Organic Computing at the System on Chip Level

A. Bouajila, J. Zeppenfeld, W. Stechele, A. Herkersdorf
Institute for Integrated Systems
Munich University of Technology
D – 80290 München

A. Bernauer, O. Bringmann*, W. Rosenstiel
Wilhelm Schickard Institute for Informatics
University of Tübingen, D – 72076 Tübingen

* FZI – Forschungszentrum Informatik
University of Karlsruhe, D – 76131 Karlsruhe

***Abstract*—The evolution of CMOS technologies leads to integrated circuits with ever smaller device sizes, lower supply voltage, higher clock frequency and more process variability. Intermittent faults effecting logic and timing are becoming a major challenge for future integrated circuit designs. This paper presents an Organic Computing inspired SoC architecture which applies self-organization and self-calibration concepts to build reliable SoCs with lower overheads and a broader fault coverage than classical fault-tolerance techniques. We demonstrate the feasibility of this approach by example on the processing pipeline of a public-domain RISC CPU core.**

I. INTRODUCTION

System on Chip (SoC) semiconductor components have reached a functional diversity, heterogeneity and complexity which fully deserves their consideration as a "system". According to Encyclopaedia Britannica [1], a "system" is a complex man-made construct which consists of several subsystems of different functionality, whereby the sub-systems interact heavily and are strongly dependant on each other. This definition applies to what we know today as the internet, to the embedded electronic infrastructure in mid- and high-class automotive vehicles, as well as to individual components of such systems. SoC components may contain multiple processor cores capable of up to several giga-operations per second, along with sophisticated memory hierarchies, interconnected on-chip bus systems or networks on chip (NoC), a diversity of system and network interfaces, as well as application specific hardware accelerators.

The ever increasing numbers of SoC components packed into a single chip have only been possible due to the steadily decreasing size of the underlying devices. This trend towards MOSFET transistor channel lengths in the deep sub-micron range (headed towards <10 nm) has introduced a number of technological challenges which will become even more acute in the future. Previously negligible effects such as those resulting from ionizing radiation (soft errors) will come into play for combinatorial logic in addition to memory, while existing problems such as dealing with variations in environmental temperature or supply voltage have become exceedingly difficult to solve. In addition, parasitic effects during the manufacturing process will result in increased leakage, sub-threshold and gate currents, each contributing to a floor of substantial static power dissipation. These trends will lead to a decrease in the reliability and life-time expectancy of complex VLSI and SoC systems [2] and will result in significantly decreasing production yields.

We believe that the aforementioned observations require a fundamental conceptual change in future SoC design processes and architectures. It will become necessary to deviate from the standard practice that 100% of all SoC circuit elements must successfully pass manufacturing tests in order to be shipped as a product. Instead, future SoCs will have to "learn to live" with temporary and sporadic faults and errors. In analogy to the immune system of living beings, such faults must be localized and dynamically corrected before they have a chance to propagate and "infect" other parts of the system. This shall be made possible by means of autonomous self-healing concepts [3]. We refer to an SoC that demonstrates such "learn to live with defects" properties as an Organic or Autonomic SoC (ASoC).

We currently have a yet incomplete understanding of what an ASoC architecture and design method may eventually entail, but we know that for broad industrial acceptance any changes must be of an evolutionary (rather than revolutionary) nature. This means that it must remain possible to reuse existing SoC IP (Intellectual Property) building block libraries, preserving the huge investments companies have already made into functional macros, cores, as well as into the associated design tool environments. Moreover, the total area and power overheads required by organic or autonomous enhancements to a system must not exceed a certain fraction of the area savings or decrease in power made possible through the more aggressive parameter scaling in the new technology. If they did, there would be little to no incentive for progressing to an autonomous design.

If we assume a target area overhead of 10-15% for the autonomous self-x functions, which seems acceptable for many SoC application domains, we can see from Table 1 that this assumption constrains the abstraction level at which organic computing can be applied. Since most types of error (especially soft errors) originate at the device level, it is de-

This work is funded by the German Research Foundation (DFG), under the SPP-OC 1183 research program.

TABLE I. SOC ABSTRACTION LEVELS

Abstraction Level	Example Constituent	Number of Constituents	Devices / Constituent
System	SoC	1	$> 100\times10^6$
Architecture	CPU, Bus, SRAM	some 10s	few 10×10^6
Circuit	EXE, FPU	mult. 1000	mult. 10^4
RTL	+, ×, Reg	10^5	100
Gate	NOR	$> 10\times10^6$	10
Device	MOSFET	$> 100\times10^6$	1

sirable to provide error detection and correction logic at as low a level as possible. Due to the 10-15% area overhead target, it is clear that not much reliability improvement can be accomplished with one tenth of a MOSFET. Similar reasoning holds for the gate and RTL levels, where the typical functional constituents consume few 10 to 100 MOSFETs. For this reason we propose an Autonomic SoC architecture which provides organic enhancements to building blocks starting at the circuit level of abstraction.

The abovementioned ASoC architecture is presented in Section II. Section III presents the underlying fault model required to cover single-bit transient and timing errors. Section IV presents a novel autonomous CPU datapath pipeline protection scheme which is able to detect and correct transient and timing errors (self-healing) with minimum performance impact and low area overhead. Section V concludes the paper.

II. ASOC ARCHITECTURE

Figure 1 shows the proposed ASoC architecture [4] which essentially splits the SoC into two logical layers: the functional layer which contains the usual IP components or Functional Elements (FEs), and the autonomous layer which consists of Autonomic Elements (AEs) and an interconnect structure among various AEs. FEs are either general purpose CPUs, memories, on-chip busses, special purpose processing units (PUs) or system and network interfaces as in a conventional, non-autonomous design. AEs on the other hand contain any extensions necessary to improve the reliability of the FE and, thus, convert the FE-AE pairs into autonomous units. In analogy to current IP libraries making up the functional layer, the AEs shall eventually represent a corresponding autonomous IP library. This FE/AE split represents only a logical structuring concept; in reality both FEs and AEs will be integrated on the same CMOS substrate.

The proposed ASoC architecture will integrate self-x properties. In fact, each AE contains a monitor which will sense the state of the corresponding functional IP to detect errors, an evaluator which will evaluate the state of the IP and an actuator which will initiate any necessary corrective actions. Here we notice that this architecture implements the self-correcting and self-healing properties, and that FE and AE form a closed observer-controller loop [5]. The information gathered from the AE monitors can be used to compute the error rate of an IP. This error rate will be used to adjust

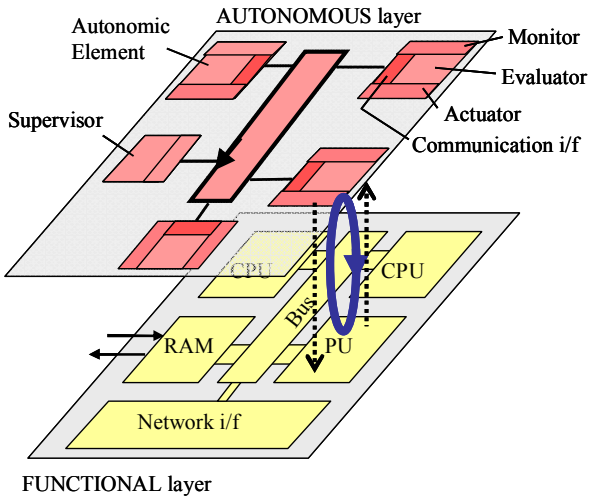

Figure 1. Autonomic SoC Architecture

frequency, voltage or other parameters (self-calibration property) to correct temporary FE malfunctions due to environmental variations. If a functional IP becomes unusable due to excessive error rates, the AE, using information shared through the autonomous interconnect, will replace the defective IP with a spare IP (self-organization property).

Our Autonomic SoC architecture presents a smooth shift from present non-autonomous to autonomous designs. In fact, semiconductor companies can continue using existing IP-libraries by extending them with the proper autonomic elements. This will preserve the huge investments used to build those IP-libraries.

III. FAULT MODEL

In order to detect and correct errors, we need to define a fault model which formally describes the nature of the errors under investigation. As stated above, we are primarily concerned with transient and timing faults caused by ionizing radiation or variations at the technology or device level, and we want to reliably detect and correct them at the circuit (micro-architecture) level.

Timing errors have their root in any form of process / environmental variation (e.g., temperature, Vdd) or crosstalk coupling between signals, and result in a set-up or hold-time violation of register elements (latches or flip-flops). Tran-

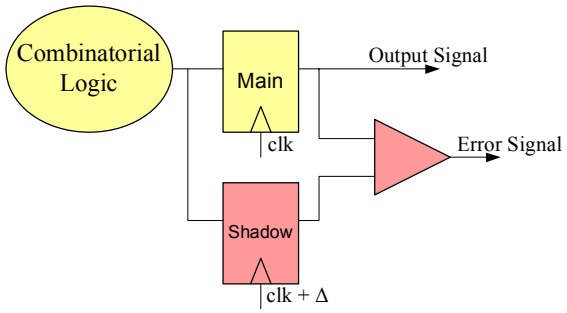

Figure 2. Time redundancy error detection technique

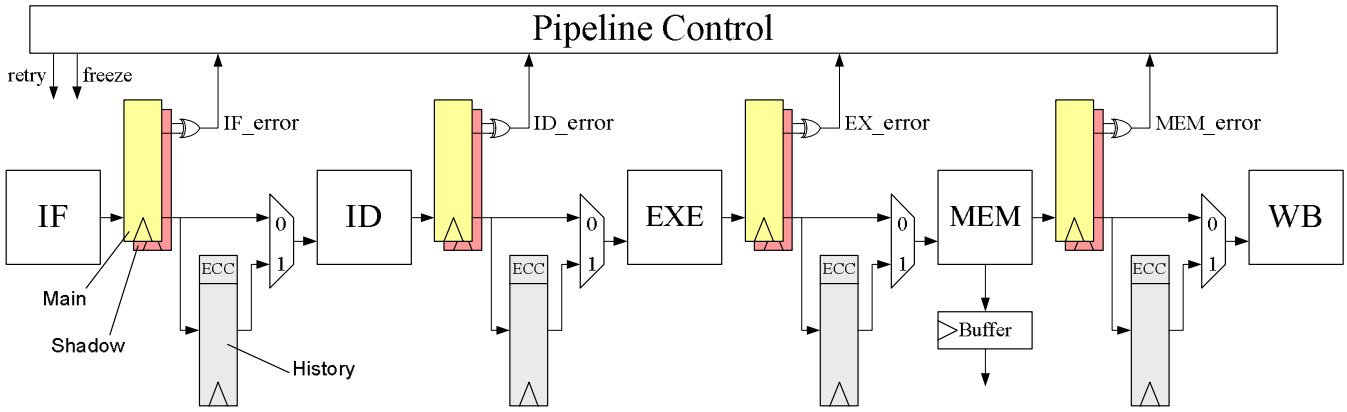

Figure 3. Self-Healing CPU Pipeline for Timing and Transient Errors (Only one comparator shown for clarity)

sient faults are due to particles (alpha particles or cosmic neutrons) striking a node in the circuit. If the node is a latch or flip-flop, this strike can flip its value, resulting in a so-called single event upset (SEU). For nodes within the circuit logic, a voltage pulse will be created and propagate to a register element, producing a single event transient (SET). This voltage pulse may potentially be latched if it propagates through a sensitized path and arrives at the register during its active time window. Therefore, we model transient errors at micro-architectural level either as a voltage pulse propagating to a flip-flop/latch, or as a bit-flip directly in that flip-flop/latch [6].

IV. Self-Healing CPU RISC Pipeline

Nicolaidis [7] suggested a protection scheme based on the concept of time redundancy to protect circuits from both transient and timing errors. The idea is to add a shadow flip-flop to every flip-flop in the circuit that should be protected (Figure 2). The shadow flip-flop works with a delayed clock, the delay of which is adjusted such that it is larger than the expected delay of the critical path under worst case conditions and also larger than the largest possible pulse duration created by a particle strike in a logic node. In case of a fault, either the main or the shadow flip-flop will therefore sample the correct value. Errors are detected simply by a comparison of the latched values in the main and shadow flip-flops. It is obvious that a particle directly hitting a flip-flop will also generate a disagreement between the main and shadow flip-flop. Simulations [8] have shown a fault coverage for transient errors of more than 99% (tested for multipliers and adders).

Chardonnereau et. al. [9] also demonstrated the timing and transient error detection capability of the shadow flip-flop technique on a Leon processor pipeline. However, error detection in itself is not enough to increase the reliability of a processor. Austin presented a variation of the Nicolaidis approach (referred to as the Razor technique [10]) to achieve timing error detection and correction in an Alpha processor pipeline. The drawbacks of the Razor approach are its inability to detect transient errors affecting the shadow latches, and that it uses pipeline flushing as its error correction mechanism. Flushing results in different performance penalties depending on which stage of the pipeline is affected, which may exhibit a considerable performance loss especially for deep processor pipelines and high error rates.

In the following we present a new scheme for CPU pipeline protection which extends the Nicolaidis shadow flip-flop technique to also correct both timing and transient errors. For correction we introduce a customized micro-rollback [11] technique which does not require a complete pipeline flush but incurs a constant pipeline stall penalty of only two cycles instead.

We integrated this protection scheme on a Leon2 Sparc-V8 RISC pipeline [12], adding a shadow register to every register separating the Leon2 pipeline stages (Figure 3). Each pair (main and shadow register) is connected to a comparator which generates an error detection signal in case of a register mismatch. The error detection signals of all pipeline stages are OR'ed in the Pipeline Control unit to generate one error detection signal for the entire pipeline. As each pipeline stage has timing and transient error detection capabilities, the entire pipeline is thereby protected against timing and transient errors. It is also possible to locate the pipeline stage in which the error occurred.

Since the error signals are available only after the rising edge of the delayed clock at the shadow register (Figures 2 and 3), the pipeline input registers have already been overwritten with the following value. Therefore we cannot simply re-calculate the stage in which the error occurred. To solve this problem, we suggest adding history registers in order to keep track of the previous values of each stage's input values. Since the detection latency is exactly one cycle we only need to store the most recent values of the input registers. The history registers themselves are protected against single-bit corruption by means of ECC coding.

Figure 4 details the error handling mechanism. Whenever an error occurs, it will be detected in the subsequent cycle. In this detection cycle, the pipeline stage will be frozen, i.e. we do not update the output registers in any pipeline stage. The corresponding freeze signal is derived directly from the aggregate of the error signals. The history registers (the con-

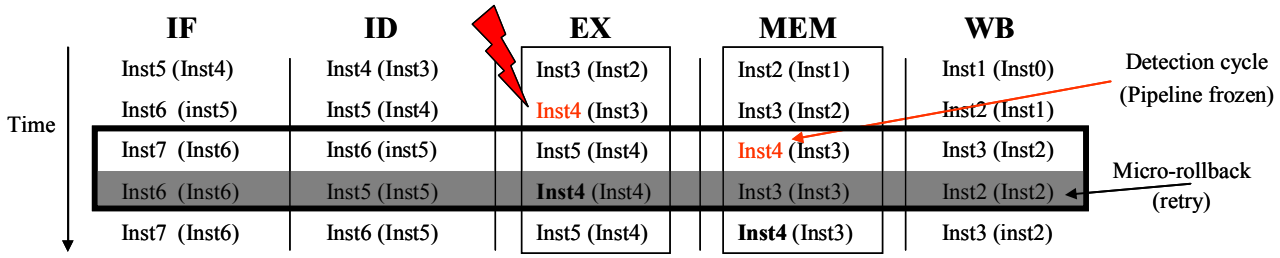

Figure 4. Timing of Error Handling

tents of which are parenthesized in Figure 4) are not updated either. In the subsequent (retry) cycle, all pipeline stages will be restarted using the history registers' contents as input values. Thus, our scheme is a RISC pipeline implementing a customized error-triggered rollback with a fixed rollback distance of one cycle.

This self-healing CPU pipeline is an example of extending an existing functional element with an autonomic element in the context of our ASoC architecture (Figure 1). The AE's monitor consists of the shadow registers, comparators and history registers. The actuator is reduced to the micro-rollback pipeline control. The evaluator is not present as such because every detected error implies the same action in the form of a pipeline micro-rollback. Note that if an error occurs within the memory access (MEM) stage, this error must be caught within one cycle to prevent a corruption of memory and hence error propagation to other devices.

Initial simulations using C program tests provided with the Leon2 processor code (version Leon2-1.0.32-xst[12]), confirmed that the detection and correction of an error does indeed result in a penalty of only two cycles, no matter where the error occurs within the pipeline. As shown in Table 2, preliminary synthesis of the protected Leon2 Core using the Xilinx ISE v8.1i for a Virtex-II Pro FPGA (XC2VP30) resulted in an area overhead of 23%. Although this is a good deal larger than the targeted 10-15% area overhead, we are confident that optimizing the code (which is currently in a behavioral state) will help approach the 15% margin. It should also be noted that large gates (such as the OR merging all error signals) require a large number of FPGA look-up-table (LUT) resources, but can be implemented very efficiently on an ASIC platform.

V. CONCLUSION

Building reliable System on Chip components from inherently unreliable elementary MOSFET devices and circuitry is a key challenge in future IC design. MOSFET devices are becoming increasingly sensitive for parameter and environmental variations as well as for ionizing radiation when approaching scaling limits in the order of 10nm active channel length. We believe that this challenge demands conceptually new approaches to SoC chip design and SoC architectures. In this paper we presented an organic computing based SoC architecture with embedded self-correction properties. By example of a RISC CPU pipeline, we introduced a

TABLE II. SYNTHESIS RESULTS ON A VIRTEX 2 PRO FPGA

	Leon2 System	System Protected	Inc. %	Leon2 Core	Core Protected	Inc. %
Slices	5795	6606	14	3473	4289	23
Registers	2521	3984	58	1252	2744	119
LUTs	10209	11601	13	6140	7549	23

new technique for autonomous detection and correction of single-bit transient and timing errors at minimum performance impact and low area overhead. This represents a small initial step towards our vision of an organic or autonomic SoC architecture. Future work will focus on demonstrating the viability and effectiveness of our ASoC approach by establishing collaboration among multiple autonomous control units of different functional IP cores.

REFERENCES

[1] Britannica 2001 Deluxe Online Edition, Encyclopeadia Britannica.

[2] De Micheli, G.: „Designing Robust Systems with Uncertain Information", Asia and South Pacific Design Automation Conference (ASPDAC 03), January, 2003.

[3] S. Mitra, W.-J. Huang, N.R. Saxena, S.-Y. Yu, E.J. Mc Cluskey: "Reconfigurable Architecture for Autonomous Self-Repair", IEEE Design and Test of Computers, pp. 228-240, May-June 2004.

[4] Gabriel Lipsa, et al "Towards a Framework and a Design Methodology for Autonomic SoC", 2nd IEEE International Conference on Autonomic Computing, 13-16 June, Seattle, USA..

[5] Thorsten Schöler, Christian Müller-Schloer: An Observer/Controller Architecture for Adaptive Reconfigurable Stacks. ARCS 2005: 139-153

[6] Michael Nicolaidis, "Design for Soft Error Mitigation", IEEE Transactions on Device and Materials Reliability, Vol. 5, NO. 3, September 2005.

[7] Michael Nicolaidis, "Time Redundancy Based Soft-Error Tolerance to Rescue Nanometer technologies", 17th IEEE VLSI Test Symposium, 1999.

[8] Lorena Anghel, Michael Nicolaidis, "Cost Reduction and Evaluation of a Temporary Faults Detection Technique", DATE2000.

[9] D. Chardonnereau et. al. "Fault Tolerant 32-bit RISC Processor: Implementation and Radiation Test Results", downloaded on july 2006 from http://www.iroctech.com/pdf/RISC_rad_results.pdf

[10] Dan Ernest, et al "Razor: A Low-Power pipeline Based on Circuit-Level Timing Speculation", the 36th Annual International Symposium on Microarchitecture (Micro-36), December 2003

[11] Yuval Tamir and Marc Tremblay, "High-Performance Fault-Tolerant VLSI Systems Using Micro-rollback", IEEE Transactions on Computers, Vol. 39, NO. 4, April 1990.

[12] Gaisler Research Homepage: http://www.gaisler.com/cms4_5_3/