# Evaluating Dependability Options when Designing Future Systems-on-Chip

Dominique Tschopp
Ecole Polytechnique Fédérale de Lausanne (EPFL)
School of Computer and Communication Sciences
CH-1015 Lausanne, Switzerland
dominique.tschopp@epfl.ch

Maria Gabrani
IBM Zurich Research Laboratory
CH-8803 Rüschlikon, Switzerland
mga@zurich.ibm.com

Paolo Ienne
Ecole Polytechnique Fédérale de Lausanne (EPFL)
School of Computer and Communication Sciences
CH-1015 Lausanne, Switzerland
paolo.ienne@epfl.ch

## Abstract

*The evolution from micro- to nano-scale devices allows much more complex systems to be build on a chip. The nanometer (nm) devices, and the resulting systems, are less dependable than we traditionally expect them to be. We address the issue of reliability-aware design at the system level and the quest for optimal reliability design point with a methodology to evaluate the costs and benefits of the various designs. The proposed methodology focuses on the most critical attributes of a design, namely, reliability, performance and cost, and introduces five metrics to describe the effect of the design on the above attributes. The paper exercises the method and evaluation methodology on two well-known designs and a simulation model, and illustrates their effectiveness in the decision-making process.*

## 1 Introduction

Technological progress, both in silicon but also in emerging technologies such as carbon nanotubes, leads towards the design of electronic devices on the nanometer scale [1]. In these geometries chips are likely to host very complex systems consisting of a large number of heterogenous components. The resulting higher-density circuits and higher frequencies introduce increasing challenges for the operating conditions of the chip components. The increasing sensitivity to soft errors in combinatorial logic [2] and degraded signal quality defy the reliability, and thus the dependability of the systems-on-chip (SoC). Verification and testing can no longer accommodate these challenges. We need to design systems that reduce the demand for design correctness by providing means to find a good operating point dynamically.

To address these challenges we need to approach the design methods and tools from a new perspective: we need to move from worse-case deterministic methods to reliability-aware, stochastic approaches. Despite the decrease in geometries, the design of chips remains cost-sensitive, emphasizing the variety of dependability requirements for different applications. Thus, designers will be faced with numerous design options early in the design process. Tools and methodologies that will provide tradeoffs and assist designers into making informed choices and taking critical decisions, are expected to be imperative.

The issues addressed above are discussed in [3] and [4]. In the latter, a design framework is proposed but the paper mainly focuses on the problems and the issues that need to be addressed in providing solutions. Specific solutions that address reliability at different levels but for specific components, in particular for processor pipelines, are emerging. CRTR [5] introduces a redundantly threaded multiprocessor for chip multiprocessors that encompasses fault recovery by comparing the states of two processors that execute the same thread asymmetrically. DIVA [6] introduces a checker processor to verify the speculative results of a core processor dynamically. Razor [7] introduces a flip-flop, a delayed shadow latch and a metastability-tolerant comparator to detect and correct timing errors in processor pipeline stages dynamically. These are only few of the options that are being pursued. Each of them claiming to address specific operating conditions and providing diverse cost and benefits tradeoffs at different architectural levels and with various redundancy techniques.

In this paper we propose a method to model and evaluate different design options. The design method, which we call Modular Hierarchical Diagnosis on Chip (MoHiDoC) allows the modelling and integration of heterogeneous reliability solutions in a SoC. The design evaluation methodology is based on the notions of cost and benefit. We focus on the most important attributes of a computing system, namely reliability, performance and cost, and define metrics to measure the effect of the design on those attributes. We measure cost in the form of area and power, performance in the form of latency, and reliability in the form of mean time between failures.

The paper is structured as follows. In Section 2 we present our design method. In Section 3 we elaborate on the five metrics that constitute the major tools of our design methodology. The proposed design evaluation methodology is described in Section 4. The evaluation of these metrics and of the proposed design method is detailed in Section 5 using two well-known designs and a simulation model. We conclude this paper with a summary and directions for further study in Section 6.

## 2 Design Method

The MoHiDoC design method introduces a new way to model reliability-aware systems-on-chip. It models diverse error detection, analysis, and reaction mechanisms, also called reliability mechanisms, under a common structure. It proposes a standardized communication structure which allows the integration of those heterogeneous reliability mechanisms. In addition it enables a hierarchical structuring of error handling, which allows different layers of analysis, information, and control. We elaborate on those topics in the subsequent paragraphs.

### 2.1 Building Block Structures

In MoHiDoC, the logic circuitry is divided into Building Blocks (BB) in order to increase the granularity of error handling. BBs can be either basic, or enhanced.

Basic BBs originate from the Mealy State Machines with on-line error detection (e.g., [8]). While the Mealy State Machine describes a circuit, the goal of the BB is to describe a logic at different levels. Enhanced BBs, in addition to having the capability of detecting errors, can also analyze and react to errors. The functionality of an enhanced BB is shown in Figure 1 and described hereunder (the numbering corresponds to the numbering in the figure). When an error is detected an error indication (EI) is passed from the error detection to the error analysis (1). The analysis mechanism can determine whether the error can be taken care of locally or not. In the first case, the analysis passes the EI to the reaction mechanism (2). In the second case, the EI is passed

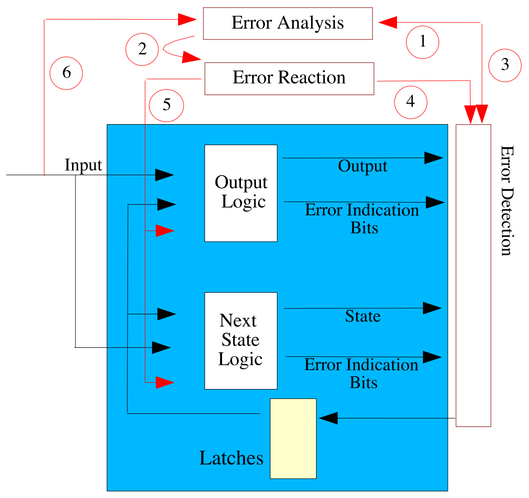

**Figure 1. Schematic diagram of an enhanced building block (BB).**

as an output of the BB to subsequent BBs on the path (3), which in turn will analyze the error. The reaction can be a direct recovery, in which case the corrected output is forwarded to the original destination as if no error occurred (4). Such a mechanism could for instance be an error correcting code (ECC). Alternatively, modifications in the output and next state logic are initiated to alleviate the error (5). For instance the operation could be repeated, under the condition that inputs were stored for repetition. Inputs to the analysis can also be used to analyze errors which occurred in previous BBs, since not all BBs have to be enhanced (6). It is important to note that one can combine basic BBs and enhanced BBs. Hence, we introduce already at this point a certain flexibility in the design and costs.

We assume that a fraction $\frac{1}{\delta}$ of all BBs are basic BBs, and a fraction $\frac{1}{\alpha} \leq \frac{1}{\delta}$ enhanced BBs, with $\frac{1}{\delta} + \frac{1}{\alpha} = 1$. We further assume that BB $i$ requires $A_{\delta_i} \geq 0$ additional area (relative to the original circuit) for error detection and $A_{\alpha_i} \geq 0$ additional area for analysis and reaction. Similarly, an additional power of $Po_{\delta_i} \geq 0$ is required for detection and $Po_{sb_i} \geq 0$ or $Po_{\alpha_i} \geq 0$ for analysis and reaction, depending on whether no error occurred (standby) or an error occurred (active).

### 2.2 Communication Structure

In MoHiDoC, the behavior of a system is modelled with Petri nets (PN). PNs are a well-known formal, graphical, executable technique for the specification and analysis of concurrent, discrete-event dynamic systems. In PNs, transitions correspond to actions leading to a state evolution when shot. The BBs defined above, as they themselves define the evolution of a system, are mapped to those transitions.

We consider a total number of transitions equal to $|T|$. Token places, which define the state of a system depending on their marking, are used as interfaces between BBs. In addition to offering a convenient way to model systems, PNs also give us a convenient way to structure the data and control information flowing from BB to BB. In advanced types of Petri nets, such as colored [9] or fuzzy [10], tokens contain some information about their own state (e.g degree of completion of a task). MoHiDoC, based on that paradigm, uses enhanced tokens, containing regular data and control information, to convey information between BBs.

The control information contained in tokens can be either analysis or reaction data. Analysis information is initialized when an error is detected and either cannot be analyzed locally or no reaction is possible. This information is propagated until a reaction is possible. Consecutively to the reaction, reaction information has to be propagated back to affected BBs. The way tokens are defined is beyond the scope of this paper. We refer the reader to [11] for detailed descriptions and examples of how analysis can be done.

We consider that the infrastructure that contains and communicates token information adds an area $A_{tok}$ to the chip area. This additional area overhead originates from the additional bandwidth required for instance by the control information, as well as from supplementary registers (e.g., for certain token places) or special buses.

### 2.3 Hierarchical Structure

The MoHiDoC framework supports hierarchical error handling. Hence, it offers several layers of analysis, information and control. The modelling of such a layered structure is straightforward with PNs as there exists mathematical formulation of PN refinements (e.g., [12]). In [11], an example with three layers is given. The lowest layer is on-chip and embedded in the original components. The second layer is part of a specialized component, still on chip, dedicated to error reaction at a lower granularity, that is it acts on a larger number of BBs. This layer can detect system problems (e.g., high error rates) and report them to the top layer. The top layer, implemented off-chip, e.g., on a service processor, can perform system-wide error reactions and influence the chip behavior (e.g., reduce frequencies in case the error rate is too high). Note that certain token places are interfaces between layers.

Considering the example above, we define the delay for a local reaction at BB $i$ as $d_i^{\mathrm{local}}$. The delay when an error occurs at the same BB and needs to be treated on the specialized component is $d_i^{\mathrm{sc}}$. Finally, if the error needs to be treated off-chip on the service processor, the incurred delay is $d_i^{\mathrm{sp}}$. Note that we consider one-way delays, i.e. from the moment the error was detected until a reaction was determined.

## 3 Design Metrics

We use the design method described in the previous section to model the reliability mechanisms and their communication in a SoC design, and from this model we derive the metrics needed to evaluate the design. We focus on the attributes of area, power, latency and reliability. The metrics that we define are cost of area ($C_{\mathrm{Area}}$), cost of power ($C_{\mathrm{Power}}$) and reliability benefit ($B_{\mathrm{Reliability}}$). In the case of latency, there are both costs and benefits. The reliability mechanisms increase the nominal latency of the system ($C_{\mathrm{Latency}}$, or delay), but at the same time the system reaction in case of error is faster ($B_{\mathrm{Latency}}$, or speedup). To evaluate the reliability options in a design we use the above metrics and define a cost vs. benefit function (CBF) $\Omega$ as

$$\Omega = \frac{\mathcal{C}(C_{\mathrm{Area}}, C_{\mathrm{Power}}, C_{\mathrm{Latency}})}{\mathcal{B}(B_{\mathrm{Latency}}, B_{\mathrm{Reliability}})}\,, \qquad (1)$$

where $\mathcal{C}(.)$ and $\mathcal{B}(.)$ denote cost and benefit functions, respectively. The $\Omega$ CBF is designed to favor local solutions, i.e., reactions close to where the errors are detected. Local solutions are advantageous as local reactions: reduce latency with respect to off-chip reactions or specialized component reactions, reduce the number of components involved (error containment), and increase the chances that a problem specific solution is found.

Consequently, we state our objective as $Pr(\mathrm{Sol}) \rightarrow 1$. In our CBF, the probability of a local solution depends on two elements. First, the error should be detected. We relate the probability of detecting an error to the fraction of BBs having error detection capabilities and set $Pr(\mathrm{detect}) = \frac{1}{\delta}$.

Second, we introduce the concept of effective distance between detection and reaction $d_{\mathrm{eff}}$. We define the probability of having an optimal effective distance as the probability of having an effective distance of 0 (local solution), when the average effective distance is $d$. Hence, we express $Pr(d_{\mathrm{opt}}) = e^{-d_{\mathrm{eff}}}$. With the Poisson model we describe the probability of having a local solution ($d_{\mathrm{eff}} = 0$) under the assumption that the occurrence of an error in independent on the occurrence of another error in the same region. If we define by $\Gamma$ the spatial coupling between errors then $Pr(d_{\mathrm{opt}}) = (1+\Gamma d_{\mathrm{eff}})^{-\frac{1}{\Gamma}}$, which is equivalent to the Poisson distribution for $\Gamma = 0$ [13]. Here we restrict ourselves to the Poisson model for the sake of simplicity. We then get

$$Pr(\mathrm{Sol}) = \mathrm{Pr(d_{opt}\ and\ detect)} = \frac{1}{\delta}\mathrm{e}^{-\mathrm{d_{eff}}} \qquad (2)$$

where the effective distance is the ponderated average of standard delays for reactions at the different layers. In the sequel we elaborate on the design metrics and how they contribute to $\Omega$.

### 3.1 Area

For area, the cost can be directly derived from the BB model. We simply sum up the additional required area for every BB and average it by the number of BBs ($|T|$). One can write

$$C_{\text{Area}} = \frac{1}{|T|}\left[\sum_{i=1}^{|T|}(A_{\delta_i} + A_{\alpha_i} + A_{tok})\right] \leq \frac{1}{\delta}A_\delta + \frac{1}{\alpha}(A_\alpha + A_\delta) + A_{tok} \qquad (3)$$

In (3), for the sake of simplicity, we use the terminology $A_j = \max(A_{j_i})$, with $j \in \{\delta, \alpha\}$.

### 3.2 Power

Similarly for power, we consider a cost reckoned as the sum of additional power consumptions per BB averaged over all BBs. Note that in the case of power, we consider that error detection is always active and that analysis and reaction are only activated when an error is detected. We consider that at most $N_{\max}$ errors occur simultaneously with a probability $Pr^N = Pr(N_{\max})$ leading to a maximum power overhead of:

$$C_{\text{Power}} = \frac{1}{|T|}\left[Pr^N Po_\alpha + \sum_{i=1}^{|T|-N_{\max}} Po_{sb_i} + \sum_{i=1}^{|T|}(Po_{\delta_i})\right] \leq \frac{1}{\delta}Po_\delta + \frac{1}{\alpha}\left[Po_\delta + (1 - \frac{N_{\max}}{|T|})Po_{sb} + \frac{N_{max}}{|T|}Po_\alpha Pr^N\right] \qquad (4)$$

Similarly, $Po_k = \max(Po_{k_i})$, with $k \in \{\delta, sb\}$.

### 3.3 Latency

The MoHiDoC framework brings both costs and benefits in terms of latency. The latency becomes a statistical variable with its mean corresponding to the average delay and its variance to the response time in case or error.

We define the cost in terms of latency as the ratio between this processing time with and without MoHiDoC, i.e:

$$C_{\text{Latency}} = \frac{t_{\text{MoHiDoC}} - t_{\text{normal}}}{t_{\text{normal}}} \qquad (5)$$

On the other hand, the framework will considerably reduce the reaction latency in case an error should occur. As explained above, our objective is to make the reaction as local as possible. Thus, we define the benefit in terms of latency as the ratio between the time necessary for a service processor reaction $t^{\text{sp}}_{\max}$ and the time necessary for a local reaction $t^{\text{local}}_{\max}$, multiplied by the probability that the reaction is local $Pr(\text{Sol})$ (Equation (2)):

$$B_{\text{Latency}} = Pr(\text{Sol})\frac{t^{\text{sp}}_{\max}}{t^{\text{loc}}_{\max}} = d_2 Pr(\text{Sol})\,. \qquad (6)$$

### 3.4 Reliability

The framework presented above is designed to tackle soft errors in the combinatorial logic. A well known parameter to characterize the reliability of systems is Mean Time Between Failure (MTBF). In MoHiDoC, the MTBF is linked to soft errors as explained below.

According to [14], if we have a constant hazard (failure rate) $\lambda$, we can approximate the $MTBF$ as $MTBF = \frac{1}{\lambda}$. We are interested in soft errors in the combinatorial logic, and consequently the hazard can be set to a constant value equivalent to the soft error rate (SER). Hence, we have $\lambda = SER$. We refer to [15] and [2] for the estimation of this error rate. The SER (affecting the system) is defined as the product of the real SER (all soft errors, including the ones which are not latched up), multiplied by the time derating (TD) and logic derating (LD). This means that we consider that a soft error occurred in the combinatorial logic when a transient error in the result of a logic circuit is subsequently stored in memory elements (latches) and affects valid data. One can set $SER = SER_{\text{real}} \times TD \times LD$, where $TD$ is the fraction of time the circuit is sensitive to SEU, and $LD$ is the probability to corrupt valid data. Then, one can write

$$MTBF = \frac{1}{SER_{\text{real}} \times TD \times LD}\,. \qquad (7)$$

We define the benefit in terms of reliability as the ratio between the MTBF with MoHiDoC and without MoHiDoC. Hence, we obtain

$$B_{\text{reliability}} = \frac{LD_{\text{old}}}{LD_{\text{MoHiDoC}}} = \frac{LD_{\text{old}}}{1 - Pr(\text{Sol})}\,. \qquad (8)$$

## 4 Design Evaluation

In this section we evaluate the $\Omega$ CBF in order to reveal some inherent tradeoffs of error handling frameworks. Even though we expect that the functions $\mathcal{C}(.)$ and $\mathcal{B}(.)$ of Equation (1) are complex, in this paper we define them as sums. To that end, we consider the different costs and benefits as percentages (e.g., $C_{\text{Area}}$ is the additional percentage of area required). Hence, we are consistent in terms of units. Rewriting the aforementioned equation, we obtain:

$$\Omega = \frac{C_{\text{Area}} + C_{\text{Power}} + C_{\text{Latency}}}{B_{\text{Latency}} + B_{\text{Reliability}}} \qquad (9)$$

where $C_{\text{Area}}$, $C_{\text{Power}}$, $C_{\text{Latency}}$, $B_{\text{Latency}}$, and $B_{R\text{eliability}}$ are defined in Equations (3), (4), (5), (6), and (8), respectively.

We plot $\Omega$ as a function of various parameters. Table 1 lists those different variables. We vary one of them while the others have the value listed in Table 2. Note that in

**Table 1. Variables for the evaluation of $\Omega$.**

| *Variable* | *Range* | *Unit* |
|---|---|---|
| $\alpha$ | [1 : 100] | Fraction of transitions (BBs) with analysis |
| $\lvert T\rvert$ | [1 : 100] | Number of transitions (BBs) |
| $C_{\text{Area}}$ | [0 : 1] | Additional %-tage of area |
| $C_{\text{Power}}$ | [0 : 1] | Additional %-tage of consumed power |
| $LD_{\text{old}}$ | [0 : 0.8] | Logic Derating |
| $B_{\text{Latency}}$ | [0 : 30] | Relative gain in terms of latency |

**Table 2. Parameters for the evaluation of $\Omega$.**

| *Parameter* | *Value* | *Equation* |
|---|---|---|
| $\alpha$ | 4 | 3,4,2 |
| $\delta$ | 1.2 | 3,4,2 |
| $A_\alpha$ | 0.01 | 3,4,2 |
| $A_\delta$ | 0.03 | 3,4,2 |
| $A_{tok}$ | 0.01 | 3 |
| $Po_\alpha$ | 0.01 | 4 |
| $Po_{\text{sb}}$ | 0.01 | 4 |
| $Po_\delta$ | 0.02 | 4 |
| $\lvert T\rvert$ | 100 | 3,4,2 |
| $N_{\max}$ | $\left\lceil \frac{\lvert T\rvert}{50} \right\rceil$ | 4 |
| $Pr(N_{\max})$ | 0.05 | 4 |
| $C_{\text{Latency}}$ | 0.026 | 5 |
| $d^{\text{local}}$ | 0.01 | 2 |
| $d^{\text{sc}}$ | 0.1 | 2 |
| $d^{\text{sp}}$ | 3 | 2 |
| $LD_{\text{old}}$ | 0.8 | 8 |

this table we list the parameters defined in the above equations. Figure 2 shows the resulting graphs. As one can see, $\Omega$ is higher when all the transitions have local analysis capabilities (low $\alpha$), and when the number of transitions is high (high $|T|$). The interpretation for that result is that the cost of having many local analysis and reaction capabilities is slightly more important than the related benefit in our model. The second result, i.e. that the number $|T|$ should be low, tends to confirm that result as it indicates that the cost of granularity is high. Note however that both curves converge toward a constant value and that above $\alpha = 10$ and $|T| = 10$, $\Omega$ remains almost constant. Further, it is interesting to point out the fact that $\Omega$ varies very little when the $C_{\text{Area}}$ and $C_{\text{Power}}$ remain below 10% of the total area and power respectively. This means that implementing the framework is advantageous as long as we remain below this threshold. It is also important to notice that $\Omega$ decreases very rapidly as the $B_{\text{Latency}}$ increases. This means that if the delay to an outside processor is high, the benefit of MoHiDoc rapidly becomes important. Finally, and maybe most importantly, one can note that $\Omega$ starts decreasing dramatically when the LD is higher than 10% (and consequently the SER increases). This result shows the usefulness of MoHiDoC in the future, when this rate will considerably increase due to the factors listed in the introduction.

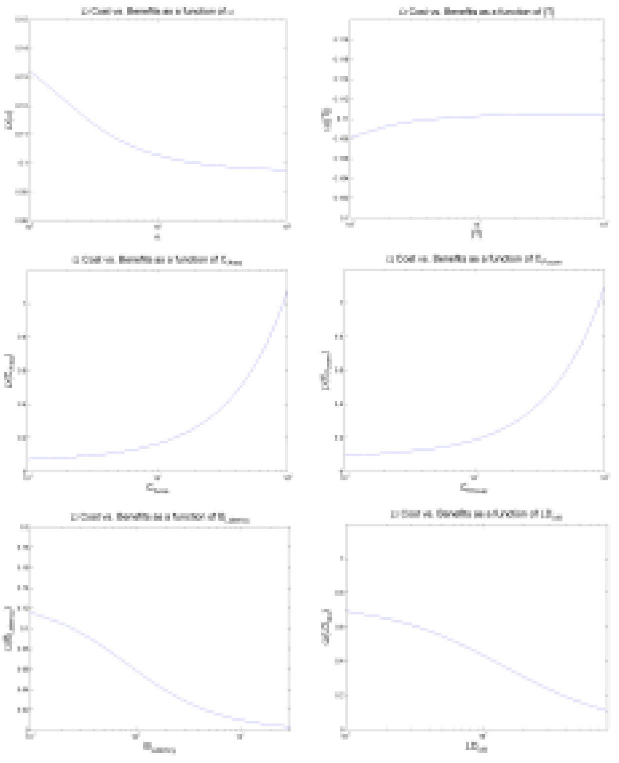

**Figure 2. Cost vs. benefits graphs.**

The paragraph above revealed some of the inherent tradeoffs of our cost versus benefits model. The rules of thumb one can extrapolate from this evaluation are the following: The cost of having high granularity for analysis decreases rapidly and becomes almost constant for less than $\frac{1}{10}$ of transitions with such capabilities. However, the variation in $\Omega$ is small. Similarly, a very low granularity in terms of number of transitions leads to a slightly lower $\Omega$. If the additional power consumption and area costs remain below 10%, $\Omega$ does not increase significantly. This means that the costs outpace the benefits only above that threshold. The higher the SER, the higher the benefit of implementing the cost versus benefits model, even if the LD is as small as 1%.

## 5 Method Evaluation

In [11] we show how the MoHiDoC framework can be used to model known component specific reliability mechanisms, namely DIVA [6] and Razor [7]. It is useful to relate our theoretical approach to previous existing approaches which were tested and implemented. This process showed that similar approaches can be implemented in practice at a reasonable cost. Moreover, MoHiDoC provides means to further expand those mechanisms and to link together different type of mechanisms. Indeed, both DIVA and Razor fit nicely to the BB model. They are are actually a subset of our BB model. We next describe a simulation model and then we apply our CBF model to those existing mechanisms and the simulation model.

## 5.1 Simulation

We developed a probabilistic simulation model of an InfiniBand (IB) Host Channel Adapter (HCA) [16]. The HCA architecture can be schematically subdivided into five main components: the IB input and output components (IBIn and IBOut), the receive and sent processors (RP and SP), and the Control Unit (CU). We introduced a number of parameters that model the behavior of the system, such as the cache hit ratio for the chip caches and the sizes of the receive and transmit packets and messages. For more detailed description of the simulation model please refer to [11].

We implemented a particular scenario on top of our IB HCA simulation model. An address translation (AT) is requested by the the RP from the CU. When the CU returns the requested address to the RP, a particle strikes and corrupts the data. Practically, the data is randomly marked as being correct or corrupted ($nb$ bits, with a higher probability for lower number of bits). The RP treats the received address as if it were correct all the time and stores the data from the IBin buffer at this location in memory. If data marked as corrupted is stored, we consider that the data integrity was challenged.

We also implemented simple detection and analysis capabilities, as well as a specialized component on-chip. The local reaction mechanisms recovers up to $nb = 1$ corrupted bits, or else passes the error to the specialized component. The specialized component reaction is a repetition of the address translation. In Figure 3 we illustrate the scenario and show the BB and PN representation. Note that we consider a delay of 1 cycle for on-chip reaction in the logic (local reaction) and 10 cycles for specialized component reaction. Unfortunately, we did not find any benchmark values for those delays. Our estimation is that local reaction can be done in the same clock cycle and specialized component reaction should add less than 10% of delay.

We first run the simulation for 1,000,000 cycles without having the framework enabled in order to evaluate the MTBF for a soft error rate of 0.002 errors/cycle (test 1). Then we run the simulation again for 1,000,000 cycles and the same SER, but this time with the framework enabled, and estimated the cost in terms of latency for a long run (test 2). Note that this cost only corresponds to a particular

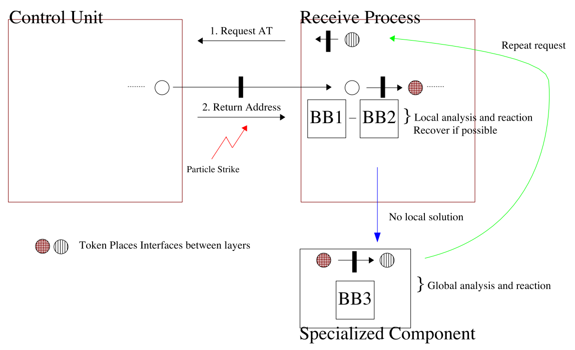

**Figure 3. Illustration of the simulated soft error scenario.**

**Table 3. Results of the two test-runs.**

| *Measures* | *Test 1* | *Test 2* |
|---|---|---|
| Average time (cycles) to receive a packet (time in RP) | 108 | 103 |
| Number of packets treated (100% of packets received) | 8389 | 8390 |
| Average size (bytes) | 1321 | 1321 |
| Average time (cycles) to transmit a packet (until in IBout buffer) | 100 | 100 |
| Number of packets transmitted | 9942 | 9956 |
| Average size (bytes) | 767 | 769 |
| Mean Time (cycles) Between Failure | 68729 | $\infty$ |
| Percentage of packets affected | 6.9% | 0 |
| Total number of errors injected | 1978 | 2019 |
| Errors corrected in Specialized Component (more than 1 bit) | NA | 24 |

scenario, and could be different in other parts of the chip. The results are exposed in Table 3. One can notice that the average delay per packet in the receive side processing is identical over a long run whether the framework is running or not. We explain it by the fact that the processing time per packet is variable, determined by the random model of our simulation, and that only a small number of packets are affected by errors, resulting in a negligible overhead over a long run. On the other hand, if we consider the delay per corrupted packet, the average delay is approximately 2.8 cycles, which represents $\frac{2.8}{108} = 2.6\%$. Below we consider this particular value. Table 4 exposes the values which we obtained from our simulation model and we use in the following section.

## 5.2 Correlation to the Mathematical Model

Our simulation model obviously did not allow us to estimate all the costs involved in a complete “costs versus benefits” estimation. On the other hand, we obtained some pa-

**Table 4. Costs obtained from the simulation model.**

| *Metric* | *Value* |
|---|---|
| $C_{\text{Latency}}$ | 2.6% |
| $MTBF_{\text{old}}$ | 68729 cycles |

**Table 5. Costs obtained from the literature.**

| *Approach* | *Parameter* | *Value* | *Ref.* |
|---|---|---|---|
| DIVA | $C_{\text{Area}}$ | 6% | [17] |
| | $C_{\text{Power}}$ | 1.5% | [17] |
| | $C_{\text{Latency}}$ | 3% | [6] |
| Razor | $C_{\text{Power}}$ | 3.1% (error free) | [7] |
| | $C_{\text{Power}}$ | 4.1% (error) | [7] |
| | $C_{\text{Latency}}$ | ≈0[1] | [7] |

rameters from the literature, i.e from the DIVA and Razor papers. In Table 5, we list the parameters we extracted from various publications. Unfortunately, not all parameters were available.

We estimate the $C_{\text{Area}}$ for Razor as being between 1 and 10%. Hence we obtain costs for Razor and DIVA of

$$Cost_{\text{DIVA}} \approx 0.105 \quad (10)$$
$$Cost_{\text{Razor}} \approx 0.063 : 0.163\ (error\ case)\,. \quad (11)$$

For both approaches, we set $B_{\text{Latency}} = 0$, since their implementation does not add any benefit in terms of latency. Again, the “reliability” benefit is tricky to estimate. For the sake of comparison, we set the $SER_{\text{MoHiDoC}}$ in Equation (8) to one, since it is impossible to quantify for Razor and DIVA, and use $SER_{\text{old}} = SER_{\text{real}} \times LD \times TD$. As before, we define $TD = 0.5$ and arbitrarily set $LD = 0.5$. For $SER_{\text{real}}$, we use the values presented in [2] for SER/Chip in the logic and obtained through simulation. In Figure 4, we present the evolution from 1992 until 2011 (DIVA and Razor almost identical). One can see that $\Omega$ decreases by nine orders of magnitude in less than 20 years. This result infers that whatever approach is taken, as expected, the importance of dealing with SEU will become increasingly high.

If we assume similar costs for MoHiDoC in terms of area, power and latency, we immediately see that in order to have better performances than the two approaches above, MoHiDoC should either considerably reduce the logic derating and/or offer a great benefit in terms of latency. The simulation has shown us that $C_{\text{Latency}}$ is very similar to that

[1]We assume that this value is averaged over a long run and that similarly to what happens in our model, the overhead costs are negligible in that case. In case an error occurs, pipeline operations are delayed one cycle, resulting in a relative cost of $\frac{7}{6} \approx 1.2\%$

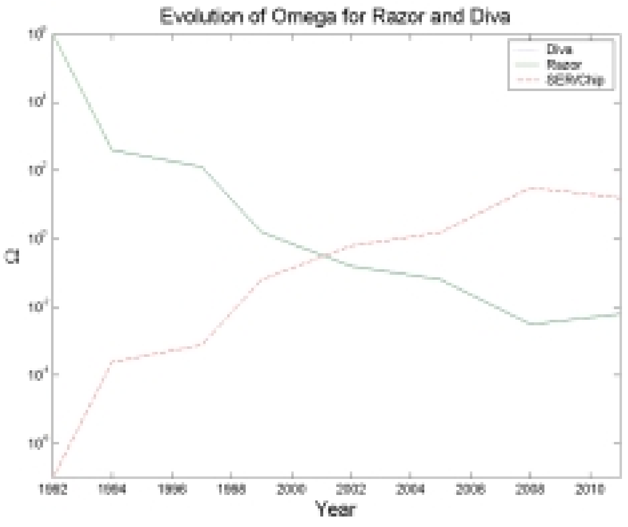

**Figure 4. Evolution of $\Omega$ for Razor and DIVA.**

of DIVA and Razor.

While it is difficult to gauge the other terms without implementing an actual MoHiDoC framework on a real chip, we feel that those objectives should be achievable for at least two reasons. First, MoHiDoC allows the integration of the above mentioned framework at no or very small cost while in addition offering several levels of control and miscellaneous reaction mechanisms, thus increasing the overall reliability. Second, MoHiDoC allows error propagation, hence it is possible to save hardware for analysis and reaction in less critical parts of the chip.

## 6 Conclusions and Future Work

To address the emerging quest for dependability options in future SoC designs, we propose a novel design method and a cost vs. benefit evaluation methodology. The design method, called MoHiDoC, structures the various reliability mechanisms in a SoC under a common format and models their communication and hierarchical control. From the resulting design, we derive five metrics that reflect the effect of the selected dependability options on the cost, performance and reliability attributes of the Soc. We design a cost vs. benefit function (CBF) that utilizes these metrics to provide tradeoffs between the candidate dependability options.

We believe that the proposed MoHiDoC design method offers many benefits.The method leads to a standardized interface and mechanism for reporting problems and referring them to a higher-level management authority, which has always been very difficult. The actual reliability mechanisms can be defined for different levels (system, microarchitecture, circuit, etc.), which allows flexibility but is also future-proof, as new mechanisms can be adopted. Moreover, the method offers also flexibility in the cost, as it is up to the system designer to decide where the different reliability mechanisms will be placed and how they will be linked together. Currently, chip verification, code debugging during system bring-up and field problem analysis are extremely time-consuming. The method provides an architected framework that may reduce this time significantly. Most importantly, it helps in optimizing the design point regarding performance, power, chip size, and costs.

The application of MoHiDoC and CBF to two well-known designs and to a simulation model illustrates the suitability of both to model and provide dependability design tradeoffs. Several messages are derived that point to rule-of-thumb uses and trends.

This work is in its early stages, and clearly much further study needs to be done. The design method and the evaluation methodology need to be refined. More specifically, the evaluation function, as presented in this paper, sums the various cost and benefit metrics. This is a simplification. We need to study the dependencies between the various metrics, and define more accurate models for the cost and benefit functions. More work is also required regarding the various metrics. In particular, we need to derive analytical formulas for the latency metrics. In addition, we also need to consider the case of false alarms. We also would like to add proactive mechanisms, i.e., one could work on ways to reduce soft error rates before they become too high.

Finally, the method and evaluation methodology presented focus on optimizing the SoC's dependability. In future nanoelectronic designs, power and performance will also become stringent design-optimization parameters. Therefore, we aim to incorporate power and performance optimization mechanisms into MoHiDoC and into the CBF(s). We believe that such an integration will allow us to reduce costs and to link features such as temporal frequency alteration.

## References

[1] *InfiniBand$^{\mathrm{TM}}$ Architecture Specification Release 1.1*, 2002.

[2] *International Technology Roadmap on Semiconductors*, 2003.

[3] T. M. Austin. Diva: A reliable substrate for deep submicron microarchitecture design. In *Proc. 32nd ACM/IEEE Int'l Symp. Microarchitecture (MICRO-32)*, pp. 196-207, 1999.

[4] D. Das and N. A. Touba. Weight-based codes and their application to concurrent error detection of multilevel circuits. In *Proc. 17TH IEEE VLSI Test Symp.*, p. 370, 1999.

[5] D. Ernst et al. Razor: A low-power pipeline based on circuit-level timing speculation. In *Proc. 36th Int'l Symp. Microarchitecture (MICRO-36)*, 2003: http://www.microarch.org/micro36/html/pdf/ernst-Razor.pdf

[6] A. V. Ferris-Prabhu. On the assumptions contained in semiconductor yield model. *IEEE Trans. Computer-Aided Design*, 11(8), 1992.

[7] A. Herkersdorf and W. Rosenstiel. Towards a framework and a design methodology for autonomic integrated systems. In *Proc. GI Workshop on Organic Computing*, Sept. 2004.

[8] H. H. Ammar and L. Yu. Fuzzy marking Petri nets: Concepts and definition. In *Proc. 1995 IEEE Int'l Symp. Intelligent Control*, pp. 291-297, 1995.

[9] K. Jensen. A brief introduction to coloured Petri nets. In *Proc. Third Int'l Workshop on Tools and Algorithms for Construction and Analysis of Systems*, pp. 203-208. Springer-Verlag, 1997.

[10] T. V. M. Gomma, C. Scarbrough and I. Pomeranz. Transient-fault recovery for chip multiporcessors. In *Proc. 30th Annual Int'l Symp. Computer Architecture (ISCA'03)*, June 2003.

[11] G. D. Micheli. Robust system design with uncertain information. In *Proc. First ACM/IEEE Int'l Conf. Formal Methods and Models for Co-Design (MEMOCODE'03)*.

[12] H. Nguyen. A systematic approach to ser estimation and solutions. In *Proc. Second Workshop on Evaluating and Architecting System dependability (EASY)*, 2002.

[13] P. Shivakumar et al. Modeling the effect of technology trends on the soft error rate of combinational logic. In *Proc. Int'l Conf. Dependable Systems and Networks (DSN'02)*, 2002.

[14] M. L. Shooman. *Probabilistic Reliability: An Engineering Approach*. McGraw-Hill, 1968.

[15] D. Tschopp. MoHiDoc: Modular hierarchical diagnosis-on-chip. Master's thesis, Swiss Federal Institute of Technology, Lausanne, and IBM Zurich Research Laboratory, 2004.

[16] C. Weaver and T. M. Austin. A fault tolerant approach to microprocessor design. In *Proc. 2001 Int'l Conf. Dependable Systems and Networks (formerly: FTCS)*, pp. 411-420. IEEE, 2001.

[17] W. M. Zuberek and I. Bluemke. Hierarchies of place/transition refinements in Petri nets. In *Proc. 5th IEEE Int'l Conf. Emerging Technologies and Factory Automation*, pp. 355-360, 1996.