

CARUSO – Project Goals and Principal Approach

Uwe Brinkschulte^{*}, Jürgen Becker[#], Klaus Dorfmueller-Ulhaas⁺, Ralf König[#], Sascha Uhrig⁺, and Theo Ungerer⁺

^{*} Department of Computer Science, University of Karlsruhe, 76131 Karlsruhe

[#] Department of Electrical Engineering, University of Karlsruhe, 76131 Karlsruhe,

⁺ Department of Applied Computer Science, University of Augsburg, 86159 Augsburg,
brinks@ira.uka.de, {becker, koenig}@itiv.uni-karlsruhe.de,
{ungerer, dorfmueeller, uhrig}@informatik.uni-augsburg.de

Abstract: This paper proposes CARUSO – a new SoC approach that emphasizes Connectivity, Autonomic/Organic computing principles, Real-time, and Ultra-low power requirements. The requirements shall be fulfilled by a multithreaded processor core within a reconfigurable SoC. Helper threads running with low priority in own thread slots concurrent to the application implement autonomic/organic managers that monitor the application and decide if self-configuration, self-healing, self-optimization, or self-protection must be triggered.

1 Introduction

State-of-the-art embedded computing systems with limitations in space and energy consumption are manufactured as so-called Systems on Chip (SoC), where all electronic components are placed on a single chip. CARUSO (Connective Autonomic Real-time Ultra-low-power System on Chip) is a new SoC project aiming to integrate hardware and software for high performance embedded computing with respect to further requirements:

- **Autonomic and Organic Computing (AC/OC)** aim at improved controllability of complex systems and require future systems to fulfill the self-x properties, i.e. self-configuration, self-healing, self-optimization, and self-protection. Future systems will therefore act more independently, flexible and autonomously, i.e. they will be more life-like.
- **Connectivity** to enable several SoCs to dynamically form networks. Future embedded systems will consist of multiple small computing components which cooperate to solve a common task. The principles of autonomic/organic computing can require a change of the network topology.
- **Real-time** capabilities (hard, firm or soft) to keep the timing requirements introduced by many embedded applications.
- **Ultra-low-power** to increase the battery life time for mobile applications and to reduce the heat dissemination in temperature-critical environments.

Our solution is to fulfill the demands at the hardware level by combining a multithreaded processor core with reconfigurable hardware within a SoC. So called helper threads are running at their own hardware thread slot concurrent to the application to implement autonomic/organic managers at system software and middleware level. These managers monitor the application and decide if self-configuration, self-healing, self-optimization, or self-protection techniques must be triggered at their respective levels. The special real-time scheduling scheme (GP scheduling as proposed in [1] is used as starting point) isolates the autonomic/organic manager threads from the application real-time thread(s). Application threads as well as the autonomic/organic managers can take advantage of the reconfigurable chip component. The currently needed processing power defined by the real-time scheduling of the active threads is used to manage energy consumption of the SoC (as proposed in [2]).

2 State of the Art

Autonomic Computing [3, 4] has been introduced by IBM at the beginning of this millennium. The basic idea is to make computing systems behave more like organic entities, which adapt to new challenges, heal themselves after injuries and protect themselves against attacks. The Organic Computing Initiative [5, 6] launched by the two German national computer societies GI and ITG defines an Organic Computing system as “a technical system which adapts dynamically to the current conditions of its environment. It is self-organizing, self-configuring, self-healing, self-protecting, self-explaining and context-aware”. While Autonomic Computing focuses mainly on servers and computing centers, Organic Computing aims at the development of robust, flexible and highly adaptive embedded systems.

To manage highly connective distributed systems, middleware is a well known and researched approach. State-of-the-art middleware is available for general distributed systems, embedded systems and real-time systems [7]. AC/OC Systems on Chip introduce new challenges for middleware: Future SoCs will contain more and more dynamically reconfigurable hardware [8, 9]. This allows to decide at runtime, if a certain functionality is realized by a software or a hardware module. The middleware will have to handle such distributed software and hardware modules, which have to be reconfigured or even migrated in real-time between the computation nodes. In general, middleware will become a key component to realize AC/OC principles in a highly distributed system.

Multithreaded processor architectures support the execution of multithreaded programs by special hardware attributes like multiple register sets, multiple program counters and special pipeline design to allow the mixed pipelined execution of instructions from different threads [10]. The multithreaded Komodo microcontroller [1] has been developed to explore the properties of hardware multithreading with hardware-integrated real-time scheduling for embedded real-time systems. Infineon incorporated multithreading into its Tricore-2 signal processor [11]. The multithread application-specific extension (MT-ASE) of MIPS proposes to use multithreading on the ASIC level [12]. The integration of a dynamically reconfigurable SoC and a multithreaded processor core has not been explored yet even so several advantages can be expected: latencies caused by the dynamic reconfiguration could be masked by the execution of

another thread, real-time scheduling could be done within the processor core, the energy consumption might be reduced and helper threads could support the implementation of the AC/OC principles.

Reconfigurable SoCs contain a processor core and a reconfigurable part, usually a FPGA. The reconfigurable part is used to adapt the SoC to a specific task. Reconfiguration can be done statically or dynamically. In static reconfiguration, the SoC is preconfigured for the given task and the configuration never changes during runtime. In dynamic reconfiguration, the SoC changes parts or the complete configuration during runtime. This is usually done to optimize performance by replacing software by hardware algorithms. Reconfiguration can be done on a fine-grained (gates) or coarse-grained (function blocks) level. The latter stretches from reconfigurable data paths between existing ALUs to reconfigurable functional units which reconfigure according to the current needs to become e.g. an integer unit, a floating point unit, etc. Combining coarse-grained dynamically reconfigurable SoCs with a multithreaded processor core and the AC/OC principles is a new promising approach which will be followed in the CARUSO project. A main research focus of the CARUSO project is not to have an isolated view on each requirement, but to explore and exploit the relationships between the requirements and attributes.

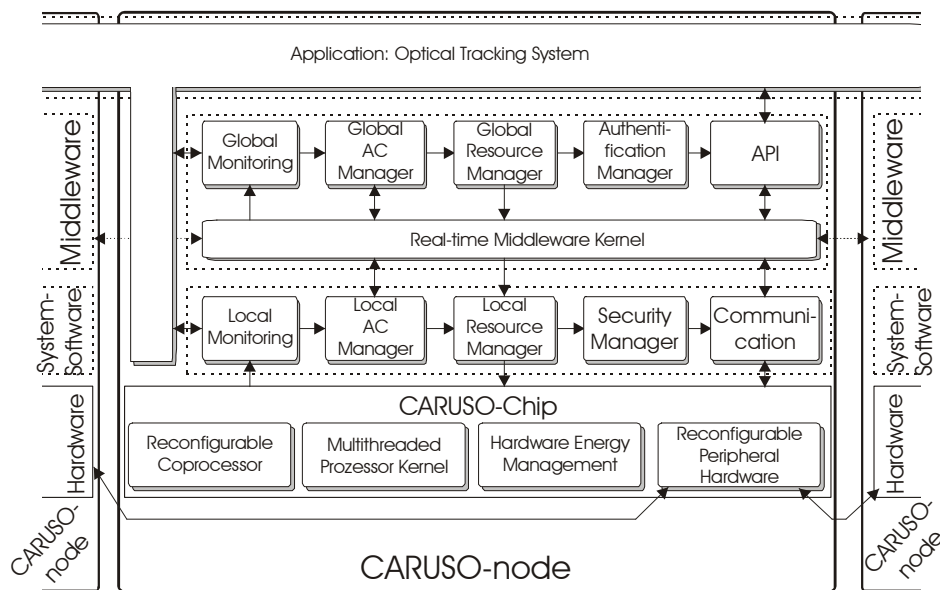


Figure 1: CARUSO architecture

3 CARUSO System Architecture

Figure 1 shows the proposed overall CARUSO system architecture. It is a distributed architecture consisting of hardware and software. The system basis is the CARUSO

chip, which combines a multithreaded processor core with reconfigurable hardware and power management. The processor core will implement the Tricore instruction set. The next level is the system software which is formed by helper threads. Autonomic/Organic management on the local (i.e. node) level is realized by a closed control loop consisting of local monitoring, local AC/OC management and local resource management. Changes detected by monitoring are reflected by the AC/OC manager and adapted by the resource manager.

A similar control loop can be found on the global (i.e. middleware) level. This control loop consisting of global monitoring, global AC/OC manager and global resource manager is responsible to handle the distributed AC/OC management, e.g. migration of services, deactivation of failing chips, etc. The middleware is therefore responsible for the global system management and optimization.

The application has been chosen as a typical example for the project aims: a mobile optical tracking system. Relative changes in camera pose can be determined by tracking marked features in the environment. Due to the fact that the CARUSO system should guarantee a certain accuracy requiring different time-consuming algorithms, the re-projection error [13] can be used as a measure for the reliability of the system. Thus, middleware is able to activate a more computational algorithm when critical situations occur. The application is dedicated to demonstrate and to evaluate the CARUSO system.

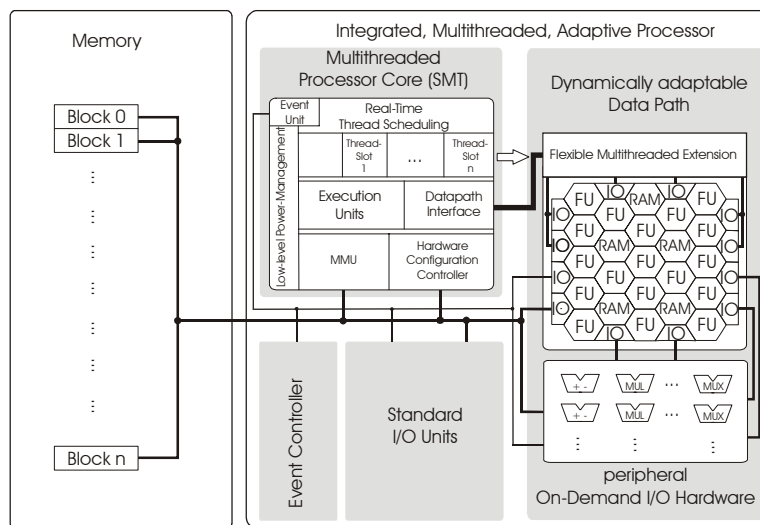


Fig. 2: The CARUSO SoC incl. adaptive multithreaded Data path

Figure 2 shows the structure of a single CARUSO SoC in more detail. It consists of a multithreaded processor core, memory to store programs, data, and configuration, a power management module, and dynamically reconfigurable modules that can be divided into two classes: The functional units (FUs) are responsible to perform tasks and services in hardware rather than in software and thereby gain performance or save energy. The configurable peripheral hardware forms a flexible interface which can be adapted to the current needs. The interface allows the connection to sensors, actors and other SoCs to build a dynamic network of SoCs.

4 Conclusions

We propose a new SoC approach called CARUSO that emphasizes connectivity, autonomic/organic computing principles, real-time, and ultra-low power requirements. The requirements shall be fulfilled by a multithreaded processor core within a reconfigurable SoC. The autonomic/organic managers are implemented as helper threads running with low priority in own thread slots concurrent to the application and decide if self-configuration, self-healing, self-optimization, or self-protection must be triggered.

The CARUSO project is in its starting phase. Theoretical and practical work has to be done to achieve the project goals. A prototypic implementation and a real-world application example are necessary to validate the expected results and to make them applicable for subsequent projects and products. This way, CARUSO aims to contribute to future embedded, ubiquitous and autonomous/organic computing systems.

References

- [1] J. Kreuzinger, A. Schulz, M. Pfeffer, T. Ungerer, U. Brinkschulte, C. Krakowski: "Real-time Scheduling on Multithreaded Processors", Real-Time Computing Systems and Applications (RTCSA), Cheju Island, South Korea, December 2000, 155-159.
- [2] S. Uhrig, Th. Ungerer: "Fine-Grained Power Management for Real-Time Embedded Processors", RTS Embedded Systems, Paris Expo, March 30 – April 1, 2004, 129-146.
- [3] P. Horn: "Autonomic Computing: IBM's Perspective on the State of Information Technology", IBM Manifesto, October 2001, <http://researchweb.watson.ibm.com/autonomic/manifesto>
- [4] J.O. Kephart, D.M. Chess: "The Vision of Autonomic Computing", IEEE Computer, Januar 2003, 41-50.
- [5] <http://www.organic-computing.org/>
- [6] [www.gi-ev.de/download/VDE-ITG-GI-Positionspapier Organic Computing.pdf](http://www.gi-ev.de/download/VDE-ITG-GI-Positionspapier%20Organic%20Computing.pdf)
- [7] U. Brinkschulte, A. Bechina, F. Picioroaga, E. Schneider: "Distributed Real-Time Computing for Microcontrollers - the OSA+ Approach", Int. Symp. on Object-Oriented Real-Time Distributed Computing (ISORC 2002), Washington D.C., April 29 - May 1, 2002.
- [8] K. Compton, S. Hauck: "Reconfigurable Computing: A Survey of Systems and Software", ACM Computing Surveys, 34, 2, June 2002, 171-210.
- [9] J. Becker: "Configurable Systems-on-Chip (CSoC)"; in: Proc. of 9th Proc. of XV Brazilian Symposium on Integrated Circuit Design (SBCCI 2002), Porto Alegre, Brazil, Sept. 5-9, 2002.
- [10] T. Ungerer, B. Robic, J. Silc: "A Survey of Processors with Explicit Multithreading", ACM Computing Surveys, 35, 1, March 2003, 29-63.
- [11] E. Norden: "A Multithreading Extension for Low-Power, Low-Cost Applications", Embedded Processor Forum, 2003, http://www.infineon.com/cmc_upload/documents/082/795/TC2_MT_EPF03.pdf
- [12] R. Wilson: MPF: "MIPS Adds Multithreading to CPU for Net Apps", iApplianceWeb 10/14/2003.
- [13] R. Hartley and A. Zisserman. "Multiple View Geometry in Computer Vision", Cambridge University Press, Cambridge, 2000.