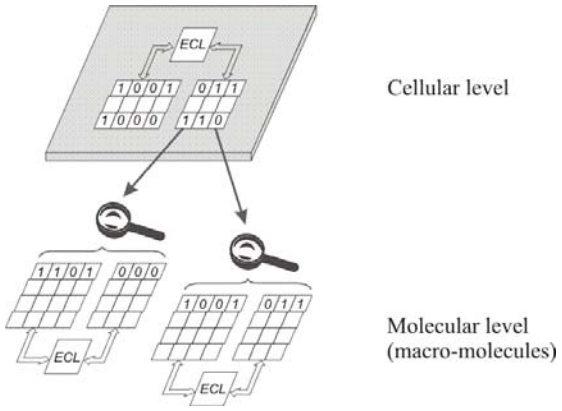


Figure 6. 2-level concatenated coding in Embryonics

In Figure 6 the following scenario is being considered: at the molecular level, genetic information is divided and stored by fault-tolerant macro-molecules using a (7,3) single error correcting Hamming code [3]. Essentially, 4 bits worth of genetic data (stored by the GENOME MEMORY in Figure 4) are encoded into a 7-bit codeword, which makes up the elementary piece of information at this level. The redundant check bits, stored by the CONTROL MEMORY ($CM_{0\div2}$ in Figure 4) are derived as follows [14]:

$$\begin{cases} c_0 = u_0 \oplus u_2 \oplus u_3 \\ c_1 = u_0 \oplus u_1 \oplus u_2 \\ c_2 = u_1 \oplus u_2 \oplus u_3 \end{cases} \quad \textbf{(22)}$$

where u_0, u_1, u_2, and u_3 are the 4 genetic data bits.

At the cellular level, each 7-bit code word from the molecular level make up for a single higher-order bit of actual data, which will be called Bit from this moment; its value can be derived, for instance, as the parity value from Equation 23:

$$U^i = u_0^i \oplus u_1^i \oplus u_2^i \oplus u_3^i \oplus c_0^i \oplus c_1^i \oplus c_2^i, \text{ for } i = 0 \div 3 \quad \textbf{(23)}$$

The same single error correcting, Hamming coding, from the molecular level (see Equation 22) can now be applied to the 4 Bits $U^{0\div3}$ in order to generate the redundant check Bits $C^{0\div2}$:

$$\begin{cases} C^0 = U^0 \oplus U^2 \oplus U^3 \\ C^1 = U^0 \oplus U^1 \oplus U^2 \\ C^2 = U^1 \oplus U^2 \oplus U^3 \end{cases} \quad \textbf{(24)}$$

At this point, a structure that encodes genetic information in a hierarchical manner by using concatenated codes has been established. At the molecular level, the basic units (the memory molecules) are assembled to build a fault tolerant macro-molecule (FTMM), which is shown in Figure 7 (more details are given in [10]). The *FTMM* computes the value of the corresponding Bit by implementing Equation 23, while the *ECL* keeps the code word accurate by implementing Equation 22.

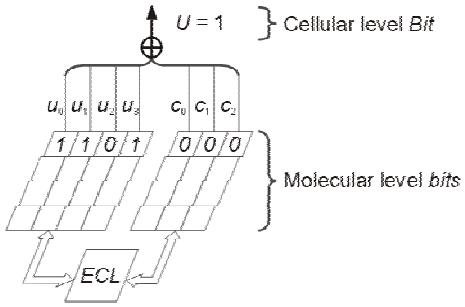


Figure 7. The first level in concatenated coding is the FTMM (Fault-Tolerant Macro-Molecule)

At the cellular level a similar structure is assembled (see Figure 8), with the basic units being the *FTMM*s. Each *FTMM* computes a Bit, with 7 such Bits making up a (7,3) Hamming code. The correction mechanism at the cellular level is identical to that present at the molecular level [10]: whenever a single error affects a 7-Bit word, the error is located and the corresponding value inverted.

The check Bits provide vital information for recovering a code word from a single error at the cellular level. Their value is computed directly from the Bits that carry genetic information ($U^{0\div3}$) and do not come from actual data from the molecular level. Therefore, there seems to be no real need for further encoding the check Bits. However, if the advantages of concatenated codes are to be preserved, the check Bits also require coding; if this process implies the derivation of a new value (the code) from several values known in advance (source data), in this case a reverse process is required: the value of the encoded data is known in advance at the cellular level (that is, the value of the check Bit) and the values from the molecular level (source data) need to be computed.

As it is implemented, the code word resulting from Equation 24 is also able to recover from an error affecting a single Bit. An unrecoverable situation occurs when a double error affects a code word 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 code words from the molecular level have to experience a double error. Because each Bit is encoded as suggested by Equation 23 (shown in Figure 7), such a concatenated code offers superior protection.

Considering Equation 14 and substituting with 7 the length of a code word implemented by a macro-molecule with single fault-tolerance, its reliability becomes:

$$R_{bit\_word}(t) = e^{-7\lambda t} + 7\left(1 - e^{-\lambda t}\right)e^{-6\lambda t} \quad \textbf{(25)}$$

Because the length of the code word and the fault-tolerance are similar at the cellular level, the reliability of the code word at this level is:

$$R_{Bit\_word}(t) = R^{7}_{bit\_word}(t) + 7\left[1 - R^{7}_{bit\_word}(t)\right]R^{6}_{bit\_word}(t) \quad \textbf{(26)}$$

A direct comparison between Equations 25 and 26, which define the reliability function for the basic information unit at each level, confirms the superior protection offered by a second level of concatenated coding.

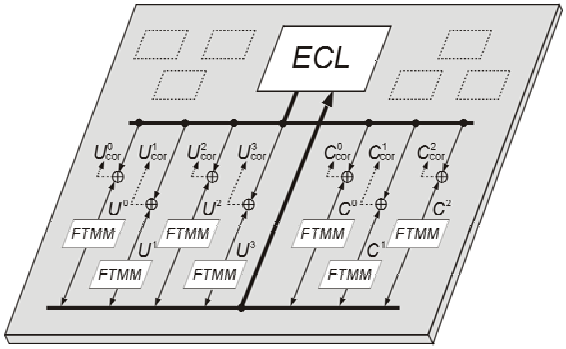


Figure 8. The second level in concatenated coding is made-up by Hamming-coded Bits

6. CONCLUSIONS

This paper identified a common problem in both bio-inspired and quantum computing: attaining superior dependability in environments inducing frequent faults. We have shown that the accuracy threshold estimation (inspired from quantum computing) can be linked to the reliability analysis of bio-inspired computing, both techniques producing similar qualitative results. Because applications targetted by both quantum computing and bio-inspired computing (also Embryonics) share the same high dependability requirements, the accuracy threshold estimation is relevant for both fields. Therefore, concatenated coding also represents a possible solution for Embryonics. Its hierarchical architecture is structurally similar to that of concatenated coding, thus facilitating the implementation presented in the paper.

Future work will focus on providing quantitative results by employing simulated fault injection. An automated reliability assessment could be used to validate a system architecture with respect to the environmental conditions intended to operate in.

7. 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] Van de Goor, A.J. *Testing Semiconductor Memories. Theory and Practice*. John Wiley and Sons, 1991.

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

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

[5] 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.

[6] 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.

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

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

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] Rao, T.R.N., Fujiwara, E. *Error-Control Coding for Computer Systems.* Prentice-Hall, 1989.

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

[16] 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.

[17] Udrescu, M., Prodan, L., Vladutiu, M.. A New Perspective in Simulating Quantum Circuits. *Proc. GECCO*, Chicago IL, July 2003, 283-290.

[18] 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.

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