Team 3 / Robust Integrated Systems (RIS)

Themes
Robust massively parallel single-chip architectures  
Power management from the OS down to silicon  
Fault tolerant and self-adaptative architectures  
3D NOC Robust Architectures  
Design in Reliability face to aging, process variation and soft errors  
Evaluation of robustness and qualification : radiation testings, fault injection  
Architectures for Nanotechnologies

Expertise

Fields of expertise
Design for Reliability, Design for Test, Self-Repair, Fault-tolerance, Design for Soft-Error Mitigation: Methodologies, Tools and Architectures

Know-how
Multilevel platforms for fault simulation and robustness automatic insertion at several abstraction levels; 3D integration solutions  
test platform for radiation faults measurement ; SEE error-rate prediction of circuits and systems

Research keywords
Fault tolerance, multi-core systems robustness, 3D circuits, aging, fault-injection

Contact

Michael NICOLAILDIS  
CNRS Research Director - TIMA  
(+33) 476 57 46 96  
michael.nicolaidis@imag.fr

Raoul VELAZCO  
CNRS Research Director – TIMA  
(+33) 476 57 46 89  
raoul.velazco@imag.fr
The Cells Framework: Overall Description

Members: Michael Nicolaidis, Nacer-Eddine Zergainoh, Thierry Bonnoit, Amir Charif, Fraidy Bouesse, Michael Dimopoulos, Panagiota Papavramidou

Keywords: Ultimate CMOS and post-CMOS technologies, high defect densities, reliability, yield, low-power, massively parallel single-chip tera-device computers,

Cooperation: STmicroelectronics, iRoC, Atmel

Contracts: ELESIS, RESIST

Ultimate-CMOS and post-CMOS technologies promise integrating trillions devices in a single die, leading to single-chip massively parallel architectures comprising thousands interconnected processors, and enabling the next computation turn. But the aggressive technology scaling that paves the way to the ultimate CMOS nodes has dramatic impact to: process, voltage and temperature (PVT) variations; sensitivity to electromagnetic interferences (EMI), to atmospheric radiation (neutrons and protons) and to alpha particles; and circuit aging. It also imposes stringent power dissipation constraints. The resulting high defect levels, heterogeneous behavior of identical circuits, accelerated circuit degradation over time, and extreme complexity, affect adversely fabrication yield and/or prevent fabricating reliable chips in ultimate CMOS and post-CMOS technologies. These issues are becoming the main show-stoppers in the path leading to these technologies. The Cells framework addresses the severe issues related to the design of massively parallel tera-device processors affected by high defect densities, in which we severe issues have to be addressed such us:

- After fabrication, all processing and routing nodes may be affected by some temporary faults such as delay faults, or clock skews.
- Fabrication faults altering persistently the circuit behavior may massively affect one or more regular blocks (RAMs, FIFOs, buses) in a large fraction of nodes. Such faults may also very frequently occur during product life.
- Fabrication faults altering persistently the behavior of irregular blocks (thus difficult to repair) may affect a significant portion of nodes. Such faults may also frequently occur during circuit life (e.g. every few days), and thus during application execution.
- New timing faults induced by circuit aging, as well as soft errors (SEUs and transients) may frequently occur during circuit life (and thus during application execution).
- Circuit degradation is continuous and requires continuous self-regulation of circuit parameters (clock-frequency, voltage levels, body bias), to maintain operational each processor node.

Clearly, no existing solution can cope with such massively defective systems, which invalidate even massive redundancy schemes (e.g. duplication, TMR), as all replicated parts may be defective. Such schemes also induce high area and power penalties.

Some approaches targeting the design of reliable single-chip massively parallel processors avoids massive redundancy by using self-tests (hardware implemented or software implemented to detect failures and create routing tables that are used subsequently to avoid failed processing nodes or failed routes. However, such approaches could not cope with the issues affecting ultimate CMOS and post CMOS technologies as:

- In highly defective technologies, the vast majority of nodes (processing elements and routers) may include one or another kind of faults (e.g. timing faults produced by process, voltage and temperature variations, EMI, or aging). Thus, declaring defective the nodes affected by any kind of faults will quickly waste the computational resources of the chip.
- Achieving high fault coverage for timing faults is very difficult. Thus, many of these faults may escape fabrication test and also periodic self-tests and produce run-time errors.
- Faults occurring during application execution can not be covered by self-tests.

In this project we develop a comprehensive approach enabling using in efficient and reliable manner all parts able to perform useful computations.

The Cells framework (On-Chip Self-Healing Tera-Device Processors) [1-2] comprises several techniques spanning at all levels of the system: circuit, processor/architectural, array/routing, task-scheduling/allocation. Innovations are introduced at all levels of this framework, including its overall architecture, its particular components, and the way the cooperation of these components is architected to optimize the outcome. Developments concerning some of these components are presented in the next sections.

References
Publications on the Overall Cells Framework
Recent Publications on Cells’ Components


Early Publications on Cells’ Components


Circuit-level fault-tolerance in the Cells Framework

Members: Michael Nicolaidis, Thierry Bonnoit, Fraidy Bouesse, Michael Dimopoulos, Panagiota Papavramidou

Keywords: PVT variability; aging; timing faults; soft errors; design for reliability, yield, and low-power; double-sampling schemes.

Cooperation: iRoC, Atmel, EADS

Contracts: ELESIS, RESIST

In massively parallel processor chips, a possible (and rather straightforward) approach may consist in exploiting the existence of large numbers of processing cores to implement fault tolerance, by using two processor cores to duplicated the execution of each task and comparing their outcome to detect errors (double modular redundancy-DMR), or by using three processor cores to triplicate the execution of each task and voting their outcome to correct errors (triple modular redundancy-TMR). These approaches have two major flaws: drastic decrease of processing power, and drastic increase of energy dissipation per task.

Similarly, software implemented fault-tolerance drastically impacts performance and power, as it replicates the execution of each task. Hence, a breakthrough in fault-tolerant design is required for reducing drastically performance and energy dissipation penalties, and improve reliability.

Such a breakthrough was achieved by a scheme (referred hereafter as double-sampling) we proposed in reference [1] and evaluated in [2]. This scheme reduces drastically hardware and power costs, by avoiding both hardware replication and operations replication. Instead, to check the correctness of an output signal of a logic block; this scheme observes this signal at two different instants, by adding a redundant latch and driving it by means of a delayed clock signal (Figure 1a). This scheme was later extended in the RAZOR architecture [3] to also perform error correction: upon error detection RAZOR used the contents of the redundant latch to replace the contents of the functional flip-flop (Figure 1b). Reference [3] demonstrated that this scheme can achieve drastic power reduction, by reducing the supply voltage to very low levels and using RAZOR to detect and correct the timing errors induced by this reduction.

While the double-sampling scheme reduces drastically the area and power cost with respect to traditional fault-tolerant schemes, some area and power penalties are still introduced due to the redundant latches. Furthermore it suffers from two drawbacks:

- To avoid false alarms and miss-corrections, the path delays of the combinational circuits signals checked by the schemes in figure 1, must exceed the delay \( \delta \) of the delayed clock. To ensure this constraint, buffers should be needed in the paths with delays shorter than \( \delta \), inducing no negligible area and power cost.
- Also, due to this issue, to avoid significant area and power penalties, we have to use moderate values for \( \delta \), limiting the duration of detectable faults.

**GRAAL Architecture**

One goal of Cells is to explore new fault tolerant architectures that are exempt of the above problems. These problems are due to the fact that all the stages of a FF-based design compute at the same time. This leaves short stability time to the combinational circuit outputs that we could exploit for error checking. In flip-flops, when the master latch is transparent, the slave is on memorization and vice versa. Thus, if we transform a flip-flop-based design (Figure 2a), into its equivalent latch-based design, by moving the slave latches of the FFs to the middle of the combinational circuits, (as shown in Figure 2b where \( Ck \) and \( Ckb \) are replaced by non-overlapping clocks \( \Phi 1 \) and \( \Phi 2 \)), we obtain a design that can work at the same clock frequency as the original one. In addition, the master...
and slave latches operate at different phases. Then, the outputs of any combinational block (inputs of latches), are stable during the period in which its adjacent blocks are in computation. Thus, we can compare the outputs of the latches against their inputs to detect timing and transient faults of large duration, according to the GRAAL architecture (Figure 3) that we have introduced few years ago.

Figure 2: FF-based design (a), and latch-based design (b)

This scheme also detects SEUs as well as all kinds of latch faults (transition faults, retention faults). Thorough analysis shows that GRAAL detects delay faults with duration up to 100% of the circuit delays without imposing short path constraints. Thus, it offers comfortable timing margins that can be employed for detecting large delay faults, and at the same time reduce power dissipation by using the idea introduced by the University of Illinois (reduce Vdd and detect and correct induced timing errors). Note that, fault multiplicity is not an issue for GRAAL. Faults of any multiplicity are detected as far as their duration does not exceed this value. GRAAL also tolerates clock skews of up to 25% of the clock period, as an inherent property of latch-based designs. In addition, we show that larger clock skews that are not tolerated are always detected (thus they are recoverable by using instruction replay).

Figure 3: The GRAAL error detection architecture

Our GRAAL architecture can be implemented in two ways. The one version uses redundant latches, which introduce non-negligible area and power penalties. Recently this first version of our GRAAL architecture was used by others in the so-called Bubble Razor implementation. The second version of GRAAL architecture is much more efficient as it does not use of redundant latches. Only a two-input XOR gate is added to each latch to compare its inputs against its outputs. Thus, this version of GRAAL induces drastically lower area and power penalties with respect to any known error detection scheme.

Table 1 shows the evaluation results of the GRAAL architecture. These results are based on the implementation of GRAAL in the icyflex processor from CSEM [4][5], in ST Microelectronics 40nm technology. In order for the comparisons to be consistent, all implementations of icyflex, shown in table 1, were synthesized for the same clock frequency. Thus, the 150MHz clock frequency suggested by CSEM was selected for all these implementations.

Latch-based designs can be synthesized with or without time borrowing, and the synthesis was done by means of the Design Compiler [6], which supports latch-based design for both cases. Latch-based designs can be synthesized with or without time borrowing, and the synthesis was done by means of the Design Compiler [6], which supports latch-based design for both cases. The maximum duration of detectable faults is achieved by GRAAL when no time borrowing is used. Thus, no-time borrowing has been first selected for achieving the highest possible reliability.

Table 1: GRAAL evaluation

<table>
<thead>
<tr>
<th>icyflex2 Implementations</th>
<th>Reference Latch no-time Borrowing</th>
<th>Latch GRAAL_A no-time Borrowing</th>
<th>Flip-Flop</th>
<th>Latch Time Borrowing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell Library</td>
<td>1.10V</td>
<td>1.10V</td>
<td>0.90V</td>
<td>0.90V</td>
</tr>
<tr>
<td>Area</td>
<td>100%</td>
<td>103.60%</td>
<td>92.72%</td>
<td>100.98%</td>
</tr>
<tr>
<td>Power</td>
<td>100%</td>
<td>103.95%</td>
<td>156.7%</td>
<td>64.52%</td>
</tr>
<tr>
<td>Duration Detectable Faults</td>
<td>—</td>
<td>100% path delay</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

No time borrowing implementation

For comparison purposes, we use “Reference” version, which is the unprotected icyflex version, synthesized without time borrowing by using a cell library targeting 1.10V supply voltage. The area and power of the “Reference” version are given in normalized terms (100% area and 100% power) in column 2 of table 1.

Column 3 shows the results for the GRAAL implementation, using the same cell library and no time-borrowing. We observe that this implementation achieves detection of faults that increase by 100% the delay of the affected path. This duration is very high and is achieved by means of very low cost (3.06% area and 3.95% power cost). Thus GRAAL architecture can achieve very high reliability in future highly defective technologies that could increase dramatically PVT (process, voltage, and temperature) variations, and circuit aging, due to a very aggressive reduction of device geometries. Thus, GRAAL is very suitable for the Cells framework.

To validate the efficiency of GRAAL also with respect to flip-flop based design, column 4 shows the area and power of the flip-flop implementation of icyflex, using
the same cell library and targeting the same clock frequency (150 MHz). We observe that, the flip-flop version requires 7.28% less area with respect to the “Reference” latch-based version, but much larger power (56.77%). Thus, the GRAAL no-time borrowing implementation achieves its high fault detection efficiency (faults that increase by 100% the delay of the affected paths) at a 11.73% area cost with respect to the unprotected flip-flop-based implementation, but at much lower power (33.66% less than the unprotected flip-flop-based implementation).

No time borrowing implementation

The implementation presented in the previous section does not use time borrowing and achieves the highest robustness, as faults of very high duration are detected (i.e. faults of so large duration that may duplicate the delay of the affected path). Note that the detection of faults of high duration can also be exploited for reducing power dissipation, by aggressively reducing the supply voltage level, and using GRAAL to detect and recover the resulting timing errors. However, a more efficient approach consists in using combining GRAAL with time borrowing.

To reduce power, we used the TT/0.90V/125°C cell library, which is designed for 0.90V supply voltage. However, as the delays of this cell library are higher than the delays of TT/1.10V/125°C, achieving operation at 150 MHz will impact power dissipation and reduce the gains achieved by the reduction of supply voltage. Thus, to maximize the power reduction, we used the Design Compiler to synthesize the time-borrowing version of icyflex2 for the TT/0.90V/125°C.

Column 5 in table 1 shows the results of the unprotected version of icyflex2 using this cell library and time borrowing, and targeting 150MHz clock frequency. This version requires 0.98% more area with respect to the “Reference” version of column 2, and reduces power by 35.48%.

Column 6 in table 1 shows the results concerning the GRAAL version of the time-borrowing implementation of icyflex2, for the same cell library and clock frequency. This version detects faults of duration up to 72.4% of the delay of the path affected by the fault. Thus, the GRAAL architecture detects faults of very large duration, even for time borrowing implementations. This significant advantage is achieved at the small cost of 7.18% more area and 6.29% more power with respect to the unprotected time borrowing implementation of column 5. In comparison with the unprotected flip-flop implementation, this high detection efficiency is achieved at the cost of 16% more area and at the drastic advantage of 45.19% less power. Hence, the combination of GRAAL with time borrowing is also very suitable for the Cells framework, as power reduction is very important for Cells, and fault detection efficiency remains very high.

Protecting flip-flop based designs

Note that though the preferred error detection scheme in Cells is GRAAL, Cells is also compatible with any other error detection scheme. For flip-flop based designs, two double-sampling architectures were developed:

- The one, referred a ADDA architecture [6][7], allows operating the same design in 3 modes: the first is the mode used in earlier double-sampling schemes; the second allows early failure prediction; the third allows detecting transient faults of large duration, that could be encountered in hostile environments like space.
- The other, presented in [7][8], reduces drastically the area and power penalties of the double-sampling architecture, by removing the redundant sampling element. With this solution, the only cost for implementing double-sampling consists in the comparison of the flip-flop outputs against their inputs.

The evaluation of these architectures is on-going by implementing them in the LEON3 processor.

References

[4] Icyflex2 user manual, from CSEM.
Memory and Interconnect Self-Repair for High Defect Densities in the Cells Framework

Members: Michael Nicolaidis, Lorena Anghel, Mounir Benabdenbi, Panagiota Papavramidou, Vladimir Pasca

Keywords: PVT variability; aging; timing faults; soft errors; memory repair; design for reliability, yield, and low-power.

Contracts: TOETS

Embedded memories occupy the largest part of SoCs and include even larger amounts of transistors. As memories are designed very tightly to the technology limits, they are more prone to failures than other circuits. Thus, they concentrate the large majority of fabrication defects and may affect yield adversely. In addition, failure rates are expected to be exacerbated as we approach the ultimate limits of CMOS, and should further worsen in post-CMOS technologies. Furthermore, low power requirements, which are stringent in modern electronic systems, will also be exacerbated as we move towards ultimate CMOS and post-CMOS, requiring aggressive reduction of supply voltage. Unfortunately, voltage reduction results in additional failures, as weak memory cells will not operate correctly at reduced voltage levels. Furthermore, as ultimate CMOS and post-CMOS are expected to exacerbate PVT variability and circuit aging, the ratio of weak cells will sharply increase. As a consequence, techniques enabling coping with high defect densities in memories are required in the Cells framework.

Memories and interconnection buses have very regular structure and enable easy repair. Various efficient memory and interconnections Built-In Self-Repair (BISR) schemes have been proposed in the past, and several of them are already industrial practice. However, for high defect densities, conventional memory repair schemes will induce excessive area and power penalties. Thus, innovative solutions are required able to drastically reduce these penalties. The ECC-based memory repair scheme, introduced by our group, is the only known scheme able to cope with high defect densities. However, while the ECC-based repair scheme drastically reduces area and power penalties with respect to conventional schemes, power penalty is still non-negligible. Furthermore, implementing self-diagnosis for the ECC-repair scheme, further increase area and power penalties. Our recent developments have eliminated these issues [1]. First, a new family of memory test algorithms was proposed eliminating diagnosis-related area and penalties at the expense of test length [2][3]. Then, an iterative diagnosis architecture was proposed [4], trading test length with area cost. In a third innovation [5], a hardware architecture is proposed drastically reducing power dissipation for both ECC-based repair and non-ECC-based repair. Further reduction of this power is achieved by introducing an approach that rearranges the faulty addresses stored in the repair circuitry, in a manner that disables most of the time a part of this circuitry [6]. Thanks to these improvements and the use of the ECC-based repair, an impressive power reduction is achieved with respect to conventional repair.

For instance, for a SoC comprising a total of 10 Gbit memory capacity distributed over 300 embedded SRAMs using words of 32 bits (plus 7 ECC check bits), conventional repair for a $10^{-3}$ probability that a memory cell is faulty, conventional repair induces a huge power penalty of 1629%, which is due to the fact that repairing such high fault rates requires using large repair CAMs, and also that CAMS are power hungry. Then, the combination of ECC-repair with the improved repair architectures in [5][6], reduces this penalty to only 7.36% [6].

Hence, these developments provide a comprehensive framework enabling low cost memory repair for high defect densities, which is very suitable for the Cells framework. In other developments, we proposed a BIST architecture enabling the cooperation of transparent BIST with ECC-based repair [7]. Transparent BIST is an approach we introduced at 1996, and further developed and used by numerous authors and fault-tolerant systems. This approach transforms memory test algorithms into reversible processes, which preserve the information stored in the memory. Transparent BIST is essential in the context of Cells in order to preserve application context during periodic memory self-tests. The fundamental problem for making cooperating transparent BIST with ECC-based repair is that transparent BIST uses signature analysis allowing determining if a memory is faulty (go-nogo test), while ECC-memory repair requires more subtle diagnosis (i.e. determining if there are memory words containing multiple faulty cells). While the signature analysis alone does not allow making such diagnosis, and also the error detection and correction circuitry alone can also misdiagnose words containing more than two faulty cells, the new transparent BIST architecture we have developed in the context of Cells resolves this problem thanks to a subtle cooperation between the signature analysis and the ECC circuitries.
Integrating these approaches in the Cells project will enable healing tera-device processor chips comprising billions of defective memory cells. For instance, a massively parallel tera-device chip comprising thousands of embedded memories of a 1 Tera-bit total capacity and affected by a $10^3$ defect density, will comprise $10^9$ faulty memory cells (including definitely faulty cells as well as weak cells failing during aggressive voltage reduction modes). The Cells project will be able making such a chip operational, and last but not least, this unprecedented goal will be achieved at very low area and power penalties.

Interconnections may also represent an important reliability challenge, especially for 3D systems where the vertical interconnects (TSVs) may be affected by high defect densities. Efficient Built-In Self-Repair schemes for TSV interconnect, using parallel message transmission and serial message transmission, allow resolving this issue in cost effective way.

References


Array-Level Approaches in the Cells Framework

Members: Michael Nicolaidis, Nacer-Eddine Zergainoh, Amir Charif, Michael Dimopoulos

Keywords: Ultimate CMOS and post-CMOS technologies, high defect densities, reliability, yield, low-power, massively parallel single-chip tera-device computers, array level self-adaptive approaches.

Adaptive fault tolerant routing
In complex grids comprising thousands of nodes, routing algorithms based on routing tables are congestion prone and have low adaptability to new failures. Hence, we developed distributed algorithms using local opportunistic decisions (get another path when a node/router/link is faulty or congested). Our simulation results show that they easily scale for grids comprising thousands nodes; tolerate multiple faulty nodes/routers/links; avoid congestions; and cope with new failures occurring at any time. But, as they are based on local decisions, they are deadlocks prone. To cope with we use virtual networks and pertinent rules acting at the local level.

Adaptive fault tolerant task scheduling and allocation and error recovery
At a first step we integrate in a single algorithm our FT routing approach (discussed above) with distributed FT scheduling and allocation. We also integrate in the same algorithm check-point-free rollback recovery. This is done by an approach which: uses an hierarchical task organization into parent-children trees; maintains this hierarchy during execution; and if a persistent fault occurs in a resource executing a child it goes back to the parent to redistribute the workload in fault-free resources. This way we avoid saving the internal states of the grid to external media (no check-pointing), which would congest the grid I/Os, and we can recover correct operation even after the occurrence of any multiple fault (by going up in the tree until a fault-free parent, which redistributes the aborted workload to fault-free resources).

Error recovery algorithms based on check-pointing were also implemented. A basic issue is related to the fact that as the state of the system is huge; we have to perform check-pointing for partial states of the system (e.g. states of tasks), while preserving its overall coherence. To cope with this issue a check-pointing algorithm for parallel processors maintaining system coherence was developed.

Variability awareness
In further developments we extended the algorithm to enable variability aware and power aware task scheduling and allocation. Again we use a distributed non-deterministic approach to handle computation complexity. One of the developed schemes maps different groups of tasks into different regions of the grid according to the energy dissipated by each group and the power-dissipation characteristics of the regions. Then, the leader of each region maps task clusters into sub-regions, and the leader of each sub-region maps each task to a node, by using similar power dissipation considerations. The clock frequency required for meeting the deadline of each task and the power/reliability priorities of the task are encapsulated in the header of each task. Each node knows its frequency/Vdd operating domains for various error occurrence rates (by monitoring its concurrent error detection signal). Thus, it can determine its clock frequency and Vdd level to reach the task deadline, minimize the energy dissipated, and achieve the target reliability level (in terms of error occurrence rate). Thus, circuit parameters (clock frequency and Vdd) are continuously regulated to minimize power, preserve reliability and adapt to circuit degradation induced by aging.

Our work on array level self-adaptive algorithms was presented in numerous articles published in various international conferences during the past few years [1][10]. The next step of our efforts concern the integration of these algorithms (together with the other techniques used in Cells), in a unified framework implemented in the GEM5 simulator and the Structural Simulation Toolkit (SST).

References


