Towards a Framework and a Design Methodology for Autonomic SoC

Gabriel Lipsa[1] Andreas Herkersdorf[2] Wolfgang Rosenstiel[3] O. Bringmann[1] W. Stechele[2]
FZI, Microelectronic System Design, Karlsruhe[1]
Technical University of Munich, Institute for Integrated Systems[2]
University of Tuebingen, Department of Computer Engineering[3]
{lipsa, bringman}@fzi.de { a.herkersdorf, w.stechele}@ei.tum.de rosenstiel@fzi.de

Abstract

This paper proposes autonomic or organic computing principles to be applied to hardware design methods for future SoC solutions. Incorporating self-calibration, fault tolerance or even self-healing concepts into integrated circuit systems represents a major conceptual shift, which requires new design processes and tools. In the future, guarantee of functional correctness at the chip level will include self-configuration of adaptable components and flexible interfaces supporting a flexible component composition within complex SoC systems.

1. Introduction

The 2003 ITRS Roadmap [It03] projects micro- and nanoelectronic integrated CMOS circuits to witness a continued capacity growth rate corresponding to doubling transistor count every two to three years ("Moore's Law"). Today and in the future the primary driver will no longer be how to integrate even more transistors on a single chip, but how to develop such complex ICs with affordable cost and within reasonable time frames.

We propose autonomic or organic system properties [Ho01], self-configuring, self-administrating and self-healing, to be incorporated into future IC designs and be supported by corresponding tools. So, our proposition is to rededicate a fraction of the abundant transistor capacity of future SoCs to implement organic computing principles for the sake of higher fault tolerance, performance, power efficiency, easier system diagnosis and the capability to autonomously adapt to changing environmental conditions – be it either externally imposed workloads or temperature variations.

2. Overview of ASoC Framework

Our focus is on the chip level hardware layer of SoCs (Systems on Chip) solutions. Translated into the world of semiconductor IC systems, the future may look as follows: with increasing, externally imposed workloads, the clock frequency and supply voltage of individual processor cores are increased to elevate processing performance. At the same time, critical transactions on on-chip interconnect buses are prioritized and the bus bandwidth of less critical transactions is reduced. Redundant building blocks, being deactivated under normal operating conditions, are activated on demand to increase system performance.

Self-organization also has to deal with the phenomenon of graceful degradation of SoCs, or said in other words, to guarantee minimum SoC functionality and performance in the event of entirely or partially failing system components for which no redundant replacement is available. The self-healing concepts attempt to replace a faulty processing unit with an equivalent counter part which will adopt the functionality of the failing element. The replacement unit can either be an idle stand-by element, or a processing unit that performs other tasks prior to the error occurrence.

The self-healing concept does not just mean to fix an error, but also to prevent errors in cases where the system risks getting into a critical state (e.g. performance wise due to component overload, or temperature wise due to excessive power consumption). The ASoC will supervise the behavior of its constitutional components and build up fallback scenarios, which are activated under certain trigger conditions before system failure. Once a fallback solution has been deployed, the self-organization process will again try to improve performance and eventually switch back to the original system configuration, so the fallback solutions have to be good enough to respect the constraints.

The ASoC architecture platform related aspects are closely interlocked with the ASoC design methodology. The ASoC design methodology defines the scope and practices within which both the functional and autonomic layers of the ASoC are designed.

3. Autonomic SoC Architecture

Figure 1 shows the proposed ASoC architecture platform. The ASoC is split into two logical layers: The functional layer contains the IP component or Functional Elements (FEs). The functional layer consists of today's IP blocks, which need just some small modification in order to be able to communicate their status to the AEs, and then the AEs will be able to control their voltage and frequency. The autonomic layer consists of Autonomic Elements (AEs) and an interconnect structure among the AEs. In analogy to the functional layer IP library, the AEs shall eventually represent an autonomic IP library (AE_lib).

Each AE contains a monitor or observer section, which senses signal or state information from the associated FE, an evaluator, which merges and processes the locally obtained information with state from other AEs and/or memorized local knowledge, and an actuator, which executes a possibly necessary action on the local FE. The combined evaluator and actuator can also be considered as a controller. Hence, our two-layer Autonomic SoC architecture platform can be viewed as distributed (decentralized) observer-controller architecture.

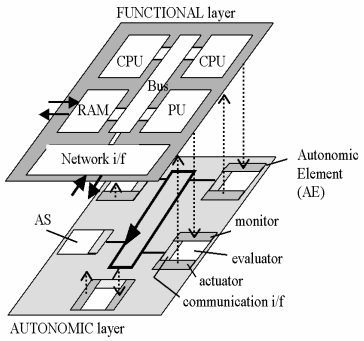

Figure 1 Two-layer autonomic SoC platform

There will also be a special type of AE, an autonomic supervisor (AS), which has no counter part on the functional layer. The AS will monitor the correct operation of and interaction between other AEs. Very important will be the AIConS (Autonomic Interconnect Structure), which will link all the AEs with the central AS. The AEs take decisions based on the local status plus the communicated information from the other AEs. Since only local actions which are "in harmony" with the local functional macro are allowed, this approach implicitly guarantees controlled emergence.

4. Design Methodology for Autonomic SoC

The methodology has to consider fault tolerance as an additional parameter beside, area, performance and power consumption, and will provide a technique to build SoC architecture with autonomic or organic properties. The design flow accounted to this methodology, is depicted in the Figure 2.

In short, the methodology will consider the characteristics of the architecture and the requirements from the application. Making use of them, we will obtain a model, consisting of AE and FE templates (AE/FE Model). Upon this model, the reliability driven architectural optimization and evaluation will be performed.

The **reliability driven architectural optimization** will give notice of how many resources are needed in order for the application to work. Because the system has to be able to work under the required constraints, we propose to solve this problem using redundancy and reorganization. This step of the methodology will compute all the resources needed, including the redundant ones.

After the selection of both the FE and the corresponding AE, a **FE/AE Model** is obtained. Upon this model, an **evaluation** has to be performed to see if it complies with the requirements. The results of the evaluation will be used, if the architecture will not be accepted, as **evaluation parameters** for selecting another architecture. The evaluation of the FE/AE model will contain three parts: the exploration of the architecture, the change of the system's state after a certain failure and the evaluation of the new state. Our proposal is to build a dynamic fault tree for exploring the architecture. From the existing approach on dynamic fault tree analysis [DuAs01], we bring two new concepts. First, the obtaining of the new state or leaf, we have to take into consideration the self-organizing algorithm and the current state in order to obtain the new state. The second concept that we are bringing is an analysis performed over the state, which will be chosen.

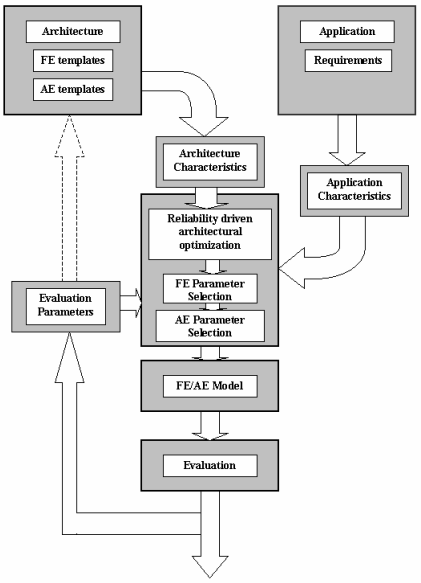

Figure 2 Design Flow

5. Conclusions and Outlook

In order to prevent isolated, proprietary solutions, the ASoC framework must seamlessly fit into the SoC design methodology and form an open commonly available industry standard. For the time being, our target is an ASoC framework where the full spectrum of autonomic behavior is considered by the designer prior and during system architecture development. This will result in optimized AE operations and inter-AE communication structures developing towards truly emergent system behavior.

[DuAs01] Joanne Bechta Dugan and Tariq S. Assaf. "Dynamic Fault Tree Analysis of a Reconfigurable Software System". *The 19th International System Safety Conference*, Huntsville, Alabama, September, 2001.

[Ho01] Horn, P: "Autonomic Computing: IBM's Perspective on the State of Information Technology", IBM Corporation, Oct. 2001, http://www.research.ibm.com/autonomic/manifesto

[It03] "International Technology Roadmap on Semiconductors 2003"