A Dependability Perspective on Emerging Technologies

Lucian Prodan   Mihai Udrescu   Mircea Vladutiu

Advanced Computing Systems and Architectures (ACSA) Laboratory,
Computer Science and Engineering Department, "Politehnica" University of Timisoara,
2 V.Parvan Blvd, 300223 Timisoara, Romania
www.acsa.upt.ro

+40-722-664779   +40-723-154989   +40-256-403258

lprodan@cs.upt.ro   mudrescu@cs.upt.ro   mvlad@cs.upt.ro

ABSTRACT

Emerging technologies are set to provide further provisions for computing in times when the limits of current technology of microelectronics become an ever closer presence. A technology roadmap document lists biologically-inspired computing and quantum computing as two emerging technology vectors for novel computing architectures [43]. But the potential benefits that will come from entering the nanoelectronics era and from exploring novel nanotechnologies are foreseen to come at the cost of increased sensitivity to influences from the surrounding environment. This paper elaborates on a dependability perspective over these two emerging technology vectors from a designer's standpoint. Maintaining or increasing the dependability of unconventional computational processes is discussed in two different contexts: one of a bio-inspired computing architecture (the Embryonics project) and another of a quantum computational architecture (the *QUERIST* project).

Categories and Subject Descriptors

B.8.1 [**Performance and Reliability**]: Reliability, Testing, and Fault-Tolerance.
C.4 [**Performance of Systems**]: Fault-Tolerance, Reliability, Availability, and Serviceability.

General Terms

Design, Reliability, Theory.

Keywords

Dependability, emerging technologies, evolvable hardware, bio-inspired computing, bio-inspired digital design, Embryonics, reliability, quantum computing, fault-tolerance assessment.

1. INTRODUCTION

High-end computing has reached nearly every corner of our present day life, in a variety of forms taylored to accommodate either general purpose or specialized applications. Computers may be considerred as fine exponents of the present days' technological wave – if not their finest, they certainly do count as solid, indispensable support for *the* finest.

From the very beginning of the computing advent, the main target was squeezing out any additional performance. The inception period was not always trouble-free, accurate computation results being required at an ever faster pace on a road that has become manifold: some applications do require computational speed as a top priority; others are set for the highest possible dependability, while still delivering sufficient performance levels.

Several definitions for dependability have been proposed: "*the ability of a system to avoid service failures that are more frequent or more severe than is acceptable*" [2], or "*the property of a computer system such that reliance can justifiably be placed on the service it delivers*" [9][45]. Dependability is therefore a synthetic term specifying a qualitative system descriptor that can generally be quantified through a list of attributes including reliability, fault tolerance, availability, and others.

In real world, a dependable system would have to operate normally over extended periods of time before experiencing any fail (reliability, availability) and to recover quickly from errors (fault tolerance, self-test and self-repair). The term "*acceptable*" has an essential meaning within the dependability's definition, setting the upper limits of the damages that can be supported by the system while still remaining functional or computationally accurate. A dependability analysis should take into consideration if not quantitative figures for the acceptable damage limit, at least a qualitative parameter representation for its attributes.

Dependable systems are therefore crucial for applications that prohibit or limit human interventions, such as long-term exposure to aggressive (or even hostile) environments. The best examples are long term operating machines as required by managing deep-underwater/nuclear activities and outer space exploration.

There are three main concerns that should be posed through a system's design in order to achieve high dependability [42]:

1. Specifying the dependability requirements: selecting the dependability requirements that have to be pursued in building the computing system, based on known or assumed goals for the part of the world that is directly affected by the computing system;
2. Designing and implementing the computing system so as to achieve the dependability required. However, this step is hard to implement since the system reliability cannot be satisfied

simply from careful design. Some techniques can be used to help to achieve this goal, such as using fault injection to evaluate the design process.

3. Validating a system: gaining confidence that a certain dependability requirement/goal has been attained.

This paper will address these main concerns through an attempt to provide an in-depth view over modern computing directions and paradigms, which we consider to be representative for the efforts involved in improving overall dependability.

1.1 Motivations

We have listed some of the applications of dependable computing systems as linked to activities that take place in special environments, such as deep underwater or outer space. At a very first sight, these applications would appear specific enough to not encourage a specific design for dependability approach in computing. However, evidence suggest this is hardly the case; on the contrary, it is difficult to imagine a domain left unconquered by computer systems during times when industrial, transport, financial services and others do rely heavily on accurate computer operation at any given moment. If computer innacuracies could be more easily overlooked at home, professional environments cannot accept such missbehaviors.

Yet the recent history of computing provides evidence that dependability is not a *sine qua non* feature. During their life cycle, electronic devices constantly suffer a number of influences that manifest predominantly over transient regimes, which in turn introduce a variety of errors unified in the literature under the name of *transient faults*, *soft errors* or *single event upsets* (SEUs). The rate electronic devices are affected with is known under the term of *soft error rate* or simply *SER* and is measured in fails per unit time. Because it relies on transient phenomena due to changing states and logical values, digital electronics makes up for a special category that is also affected by soft errors. No matter the name they are referred under, these errors affect the computing processes and are due to electromagnetic noise and/or external radiations rather than design or manufacturing flaws [28].

One cause at the origin of soft fails affecting digital devices is known to be due to radioactive decay processes. Radioactive isotopes, widely used for a range of purposes, might contaminate semiconductor materials leading to soft errors; evidence is available throughout the literature, both by empirical observations and experimental results [20]. Consequently, cosmic rays, containing a broad range of energized atomic/subatomic particles may lead to the appearance of soft fails.

Computers therefore are susceptive to soft errors, an issue that will potentially become essential with the advent of emerging technologies. As acknowledged by the International Technology Roadmap for Semiconductors (ITRS), issued at the end of 2004 [43], the microelectronics industry faces a challenging task in going to and beyond 45nm scale in order to address “beyond CMOS” applications. Scaling down the technology will enable an extremely large number of devices to be integrated onto the same chip. However, the great challenge will be to ensure the new devices will be operational at this scale [6], since they will exhibit a sensitive behavior to soft fails. In order to address the negative effects brought by technology scaling, it is to be expected that significant control resources will need to be implemented [3].

Another challenging aspect concerning emerging technologies is to match the newly developed device technologies with new system architectures, a synergistic/collaborative development of the two being seen as likely to be very rewarding. The potential of biologically-inspired and quantum computing architectures is acknowledged by the ITRS report on emerging technologies [43] (see Figure 1). This paper will investigate the relevance of soft fails and attempt to provide means of harnessing their negative effects on modern computing in the context of biologically-inspired and quantum computing architectures.

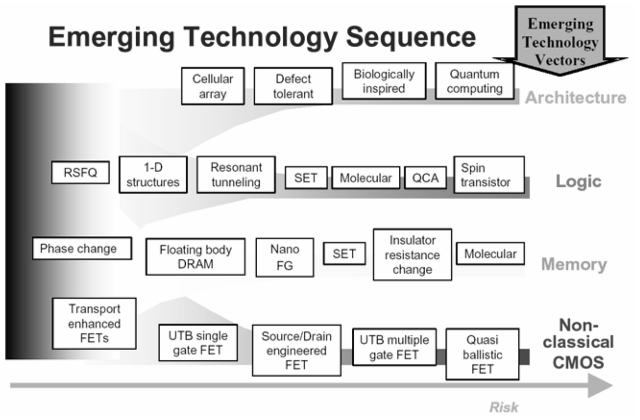

Figure 1: Bio-inspired and quantum computing are acknowledged as architectural technology vectors in emerging technologies [43]

1.2 Paper Outline

This paper is structured as follows. Section 2 will address the first main concern, that is, *specifying and selecting dependability requirements* that will have to be pursued when building a computational platform. Parameters that describe and quantify dependability attributes, such as reliability, will be introduced, with a highlight on their accepted models and their issues. A particular consideration will be given to the failure rate parameter, which is the basis of all reliability analyses.

Section 3 will approach some of the means for design for dependability; it will therefore elaborate upon two emerging technology vectors, as seen by the ITRS report [43], which define two novel architectures, namely biologically-inspired (or bio-inspired) and quantum computing. We will introduce two projects and their corresponding architectures, called Embryonics (as a biologically-inspired computing platform) and *QUERIST* (as a quantum computing platform designed to allow and study error injection). These two architectures are representative for the coming age of nano-computing, where computational processes take place as encoded at the very inner core level of matter, be it semiconductor material (for nanoelectronics, targetted here by the Embryonics project) or atomic scale dynamics (for quantum computing, targetted here by the *QUERIST* project). This section will then introduce dependability aspects within bio-inspired computing (the Embryonics project being investigated in SubSection 3.1) and within quantum computing (the *QUERIST* project being investigated in SubSection 3.2).

Finally, Section 4 will present the conclusions and prospects for designing emerging technology dependable computing systems, as we see them.

2. DEPENDABILITY ATTRIBUTES

An important dependability attribute for any given system lies in its capacity to operate reliably for a given time interval, knowing that normal operation was delivered at initial time [8]. Reliability functions are modelled as exponential functions of parameter λ, which is the failure rate. The reliability of a system is the consequence of the reliability of all of its subsystems. The heterogeneity of the system leads to a difficult quantitative assessment of its overall reliability; moreover, estimating the reliability functions is further made difficult because formal rigour is not commercially available, this being kept under military mandate [44].

The failure rate for a given system can be modelled as a function of the failure rates of its individual subsystems, suggestions being present in the MIL-HDBC-217 document, which is publicly available [44]. However, this document has been strongly criticized for its failure rate estimations based on the Arrhenius model, which relates the failure rate to the operating temperature:

$$\lambda = Ke^{-E/K_B T} \qquad (1)$$

where K is a constant, K_B is Boltzmann's constant, T is the absolute temperature and E is the "activation energy" for the process [18]. Quantitative values for failure rates show significant differences between those predicted using MIL-HDBC-217 and those from testing real devices (see Figure 2). There are two conclusions that can be drawn from this:

1. quantitative estimations for failure rate values are strongly dependant on the quality of information used; unfortunately, current reliable information about electronic devices is known to be lacking [44];
2. despite differences between predicted and real values, the MIL-HDBC-217 methodology can be useful for qualitative analyses in order to take decisions regarding sub-system parts that should benefit from improved designs.

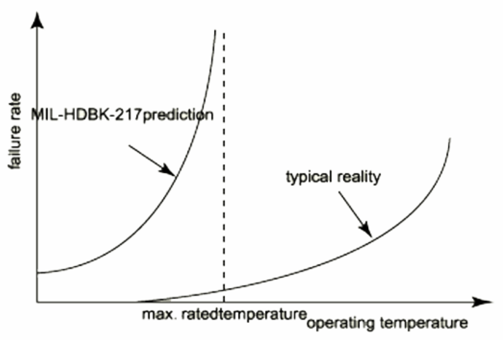

Figure 2. Predicted vs real failure rates plotted against temperature [18]

So far the failure rate of digital devices has been considerred as due to internal causes. However, this is not always the case, soft fails being equally present due to the aggressive influences of the external environment, which also have to be modelled [22]. The external envirnment features highly dynamic changes in its parameters, which will eventually affect the normal operation of digital devices that lack sufficient protection or ability to adapt. Ideally, computing devices would behave in a consistent and accurate manner regardless of fluctuations in environmental parameters. This is either a consequence of soft error mitigation techniques or due to flexible hardware/software functionality that allow the system as a whole to adapt to environamental changes and tolerate induced faults.

While certain soft error mitigation techniques are available, the technology scaling towards nanoelectronics affects their efficiency by integrating a larger number of devices per chip (which requires a larger amount of redundant/control logic or other measures), which feature, at the same time, smaller dimensions (which renders an electronic device much more senzitive to the influence of stray energetic particles that reach it as part of cosmic rays). Both aspects are involved in the development of the two emerging technology vectors mentioned in SubSection 1.1, although having slightly different motivations: while the nature of the quantum environment prohibits precise computation in the absence of fault tolerance techniques, such techniques are targetted by bio-inspired computing as means of improving the dependability of a computing platform.

2.1 Bio-Inspired Computing

If living beings may be considered to fulfill computational tasks, then Nature is the ultimate engineer: each of the living beings exhibit solutions that were successfully tested and refined in such ways human engineers will never afford. One reason is time: the testing period coinciding with the very existence of life itself. Another reason is variety and complexity: Nature has found and adapted a variety of solutions to address complex survivability issues in a dynamically changing environment. No matter how Nature approached the process of evolution, engineering could perhaps benefit most from drawing inspiration from its mechanisms rather from trying to develop particular techniques.

Bio-inspired computing is not a new idea. John von Neumann was preoccupied to design a machine that could replicate itself and was quite interested in the study of how the behavior of the human brain could be implemented by a computer [13][14]. He also pioneered the field of dependable computing by studying the possibility of building reliable machines out of unreliable components [15]. Unfortunately, the dream of implementing his self-reproducing automata could not become true until the 1990s, when massively programmable logic opened the new era of reconfigurable computing.

But when trying to adapt nature's mechanisms in digital devices, it becomes most evident that biological organisms are rightfully the most intricate structures known to man. They continuously demonstrate a highly complex behavior due to massive, parallel cooperation between huge numbers of relatively simple elements, the cells. And considering 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, it seems nature is the closest spring of inspiration for designing dependable, fault tolerant systems.

Investigating the particularities of natural systems, a taxonomy of three categories of processes can be identified [32]:

1. *Phylogenetic processes* constitute the first level of organization of the living matter. They are concerned with the temporal evolution of the genetic heritage of all individuals,

therefore mastering the evolution of all species. The phylogenetic processes rely on mechanisms such as recombination and mutation, which are essentially nondeterministic; the error rate ensures here nature's diversity.

2. *Ontogenetic processes* represent the second level of organization of the living matter. They are also concerned with the temporal evolution of the genetic heritage of, in this case, a single, multicellular individual, therefore mastering an individual's development from the stage of a single cell, the zygote, through succesive cellular division and specialization, to the adult stage. These processes rely on deterministic mechanisms; any error at this level results in malformations.
3. *Epigenetic processes* represent the third level of organization of the living matter. They are concerned with the integration of interactions with the surrounding environment therefore resulting in what we call learning systems.

This taxonomy is important in that it provides a model called POE (from Phylogeny, Ontogeny and Epigenesis) that inspires the combination of processes in order to create novel bio-inspired hardware (see Figure 3). We believe this is also important from a dependability engineering perspective, for the following reasons:

1. Phylogenetic processes were assimilated by modern computing as evolutionary computation, including genetic algorithms and genetic programming. The essence of any genetic algorithm is the derivation of a solution space based on recombination, crossover and mutation processes that spawn a population of individuals, each encoding a possible solution. One may consider that each such step, with the exception of discovering the solution, is equivalent to a process of error injection, which in turn leads to wandering from the optimal solution (or class of solutions). However, genetic algorithms prove to be successful despite this error injection, the fitness function being responsible for the successful quantification of the significance of the "error". Therefore genetic computation is *intrinsicaly resilient* to faults and errors, largely due to the fact that they are part of the very process that generates the solutions.
2. Ontogenetic processes have been implemented in digital hardware with modular and uniform architectures. Such an architecture enables the implementation of mechanisms similar to the cellular division and cellular differentiation that take place in living beings [31]. These mechanisms bring the advantage of *distributed and hierarchical fault tolerance* strategies: the uniformity of the architecture also makes any module to be universal, that is, to be able to take over the role of any other damaged module.
3. Epigenetic processes were assimilated by modern computing mainly as artificial neural networks (or ANNs) as inspired by the nervous system, and much less as inspired by the immune or endocrine systems from superior multicellular living beings. ANNs are known to have a generalization capacity, that is, to respond well even if the input patterns are not part of the patterns used during the learning phase. This means that ANNs possess a certain ability to tolerante faults, whether they manifest at the inputs or inside their intenal architecture.

With the advent of field programmable logic (of which the most salient representative are the FPGAs) it is now possible to change hardware functionality through software, thus allowing information to govern matter in digital electronics. This is not dissimilar to what happens in nature: information coded in DNA affects the development of an organism. A special kind of such digital devices that change dynamically their behavior are known as evolvable or adaptive hardware; they are bio-inspired computing systems whose behaviors may change according to computational targets, or, if harsh or unknown environments are to be explored, for the purpose of maximizing dependability.

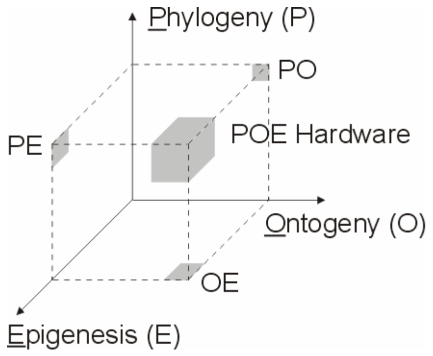

Figure 3. The POE model of bio-inspired systems [32]

2.2 Quantum Computing

Error detection and correction techniques are vital in quantum computing due to the destructive effect of the environment, which therefore acts as an omnipresent error generator. Error detection and correction must provide a safe recovery process within quantum computing processes through keeping error propagation under control. Without such dependability techniques there could be no realistic prospect of an operational quantum computational device [19].

There are two main sources of errors: the first is due to the erroneous behavior of the quantum gate, producing the so-called *processing errors*; the second is due to the macroscopic environment that interacts with the quantum state, producing the *storing* and *transmitting errors*.

The consistency of any quantum computation process can be destroyed by innacuracies and errors if the error probability in the basic components (qubits, quantum gates) excedes an *accuracy threshold*. This is a critical aspect since the microscopic quantum states are prone to frequent errors.

The main error source is the *decoherence* effect [16]. The environment is constantly attempting to measure the sensitive quantum superposition state, a phenomenon that cannot be avoided technologically since it is not (yet) possible to isolate them perfectly. The superposition state will decay through measuring and will therefore become a projection of the state vector onto a basis vector (or eigenstate). The most insidious error, however, appears when decoherence affects the quantum amplitudes without destroying them; this is similar to small analog errors. Issues stated above are solved, on one hand, through intrinsic fault tolerance by technological implementation (topological interactions [1]) and, on the other hand, by error correcting techniques at the unitary (gate network) level. We will focus on the error detecting and correcting techniques, which are difficult to approach due to quantum constraints: the useful state can neither be observed (otherwise it will decohere), nor can it be cloned.

2.2.1 Background

As expressed in bra-ket notation [16], the qubit is a normalized vector in some Hilbert space $\mathcal{H}^2$, $\{|0\rangle, |1\rangle\}$ being the orthonormal basis: $|\psi\rangle = a_0|0\rangle + a_1|1\rangle$ (a_0, $a_1 \in \mathbb{C}$ are the so-called quantum amplitudes, representing the square root of the associated measurement probabilities for the eigenstates $|0\rangle$ and $|1\rangle$ respectively, with $|a_0|^2 + |a_1|^2 = 1$). Therefore, the qubit can be affected by 3 types of errors:

Bit flip errors are somewhat similar to classical bit flip errors. For a single qubit things are exactly the same as in classical computation: $|0\rangle \mapsto |1\rangle$, $|1\rangle \mapsto |0\rangle$. For 2 or more qubits, flip errors affecting the state may modify it or leave it unchanged. For instance, if we consider the so-called cat state $|\psi\rangle_{Cat} = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ [19], and the first qubit is affected by a bit flip error, the resulting state will be $|\psi\rangle_{Cat} \mapsto \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle)$. But, if both qubits are affected by bit flips, there will be no change in the state: $|\psi\rangle_{Cat} \mapsto \frac{1}{\sqrt{2}}(|11\rangle + |00\rangle) = |\psi\rangle_{Cat}$.

Phase errors affect the phase of one of the qubit's amplitudes and is expressed as $|0\rangle \mapsto |0\rangle$, $|1\rangle \mapsto -|1\rangle$. This type of error is very dangerous, due to its propagation behavior but it only makes sense when dealing with superposition states. If we consider an equally weighted qubit superposition state and inject a phase error, this results in $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \mapsto \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

There is a strict correspondence between bit flip and phase error types due to the way they map onto Hilbert spaces with the same dimension but different basis. The bit flip is an error from the $\mathcal{H}^2$ space with basis $\{|0\rangle, |1\rangle\}$, whereas the phase error appears in the same space with basis $\left\{\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right\}$ or $\{|+\rangle, |-\rangle\}$.
The space basis conversion, in this case, is made by applying the Hadamard transform; Figure 4 shows an example of transforming a bit flip error into a phase error (A, and vice versa (B.

Small amplitude errors: amplitudes a_0 and a_1 of the quantum bit can be affected by small errors, similar to analog errors. Even if such an error does not destroy the superposition and conserves the value of the superposed states, small amplitude errors could accumulate over time, eventually ruining the computation. In order to avoid this situation, specific methodologies for digitizing small errors are used to reduce them to a non-fault or a bit-flip [19].

Due to the quantum physics laws, fault tolerance techniques have to comply with the following computational constraints:

– *The observation destroys the state.* Since observation is equivalent to measurement, this leads to destroying the useful state superposition.
– *Information copying is impossible.* Quantum physics renders the cloning of a quantum state impossible, meaning that a quantum state cannot be copied correctly. Therefore quantum error correction must address the following problems:

Non-destructive measurement. Despite the first constraint it is necessary to find a way to measure the encoded information without destroying it. Because the encoded state cannot be measured directly, one needs to properly prepare some scratch (ancilla) qubits, which can then be measured.

Fault-tolerant recovery. Due to the high error rate in quantum computational devices, it is likely that the error recovery itself will be affected by errors. If the recovery process is not fault-tolerant, then any error coding becomes useless.

Phase error backward propagation. If we consider the *XOR* gate from Figure 5(A, a flip error affecting the target qubit (*b*) will propagate backwards and also affect the source qubit. This is due to the gate network equivalence from Figure 5(B and the basis transformation described by Figure 4.

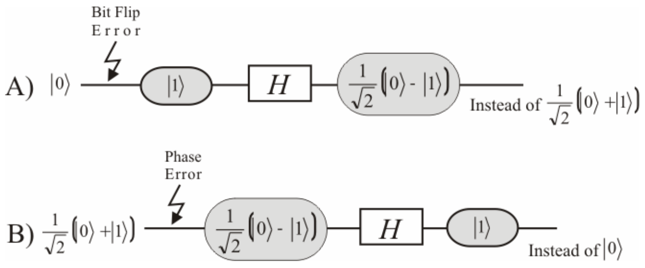
Bit Flip Error
A) $|0\rangle$
$|1\rangle$
H
$\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$
Instead of $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
Phase Error
B) $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
$\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$
H
$|1\rangle$
Instead of $|0\rangle$

Figure 4. Correspondence between bit flip and phase errors

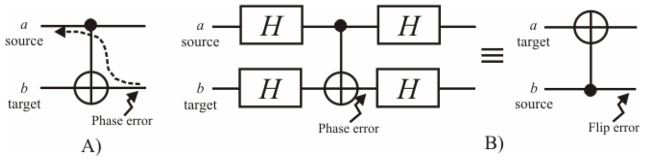

Figure 5. (A The backward propagation of a phase error for the *XOR* gate; (B Gate network equivalence

In order to deal with the problems described the next strategies have to be followed:

Digitizing small errors. The presence of small errors is not a major concern, as they can be digitized using a special technique based on measuring auxiliary (ancilla) qubits [19].

Ancilla usage. Since qubit cloning is impossible, a majority voting strategy is difficult to implement. However, by using ancilla qubits, the eigenstate information can be duplicated inside the existing superposition, resulting in the entanglement of the ancilla with the useful data. Because any measurement performed on the ancilla could have repercussions on the useful qubits, the appropriate strategy will employ special coding for both data qubits and ancilla (data errors only will be copied onto the ancilla), followed by the computation of an error syndrome, which has to be obtained through measuring the ancilla (see Figure 6).

Avoiding massive spreading of phase errors. As shown previously, a phase error on the target qubit will propagate on all source qubits. The solution is to use more ancilla qubits as targets, so that no ancilla qubit is used more than once.

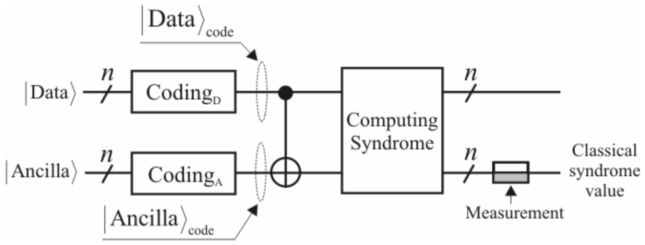

Figure 6. Fault-tolerant procedure with ancilla qubits

Ancilla and syndrome accuracy. Setting the ancilla code to some known quantum state could be an erroneous process. Computing the syndrome is also prone to errors. Hence, on one hand, one has to make sure that the ancilla qubits are in the right state by verifying and recovering them if needed; on the other hand, in order to have a reliable syndrome, it must be computed repeatedly.

Error recovery. As the small errors can be digitized (therefore, they are either corrected or transformed into bit flip errors), the recovery must deal only with bit flip and phase errors. A state that needs to be recovered is described by:

$$a_0|0\rangle + a_1|1\rangle \xrightarrow{error} \begin{cases} a_0|0\rangle + a_1|1\rangle & \text{if no error occurs} \\ a_1|0\rangle + a_0|1\rangle & \text{for a flip error} \\ a_0|0\rangle - a_1|1\rangle & \text{for a phase error} \\ a_1|0\rangle - a_0|1\rangle & \text{for both flip and phase errors} \end{cases}.$$

Correcting a bit flip error means applying the negation unitary transformation $U_N = \sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ to the affected qubit. To correct phase and combined errors, the following unitary operators will have to be applied respectively: $U_Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$, $U_Y = U_N \cdot U_Z = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$.

2.2.2 Quantum Error Correcting Codes

Quantum error coding and correcting (QECC) is performed with special coding techniques inspired from the classic Hamming codes. The classical error coding is adapted so that it becomes suitable for the quantum strategy, allowing only the ancilla qubits to be measured.

The state-of-the-art in QECC is represented by the stabilizer encoding, a particular case being the Steane codes (the Shor codes may also be used [29]). Steane's 7-qubit code is a single error correcting code inspired from classical Hamming coding and can be adapted for ancilla coding as well. Therefore it cannot recover from two identical qubit faults, but it can recover from a bit flip a phase flip. The Steane 7-qubit coding of $|0\rangle$ and $|1\rangle$ consists of an equally weighted superposition of all the valid Hamming 7-bit words with an even and odd number of 1s, respectively:

$$\begin{aligned} |0\rangle_S &= \frac{1}{2^{3/2}} \sum_{even_{(u_0u_1,u_2u_3c_0c_1c_2)}} |u_0u_1u_2u_3c_0c_1c_2\rangle \\ &= \frac{1}{2^{3/2}} \big(|0000000\rangle + |0010111\rangle + |0101110\rangle + |0111001\rangle + \\ &+ |1001011\rangle + |1011100\rangle + |1100101\rangle + |1110010\rangle \big) \end{aligned} \tag{2}$$

$$\begin{aligned} |1\rangle_S &= \frac{1}{2^{3/2}} \sum_{odd_{(u_0u_1,u_2u_3c_0c_1c_2)}} |u_0u_1u_2u_3c_0c_1c_2\rangle \\ &= \frac{1}{2^{3/2}} \big(|1111111\rangle + |1101000\rangle + |1010001\rangle + |1000110\rangle + \\ &+ |0110100\rangle + |0100011\rangle + |0011010\rangle + |0001101\rangle \big) \end{aligned} \tag{3}$$

Applying the Steane coding on an arbitrary given quantum state $|\psi\rangle = a_0|0\rangle + a_1|1\rangle$ transforms it into $|\psi\rangle_S = a_0|0\rangle_S + a_1|1\rangle_S$. This code was designed to correct bit-flip errors, but by changing the basis (through a Hadamard transform) the phase error transforms into a bit flip error, which can then be corrected:

$$\begin{aligned} |\bar{0}\rangle_S &= H \cdot |0\rangle_S = \frac{1}{\sqrt{2}} \big(|0\rangle_S + |1\rangle_S \big) \\ |\bar{1}\rangle_S &= H \cdot |1\rangle_S = \frac{1}{\sqrt{2}} \big(|0\rangle_S - |1\rangle_S \big) \end{aligned} \tag{4}$$

2.2.3 Fault Tolerance Methodologies

Quantum error-correcting codes exist for r errors, $r \in \mathbb{N}$, $r \geq 1$. Therefore a *non-correctable error* occurs if a number of $r+1$ errors occur simultaneously before the recovery process.

If the probability of a *quantum gate error* or *storage error* in the time unit is of order ξ, then the probability of an error affecting the processed data block becomes of order ξ^{r+1}, which is negligible if r is sufficiently large. However, by increasing r the safe recovery also becomes more complex and hence prone to errors: it is possible that $r+1$ errors accumulate in the block before the recovery is performed.

Considering the relationship between r and the number of computational steps required for computing the syndrome is polynomial of the order r^p. It was proven that in order to reduce as much as possible the error probability r must be chosen so that $r \sim e^{-1}\xi^{-\frac{1}{p}}$ [7][19]. By consequence, if attempting to execute N cycles of error correction without any r+1 errors accumulating before the recovery ends, then $N \sim \exp\left(\xi^{-\frac{1}{p}}\right)$. Therefore the accuracy degree will be of the form $\xi \sim (\log N)^{-p}$, which is better than the accuracy degree corresponding to the no-coding case, $\xi \sim N^{-1}$. However, there exists a $N_{\max}$ so that if $N > N_{\max}$ then non-correctable error becomes likely, which limits the length of the recovery process. Given the extremely large number of gates employed by a quantum algorithm implementation, $N_{\max}$ also has to be very large; for Shor's algorithm $N_{\max}$ must be higher than $3 \cdot 10^9$ [30].

As shown in Figure 7, the required accuracy degree approaches today's technological limits (tipically 10^{-3} for p=4) after N=10^5. For a fault tolerant encoding solution for Shor algorithm implementation this should have happened after N=10^9 [19][34].

Additional fault tolerance must be employed in order to preserve reliable quantum computation over an arbitrary number of computational steps. Concatenated coding represents one such technique, which improves the reliability by shaping the size of the code blocks and the number of code levels. It is also resource demanding and vulnerable to correlated errors [19][37].

Another approach, replacing the concatenated codes, is based on Reconfigurable Quantum Gate Arrays (RQGAs) [34][37], which are used for configuring ECC circuits based on stabilizer codes [7][33]. By using a *quantum* configuration register for the RQGA (i.e. a superposition of classical configurations), the reconfigurable circuit is brought to a state where it represents a simultaneous superposition of distinct ECC circuits. After measuring the configuration register, only one ECC circuit is selected and used; if k distinct ECC circuits were superposed and the gate error rate is ξ , then the overall gate error probability becomes ξ^k (see Figure 8). As a result, the accuracy threshold value for the RQGA solution clearly dominates the technological accuracy limit, as shown in Figure 9 [37].

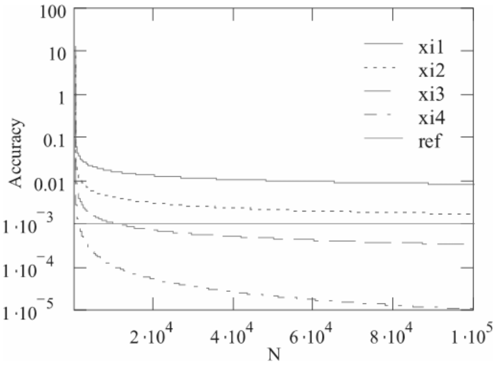

Figure 7. Accuracy plots: p=3 for xi_1, p=4 for xi_2, p=5 for xi_3; xi_4 for no-coding, *ref* is the reference accuracy (i.e. the accuracy allowed by today's state of the art technology)

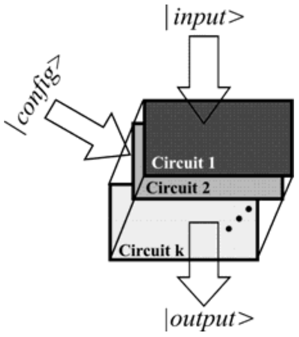

Figure 8. A quantum configuration register acts as a superposition of *k* distinct circuits sharing the same input state and the same output qubits

3. DEPENDABLE SYSTEM DESIGN

In order to model the erroneous behavior of a device of system it is necessary to understand the causality of phenomena concerned. A defect affecting a device from a physical point of view is called a fault, or a fail. Faults may be put in evidence through logical misbehavior, in which case they transform into errors. Finally, errors accumulating can lead to system failure [8]. The fault-error-failure causal chain is essential to developping techniques that reduce the risk of error occurrence, even in the presence of faults, in order to minimize the probability of a system failure, and can be architecture specific. We will elaborate next on techniques used by a bio-inspired and by a quantum computing platform.

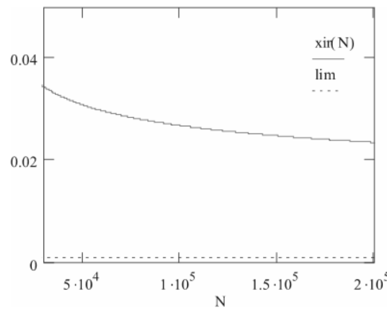

Figure 9. Evolution of accuracy threshold value for RQHW stabilizer codes (*xir*); the technological accuracy limit (*lim*) is also provided for a relevant comparison

3.1 The Embryonics Approach

Several years before his untimely death John von Neumann began developping a theory of automata, which was to contain a systematic theory of mixed mathematical and logical forms, aimed to a better understanding of both natural systems and computers [14]. The essence of von Neumann's message appears to entail the formula "genotype + ribotype = phenotype". He provided the foundations of a self-replicating machine (the phenotype), consisting of its complete description (the genotype), which is interpreted by a ribosome (the ribotype).

Embryonics (a contraction for *embryonic electronics*) is a long term research project launched by the Logic Systems Laboratory at the Swiss Federal Institute of Technology, Lausanne, Switzerland. Its aim is to explore the potential of biologically-inspired mechanisms by borrowing and adapting them from nature into digital devices for the purpose of endowing them with the remarkable robustness present in biological entities [39]. Though perhaps fuzzy at a first glance, analogies between biology and electronics are presented in Table 1 [12][31].

But if we consider that the function of a living cell is determined by the genome, and that a computer's functionality is determined by the operating program, then the two worlds may be regarded as sharing a certain degree of similarity. Three fundamental features shared by living entities are required to be targetted by Embryonics in order to embody the formula "genotype + ribotype = phenotype" into digital hardware:

- *multicellular* organisms are made of a finite number of cells, which in turn are made of a finite number of chemically bonded *molecules*;
- each cell (beginning with the original cell, the *zygote*) may generate one or several daughter cell(s) through a process called *cellular division*; both the parent and the daughter cell(s) share the same genetic information, called the *genome*;
- different types of cells may exist due to *cellular differentiation*, a process through which only a part of the genome is executed.

These fundamental features led the Embryonics project to settle for an architectural hierarchy of four levels (see Figure 10). We will not delve very deep inside the Embryonics'phylosophy, as such details were broadly covered by literature [12][20][23][24] [25][40]; we will, however, introduce each of the four levels in order to be able to see how this bio-inspired platform fits modern design for dependability efforts.

Table 1. Analogies present in Embryonics [12]

Biology	Electronics
Multicellular organism	Parallel computer systems
Cell	Processor
Molecule	FPGA Element

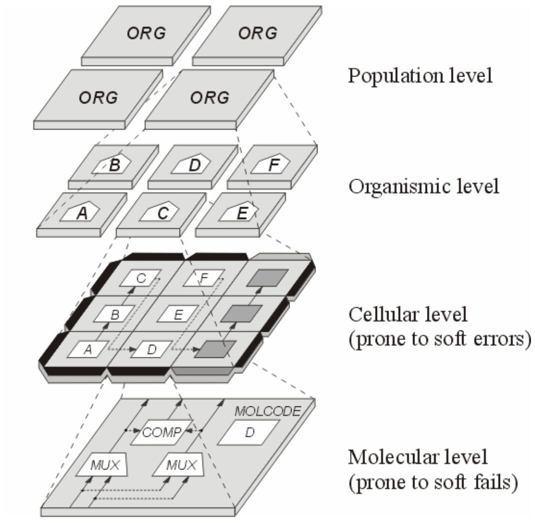

Figure 10. Structural hierarchy in Embryonics [12]

The upmost level in Embryonics, bearing a certain similarity to what is found in nature, is the population level, composed of a number of organisms. One level down the hierarchy constitutes the organismic level, and corresponds to individual entities in a variety of functionalities and sizes. Each of the organisms may be further decomposed into smaller, simpler parts, called cells, which in turn may be decomposed in 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 (see Table 1).

An extremely valuable consequence of the Embryonics architecture is that each cell is "universal", containing a copy of 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) [12]. Self-replication may be of great interest in the nanoelectronics era, where extremely large areas of programmable logic will probably render any centralized control very inefficient. Instead, the self-replication mechanism implemented in Embryonics will allow the initial colonization of the entire programmable array in a decentralized and distributed manner. Figure 11 presents an example of such colonization. At initial time the configuration bitstream (containing the genome) enters the bottom left corner of a programmable array and, at each clock cycle, the genome is pushed through and partitions the programmable space accordingly.

From a dependability standpoint, the Embryonics hierarchical architecture offers incentives for an also hierarchical self-repair strategy. Because the target applications are those in which the failure frequency must be very low to be "acceptable", two levels of self-repair are offered: at the molecular level (programmable logic is susceptible to soft fail occurrences) and at the cellular level (soft fails manifest at this level as soft errors).

Let us consider an example of a simple cell made of 3 lines and 3 columns of molecules, of which one column contains spare molecules. If a fault occurs inside an active cell, it can be repaired through transferring its functionality toward the appropriate spare molecule, which will become active (see Figure 12).

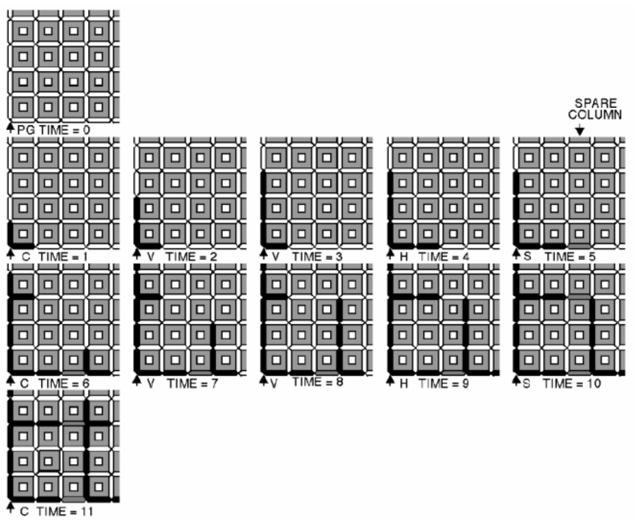

Figure 11. Space colonization in Embryonics [11]

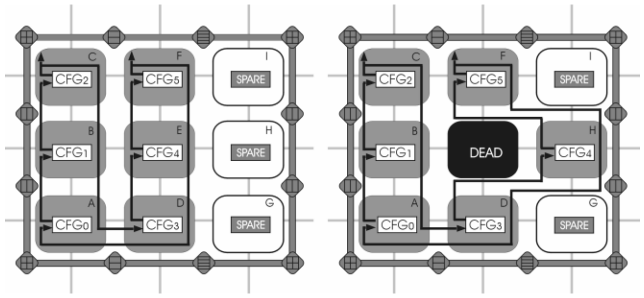

Figure 12. Self-repair at the molecular level: faulty molecule *E* is replaced by spare molecule *H*, which becomes active [39]

The self-repair process at molecular level ensures the fault recovery as long as there are spare molecules left for repair. However, it is possible for a cell to experience a multiple error, in which case the self-repair mechanism at the molecular level can no longer reconfigure the inside of the cell successfully. If such a situation arises, then a second self-repair strategy is trigerred at a higher level. The cell will "die", therefore trigerring the self-repair at the cellular level, the entire column containing the faulty cell (cell *C* in this example) being deactivated, its role being taken by the nearest spare column to the right (see Figure 13).

A critique that could be addressed to the current Embryonics design would be its strategy of self-repair at the higher, cellular level: in case of a faulty cell, an entire column containing that cell will be deactivated, its role being transferred to the first available column of spares to the right (see Figure 13). There are two points in which this strategy could benefit:

1. Instead of deactivating a whole column of cells, it would be more efficient to only deactivate the faulty cell only (see Figure 14). The resources affected by the role transfer would be greatly reduced (one cell versus an entire column), coupled with the fact that particle flux generating soft fails is unlikely to be homogeneous and isotrope. This means regions assimilable more likely to cells rather than entire column of cells would be more affected by soft fails, not to mention that during genetic data transfer (required by taking over the role of the faulty cell) there is a greater risk of enduring a new soft fail (moving data is much more sensitive to soft fails than static data) [5][10].

2. Such a strategy would be consistent with that used for the self-repair at the molecular level, which would simplify a thorough reliability analysis. Concatenated coding would also seem easier to be implemented and the strategy consistency would mean that concatenated coding would not be limited to a two-level hierarchy [20][21].

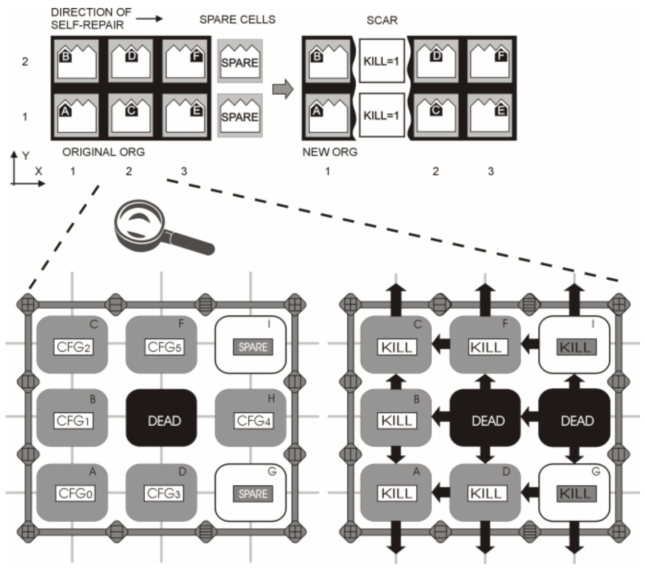

Figure 13. Molecular self-repair failure: the cell "dies" (bottom), triggering the cellular self-repair (top) [39]

We consider a cell of M lines and N columns, being composed of modules of M lines and $n+s$ columns (for instance, the cell presented in Figure 12 consists of a single such module of two active columns and one spare column), of which s are spares. In order to meet certain reliability criteria, it is necessary to know what is the number s of spare columns of molecules that correspond to n columns of active molecules, that is, the horizontal dimensions for such a module. We will not provide a thorough reliability analysis, as this has been done previously [4][17][20][21]; instead, we will analyze the influences of the proposed consistent self-repair strategy at both molecular and cellular levels through the use of logic molecules. Therefore Equation (5) holds:

$$R_{ModRow}(t)=Prob\{no\ fails\}(t)+\sum_{i=1}^{k}Prob\{i\ fails\}(t) \quad N=k(n+s) \tag{5}$$

where $R_{ModRow}(t)$ represents the reliability function for a row within a module. Then, considering the failure rate for one molecule λ, the probability of all molecules (both active and spare) to operate normally in a module's row becomes:

$$Prob\{no\ fails\}(t)=e^{-\lambda(n+s)t} \tag{6}$$

The probability of a row enduring i fails in the active molecules part is the conditional probability of having n-i active molecules operating normally, while a number of s-i spare molecules are ready to activate (that is, they are not affected by errors themselves):

$$Prob\{i\ fails\}(t)=Prob\{i\ fails\ active\}(t)\cdot \cdot Prob\{i\ spares\ ok\}(t) \tag{7}$$

$$Prob\{i\ fails\ active\}(t)=\binom{n}{i}e^{-\lambda(n-i)t}\left(1-e^{-\lambda(n-i)t}\right) \tag{8}$$

$$Prob\{i\ spares\ ok\}(t)=\binom{k}{i}e^{-\lambda it}\left(1-e^{-\lambda(k-i)t}\right) \tag{9}$$

Then the reliability function for an entire cell is the cummulated reliability functions for the total number of modules:

$$R_{Cell}(t)=\left[R_{ModRow}(t)\right]^{MN/_{n+s}} \tag{10}$$

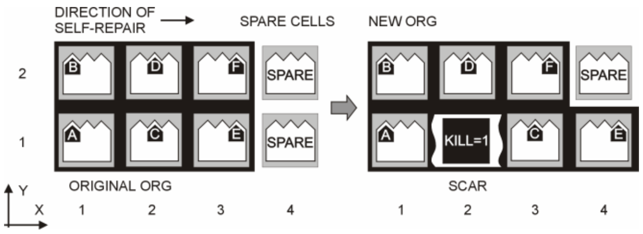

Figure 14. Proposed reconfiguration strategy at the cellular level

A self-repair strategy that conserves the consistency between the molecular and the cellular level would allow for a more straightforward reliability analysis. Basically, it would be sufficient to substitute dimension parameters in Equations (5)-(10) with those approapriate to the analysis of an organism instead of a cell. To illustrate this, we will consider an organism of M^* lines and N^* columns, being composed of modules of M^* lines and n^*+s^* columns, of which s^* are spares; we will also use the organism partitioning into modules, similar to the partitioning of cells used before. Therefore Equation (5) transforms into Equation (11):

$$R_{CellMR}(t)=Prob^*\{no\ fails\}(t)+\sum_{i=1}^{k}Prob^*\{i\ fails\}(t) \quad N^*=k^*\left(n^*+s^*\right) \tag{11}$$

where $R_{CellMR}(t)$ represents the reliability function for a row of cells within an organism module. In this case, the significance of the terms will be as follows:

$$Prob^*\{no\ fails\}(t)=\left[R_{Cell}(t)\right]^{n^*+s^*} \tag{12}$$

While Equation (7) continues to hold under the form of Equation (13), the significance of its terms will change according to the dimensions at the cellular level:

$$Prob^{*}\{i\,fails\}(t) = Prob^{*}\{i\,fails\ active\}(t)\cdot \cdot Prob^{*}\{i\,spares\ ok\}(t) \tag{13}$$

$$Prob^{*}\{i\,fails\ active\}(t) = \binom{n^{*}}{i} R_{Cell}^{n^{*}-i}(t)\left(1-R_{Cell}^{i}(t)\right) \tag{14}$$

$$Prob^{*}\{i\,spares\ ok\}(t) = \binom{k^{*}}{i} R_{Cell}^{i}(t)\left(1-R_{Cell}^{k^{*}-i}(t)\right) \tag{15}$$

Finally, similar to Equation (10), the reliability function for an entire organism is the cummulated reliability functions for the total number of its modules:

$$R_{Org}(t) = \left[R_{CellMR}(t)\right]^{M^{*}N^{*}/n^{*}+s^{*}} \tag{16}$$

Equations (5) to (16) provide the basics for a thorough reliability analysis for the proposed, uniform strategy of hierarchical reconfiguration, as opposed to the analysis provided by [21], which specifically targetted the current Embryonics architecture. Despite having settled the reliability model, both analyses are incomplete, in that the failure rate parameter is missing, which makes a precise, quantitative dependability target difficult to meet. However, a reliability analysis is still valuable from a qualitative point of view, allowing a direct comparison of different systems.

3.2 The *QUERIST* Approach

In order to deal with errors induced by the constant influence of the external environment upon computational processes, the following assumptions were made: errors appear randomly, are uncorrelated (neither in space, nor in time), there are no storage errors, and there are no leakage phenomena involved [19].

Classical HDL-based fault injection methodologies can be mapped to simulating quantum circuits without intervention provided that the new error and fault models are taken into account [35]. Of course, efficiency criteria require that they be adapted to one of the available efficient simulation frameworks [36][38][41]. *QUERIST* (from *QUantum ERror Injection Simulation Tool*) is the name of such a project, fostering simulated fault injection techniques in quantum circuits [34]. Similar to classical computation, simulated fault injection is used in order to evaluate the employed FTAMS (Fault Tolerance Algorithms and Methodologies) [26][27].

An overview of the *QUERIST* project is presented in Figure 15. The three cycles of initialization, simulation, and data computation are common to both classical and quantum approaches. The first cycle takes the quantum circuit HDL description as an input. Two abstract inputs are considered, the HDL model and the assumed error model; the first influences how the HDL description is presented, while the second one dictates the test scenario by defining the start/stop simulation states (since qubits are equally prone to error, all the signals must be observed). HDL modeling of quantum circuits in order to attain efficient simulation is discussed in [34][35][36][38].

The outputs of the first cycle, which are also inputs for the simulation cycle, consist of a test scenario and an executable HDL model with the corresponding entanglement analysis, dictated by the bubble-bit encoded quantum states [36][38]. The output of the second cycle consists of time diagrams for all qubits, from the start to the stop states. Useful information, extracted from the raw, bubble-bit-represented, qubit traces are compared to correct qubit values, the result being the probabilistic accuracy threshold value, in the third cycle. The initialization and simulation cycles depend on specific aspects of quantum circuit simulation [35]. The data processing cycle is independent from the specific simulation framework and is aimed at determining the *accuracy threshold* as the main reliability measure that also defines the feasibility of the quantum circuit implementations.

Suppose that, at simulation time t we observe signals $\{s_0, s_1, ..., s_n\}$. In our analysis, s_i is the state observed during non-faulty simulation, so for the same state in a faulty environment we will have the state s_i^{*}.

For validation of the quantum *FTAM*s, we need to compare s_i with s_i^{*}. This can be done by using operator $dif\left(s_i, s_i^{*}\right)$. This means that the total number of overall state errors at simulation time t is $e_t = \sum_{i=0}^{n-1} dif\left(s_i, s_i^{*}\right)$. The error rate on the overall observed states at moments $t_0,\ t_1, ...,\ t_{m-1}$ will be given by $\xi_{sim} = \frac{1}{m}\sum_{j=0}^{m-1} e_{t_j}$.

The used *FTAM*s are only valid if the relationship between the experimental ξ_{sim} and the assumed singular error rate ξ is of the order $\xi_{sim} \sim \xi^{2}$ [19].

4. CONCLUSIONS

This paper presented arguments in favor of two novel computing architectures for the purpose of addressing the challenges raised by the forthcoming nanoelectronics era. Distributed self-testing and self-repairing will probably become a must in the next years as centralized control logic is expected to become unable to harness the extremely large number of devices, all equally prone to errors, that will be integrated onto the same chip. Bio-inspired computing brings valuable techniques that explore the potential of massively parallel, distributed computation and fault-tolerance that will likely provide an essential help to jumpstart new nanoelectronic architectures. As one of the representatives of bio-inspired computing, the Embryonics project presents a hierarchical architecture that achieves fault tolerance through implementing an also hierarchical reconfiguration. A similar approach for maximizing fault tolerance is present in quantum computing, the *QUERIST* project; even if bio-inspired and quantum computing may seem dissimilar at a first glance, they both achieve fault tolerance by adapting the same techniques from classical computing and using essentially the same error model.

Nanoelectronics will potentially change the way computing systems are designed, not only because of the sheer number of devices that will coexist onto the same chip, but also because of the sensitivity of these devices.

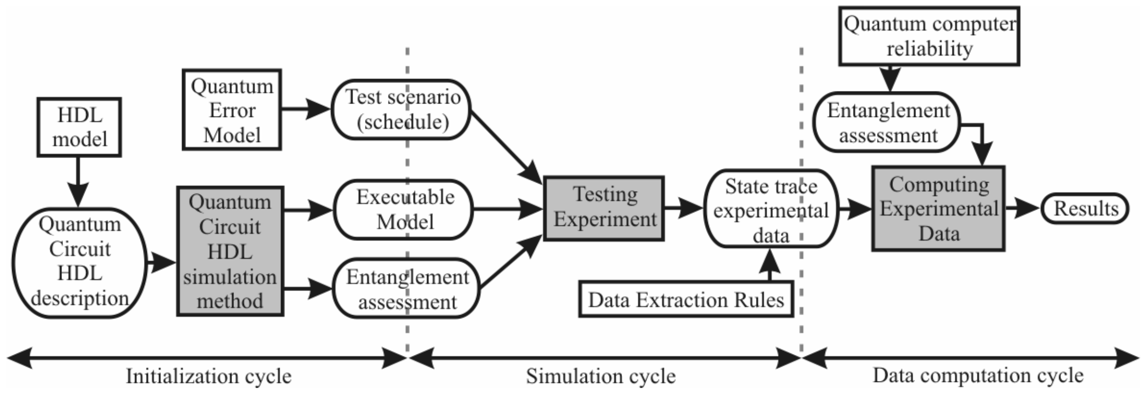

Figure 15. An overview of the *QUERIST* project

Therefore, if nanoelectronics is to be employed to build dependable computing machines (a certain contradiction notwithstanding), valuable expertise in design can be drawn from natural sciences. While biology provides countless examples of successfully implemented fault tolerance strategies, physics offers theoretical foundations, both of which were found to share common ground. It is perhaps a coincidence worth exploring in digital computing.

5. REFERENCES

[1] Aharonov, D., Ben-Or, M. Fault Tolerant Quantum Computation with Constant Error. *Proc. ACM 29th Ann. Symposium on Theory of Computing*, El Paso, Texas, May 1997, pp. 176-188.

[2] 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.

[3] Butts, M., DeHon, A., Golstein, S.C. Molecular Electronics: Devices, Systems and Tools for Gigagate, Gigabit Chips. *Proc. Intl. Conference on CAD (ICCAD'02)*, 2002, pp. 433-440.

[4] Canham, R., Tyrrell, A. An Embryonic Array with Improved Efficiency and Fault Tolerance. *Proc. IEEE NASA/DoD Conference on Evolvable Hardware*, Chicago Il, 2003, 275-282.

[5] Gaisler, J. Evaluation of a 32-Bit Microprocessor with Built-In Concurrent Error Detection. *Proc. 27th Annual Intl. Symposium on Fault-Tolerant Computing (FTCS-27)*, 1997, pp. 42-46.

[6] Goldstein, S.C. The Challenges and Opportunities of Nanoelectronics. *Proc. Government Microcircuit Applications and Critical Technology Conference (GOMAC Tech - 04)*, Monterey, CA, March 2004.

[7] Gottesman, D. Class of quantum error-correcting codes saturating the quantum Hamming bound. *Phys. Rev. A 54*, 1996, pp. 1862-1868.

[8] Johnson, B.W. *Design and Analysis of Fault-Tolerant Digital Systems*. Addison-Wesley, 1989.

[9] Laprie, J.-C. (Ed.). Dependability: Basic Concepts and Terminology. *Dependable Computing and Fault-Tolerant Systems Series*, Vol. 5, Springer-Verlag, Vienna, 1992.

[10] Liden, P., Dahlgren, P., Johansson, R., Karlsson, J. On Latching Probability of Particle Induced Transients in Combinational Networks. *Proc. Intl. Symposium on Fault-Tolerant Computing (FTCS-24)*, 1994, pp.340-349.

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

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

[13] Von Neumann, J. *The Computer and the Brain* (2nd edition). Physical Science, 2000.

[14] Von Neumann, J. The Theory of Self-Reproducing Automata. A. W. Burks, ed. University of Illinois Press, Urbana, IL, 1966.

[15] Von Neumann, J. 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.

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

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

[18] O'Connor, P.D.T. Practical Reliability Engineering. John Wiley & Sons, 4th edition, 2002.

[19] 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.

[20] Prodan, L. *Self-Repairing Memory Arrays Inspired by Biological Processes.* Ph.D. Thesis, “Politehnica” University of Timisoara, Romania, October 14, 2005.

[21] Prodan, L., Udrescu, M., Vladutiu, M. Survivability Analysis in Embryonics: A New Perspective. *Proc. IEEE NASA/DoD Conference on Evolvable Hardware*, Washington DC, 2005, 280-289.

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

[23] 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.

[24] 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.

[25] 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.

[26] Rimen, M., Ohlsson, J., Karlsson, J., Jenn, E., Arlat, J. Validation of fault tolerance by fault injection in VHDL simulation models. *Rapport LAAS No.92469*, December 1992.

[27] Rimen, M., Ohlsson, J., Karlsson, J., Jenn, E., Arlat, J. Design guidelines of a VHDL-based simulation tool for the validation of fault tolerance. *Rapport LAAS No93170, Esprit Basic Research Action No.6362*, May 1993.

[28] Shivakumar, P., Kistler, M., Keckler, S.W., Burger, D., Alvisi, L. Modelling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic. *Proc. Intl. Conference on Dependable Systems and Networks* (*DSN*), June 2002, pp. 389-398.

[29] Shor, P. *Fault-tolerant quantum computation*. arXiv.org:quant-ph/9605011, 1996.

[30] Shor, P. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. *Proc. 35th Symp. on Foundations of Computer Science*, 1994, pp.124-134.

[31] Sipper, M., Mange, D., Stauffer, A. Ontogenetic Hardware. *BioSystems*, 44, 3, 1997, 193-207.

[32] Sipper, M., Sanchez, E., Mange, D., Tomassini, M., Perez-Uribe, A., Stauffer, A. A Phylogenetic, Ontogenetic and Epigenetic View of Bio-Inspired Hardware Systems. *IEEE Transactions on Evolutionary Computation*, 1, 1, April 1997, 83-97.

[33] Steane, A. Multiple Particle Interference and Quantum Error Correction. *Proc. Roy. Soc. Lond. A 452*, 1996, pp. 2551.

[34] Udrescu, M. *Quantum Circuits Engineering: Efficient Simulation and Reconfigurable Quantum Hardware*. Ph.D. Thesis, “Politehnica” University of Timisoara, Romania, November 25, 2005.

[35] Udrescu, M., Prodan, L., Vladutiu, M. Simulated Fault Injection in Quantum Circuits with the Bubble Bit Technique**.** *Proc. International Conference "Adaptive and Natural Computing Algorithms"*, pp. 276-279.

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

[37] Udrescu, M., Prodan, L., Vladutiu, M. **I**mproving Quantum Circuit Dependability with Reconfigurable Quantum Gate Arrays**.** *2nd ACM International Conference on Computing Frontiers*, Ischia, Italy, 2005, pp. 133-144.

[38] 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.

[39] 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.

[40] Tempesti, G., Mange, D., Petraglio, E., Stauffer, A., Thoma Y. Developmental Processes in Silicon: An Engineering Perspective. *Proc. IEEE NASA/DoD Conference on Evolvable Hardware*, Chicago Il, 2003, 265-274.

[41] Viamontes, G., Markov, I., Hayes, J.P. High-performance QuIDD-based Simulation of Quantum Circuits. *Proc. Design Autom. and Test in Europe (DATE)*, Paris, France, 2004, pp. 1354-1359.

[42] Yu, Y., Johnson, B.W. A Perspective on the State of Research on Fault Injection Techniques. Technical Report UVA-CSCS-FIT-001, University of Virginia, May 20, 2002.

[43] ***. ITRS – International Technology Roadmap for Semiconductors, Emerging Research Devices, 2004, http://www.itrs.net/Common/2004Update/2004_05_ERD.pdf

[44] ***. Society of Reliability Engineers, http://www.sre.org/pubs/

[45] ***. http://www.dependability.org/wg10.4/