The Use of Runtime Configuration Capabilities for Network Embedded Systems

Carsten Nitsch, Udo Kebschull
University of Leipzig
carsten@ti-leipzig.de, kebschull@ti-leipzig.de

Abstract

Reconfiguration is a very helpful feature that can improve the design life cycle of an embedded system and its quality. Reconfiguration means that software AND hardware parts may be updated in the field. The update of system hardware implies the use of FPGAs in a shipped system. Normally, the update is done server controlled, which means that the active role comes from an external instance. We present a new automatic reconfiguration approach that stores all system configuration data in XML format. The system itself searches for the related components a component broker, and sets up during start up. A case study shows that especially when dealing with permanently connected devices, we achieve promising results while spending a reasonable price.

Concept of a RCC Embedded System[1]

The system consists of three basic parts: the hardware pool, the system management software (firmware) and code describing the user application. The system architecture provides a set of hardware facilities, like microprocessors, DSPs and FPGAs. These hardware parts are connected by a bus or another communication interface. A network interface connects the system to the internet. The basic idea of our concept is the *strict separation* of design and implementation of an application. Application designers can focus to a global view of the application and their functional aspects. The application designer defines *functional units* and describes the interaction between these parts. The implementation is the job of code writers. There can be a set of different pieces of software (*plug-ins*) implementing the functionality of a special component. The choice for a special plug-in (*mapping process*) depends on statical and dynamical constraints, like microprocessor architecture or current system load. We use XML to describe the functional structure and behavior of the application. The goal of our design study is a system, which analyzes the XML specification and locates all required plug-ins autonomously. This requires extensive use of directory and naming services etc. Fortunately, there are a variety of technologies offering these services. We decided to use the Enterprise Java Beans[2] (EJB) technology. If a client wants to execute some kind of application, the XML specification has to be analyzed first. The next step is the *mapping process*. The system has to find and access the code, which implements the functional components. These pieces of code will be loaded onto the target. The methods of Enterprise Java Beans implement this mapping functionality. The output are objects containing executable code, which can run on a microprocessor or will be used to configure a FPGA.

[1]Runtime Configuration Capable Embedded System

[2]Enterprise Java Beans is a trademark of Sun Microsystems

Results

Mapping process: Delays during the mapping process are caused by component queries, transmission of component code via the internet, etc. Due to different network architectures, dynamical change of available bandwidth, performance and system-load of the EJB Server etc, it is not easy to determine these delays. To handle real-time constraints, the target has to load code components and to cache them locally.

Configuration: The configuration process is controlled by a microprocessor. Configuration management code running under a real-time operating system has to use system calls to transmit and receive data. Without any additional hardware support like DMA transfer, there is a significant overhead. Due to these limitations, reconfiguration by software is slow, so „just in time“ reconfiguration is not possible in many cases. On the other hand, software based reconfiguration is easy to implement. When reconfiguration is not needed frequently, the software based reconfiguration approach is a good choice.

Conclusions

RCC is even available by today, presumed that our configurability features are included to such a system. Chip or system vendors simply need to include the infrastructure to their SoC that supports network access including the necessary protocols. Furthermore, XML is well suited to act as a system description memory. Our component based system synthesis technique supports modern system design methodologies since it distinguishes between IP and system design. It therefore shortens the time to market dramatically.