MICRO CONSULT

11/2015

Software Design Challenges for Real-Time Multicore Microcontroller Systems

Authors: Thomas Batt and Ingo Pohle

Complex applications like controls for automotive and industrial systems or the Internet of Things (IoT) in combination with functional safety requirements (e.g. according to ISO 26262) and security require specific hardware support for safety and security features as well as higher calculation power in the control units.

The application software is more complex. It has to execute more instructions but, at the same time, should not cause higher power consumption.

Systems like notebooks, tablets, smartphones etc. try to meet these requirements by implementing multicore CPUs that operate in parallel; however, they do not offer higher operation frequency. To benefit from this technological approach, automotive software designers requested microcontroller chip manufacturers to develop multicore microcontrollers that include additional on-chip support for safety and security functions. This add-on functionality shall meet all requirements for system control units that combine high calculation power, safety & security support and powerful peripherals for real-time applications.

Existing embedded software designed for execution on a single core now has to be adapted for execution on multicore systems. Initially, software migration seems to be easy and done. However, software developers have to consider numerous tricky aspects like function/task assignment to specific cores and interrupt controllers, data assignment to core private or global memory as well as the synchronization of data resp. results of multicore software execution.

New applications designed for portable systems like PCs, notebooks, phones etc. are typically processed in symmetric multicore microprocessors where the application software can be executed on any of the available cores. The software can be virtualized to tasks and assigned by a hypervisor, at runtime, to one of the CPUs that is free resp. can be allocated.

Real-time application software has to be extended and executed on a system according to functional safety (e.g. at one of the defined safety levels, SIL 1 - SIL 4 or ASIL-A - ASIL-D). An additional security option may be required if the application requires internet access or software tuning protection.

Depending on the real-time, safety and security requirements, highly sophisticated multicore microcontroller architectures have to be used. In contrast to multicore microprocessors (CPUs) used for entertainment and communication systems, these aspects must be considered for machine control units that may cause a machine to operate improperly, thus resulting in physical injury or direct or indirect harm to the health of people.

Another problem might arise when industrial systems operate in connection to the internet. While this type of application largely facilitates operation and provides many new options, e.g. remote control, the system must be protected against any unexpected result (start, stop, change operation, etc.) or destruction. For example, a seventeen year old hacker stopped the operation of a blast furnace in Germany, causing damage to the system and loss of several hundred million euros. New requirements call for new software solutions.

Singlecore software solutions for automotive or industrial control systems have to be redesigned so that all real-time & safety & security requirements are met. For this reason, a new generation of multicore microcontrollers was built with asymmetric core architecture. It offers on-chip hardware in a “low-cost” microcontroller design for all of these requirements.

With this approach, it is not really possible to use technologies like software virtualization or distributing systems like a hypervisor in this kind of heterogeneous microcontroller architecture.

AUTOSAR - a Software Standardization Approach for SW Reusability and Multiple SW Vendor Concept

The automotive sector was an important driving factor for new designs of powerful multicore microcontroller architectures. Automotive manufacturers (OEMs), component suppliers (Tier 1) as well as chip and software tool producers have defined the AUTOSAR standard (AUTomotive Open System ARchitecture) to reduce the effort for resp. improve the implementation of distributed systems in automotive applications.

AUTOSAR was basically developed for singlecore architectures. Some extensions have been implemented for multicore, including multicore operation for software processing. The AUTOSAR architecture model defines a Virtual Function Bus (VFB) for task communication. One new aspect was the extension of the communication layer (AUTOSAR includes the service Run-Time Environment RTE for the VFB) defined for intra-core task communication to intra-/inter-core communication (for core-to-core synchronized data exchange).

For the application described above, the RTE layer is completely ECU independent. The RTE transfer interface can be used without the knowledge of the existing microcontroller hardware. This approach is very smart for the development of hardware independent and reusable software. However, if you don’t know on which core your software task is executed and inter-task communication may require interrupt-driven data exchange, the result is probably not really efficient and predictable as regards timing. Hard real-time applications will not be able to guarantee very fast and time-critical processing. For safety relevant applications, the program flow for inter-core communication has to be supervised by a specific program flow service, e.g. to avoid data delivery timeouts.

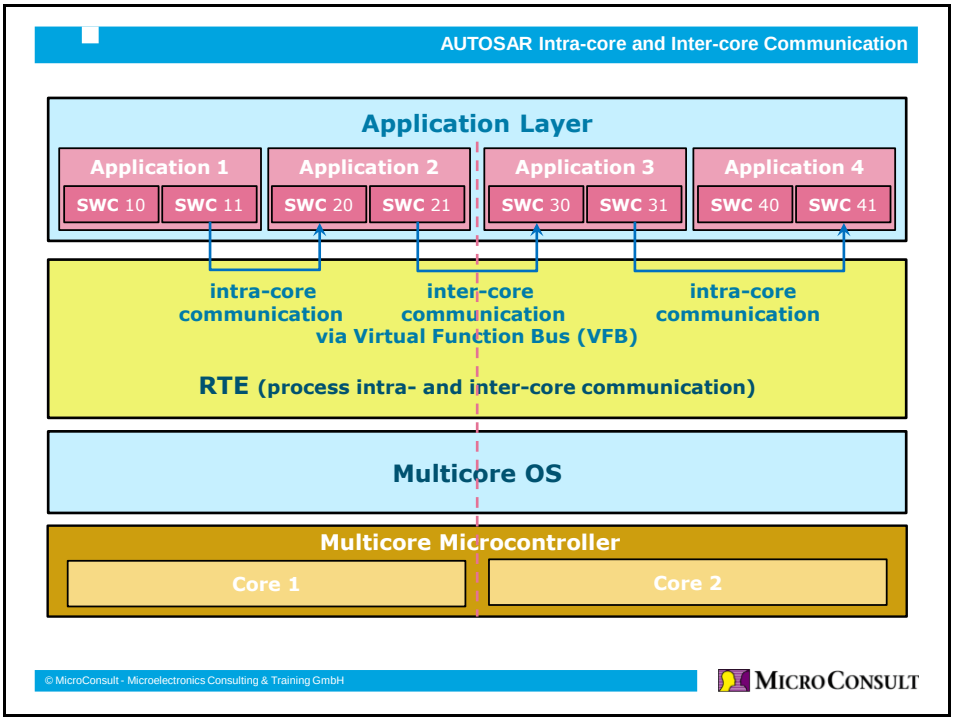

Image: AUTOSAR intra-core and inter-core communication

Software Safety and Security

Another aspect is the requirement for safety software processing that has to operate at a functional safety layer (like SIL or ASIL). Depending on the risk level, software tasks for safety applications have to be supervised through different methods, like:

- Error correction codes (ECC) for supervising and correcting memory contents and inter-chip data communication (ECC detect and correct errors stored in memory, e.g. single bit errors are corrected and multiple uncorrectable errors are monitored)
- Guaranteed error responses solved by a safety management unit SMU or fault collection and control unit (FCCU) - all detected failures resp. errors are monitored and reported to this system, resulting in a user-defined response, such as error function, reset or signaling error via pin to an external environment.
- A memory protection unit (MPU) can be programmed to allow access to preselected memory space for execute, read, write, or read + write. Access to other areas without access enable results in an error signaling.

Impact of the Memory Architecture on the Multicore Architecture Software Design

Two different types of memory architectures for code and data memory are available for the new multicore microcontrollers:

Multicore memory architecture type 1:
- less CPU-local (private) memory: level 1 – high performance access
- much global system (shared) memory: level 2 – lower performance competitive access

Multicore memory architecture type 2:
- much CPU-local (private) memory: level 1 – high performance access
- less global system (shared) memory: level 2 – lower performance competitive access

The speed of CPU access to on-chip memory involves the response times. To boost software performance, CPU private caches and SRAMs are available to improve access times for code and data. The processing time of interrupt service routines (ISRs) may be one of the most time critical aspects of embedded real-time software. If the program and variables of an interrupt service are accessible without wait states, the ISRs can be performed very fast. Thus, the nesting of interrupt processing means shorter accumulated access delay for lower priority interrupt services.

What is tricky about multicore architectures is that each CPU has its own interrupt and trap controller. Consequently, software designers have to assign the interrupt sources, like peripherals or error events, to a specific CPU-private interrupt vector table. The assignment has to be performed based on the timing requirements of the specific interrupt source.

Memory Assignment in Multicore Architectures

In multicore architectures, real-time and memory protection requirements may be the basis for fixed memory assignment in order to guarantee (worst-case) process timing and correct programming of the memory protection unit (MPU). The software architect assigns code and data to specific memory spaces. Time critical memory can be placed in core private memories, and non time critical information may be placed in global memory.

This design step has an impact on the coding of the source code, because customized section names for memory sections are needed. The user-defined section names can be generated by compiler controls in the C-source files for code (program functions) and data (variables and constants). The project build process includes a compilation of all source files and linking plus locating the resulting memory sections to the available memory spaces. The section base addresses in the memory and memory ranges can be user-defined in the project linker description file. The resulting link/locate protocol is stored in a map file that may be the basis for controlling the planned memory architecture.

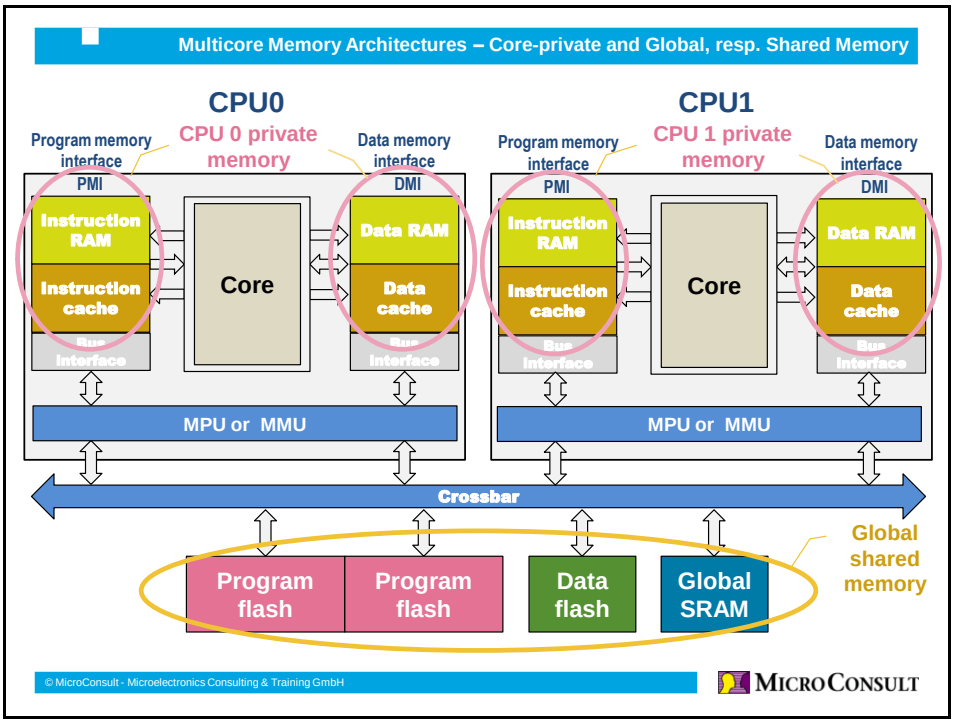

Image: Memory implementation in multicore

Using a Multicore Operating System

Another aspect regarding multicore microcontroller system architectures is the use of a real-time operating system (RTOS). This multicore RTOS has to be adapted to the specific microcontroller architecture. Operating systems mainly differ regarding available OS services, interrupt blocking times and memory protection system support (for memory access detection and protection). Tasks can be allocated to specific cores, and resources have to be used with coordinated data access between tasks and cores. In case the system shall have multiple different operating systems, e.g. a native RTOS and a Linux OS, support might be required for inter-OS data exchange.

The software architecture has to be adapted to the OS task system which may require OS software priority to meet the real-time requirements of the different software tasks. Application design is different in case one OS is used for several resp. all cores compared to one separate OS being implemented for each core.

Operating system use is based on a software architecture design that considers timing aspects (tolerable software delays, timeouts, etc.) and includes information on whether software can be processed in parallel (if the function can be operated in parallel on different CPUs), resource usage (e.g. size of required memory space), data exchange between software modules, function repletion times, etc. Depending on these aspects, the software design can be used to refine and optimize the software architecture.

Multicore Debugging

The debugger system requires extension services for multicore architectures, e.g. synchronous core start and stop of the available CPUs, core-individual breakpoint handling and multicore trace support. Trace support is the basis for resource and performance measurements. Measurement results of the trace protocol can be used for timing analysis. It can be used to check the timing aspects of the existing software architecture and serve for software optimization, such as improved and balanced load of the different CPUs.

Additional features may be debugger support for white box tests, test sequence automation and test reporting. If the multicore microcontroller includes dedicated security hardware, specific debugger access to these protected memory areas may be required.

Concluding a Migration Process from Singlecore to Multicore Microcontroller Software

The development effort for multicore microcontroller software development mainly depends on whether the multicore architecture is of homogeneous or heterogeneous design. It moreover depends on whether more CPU-local or more global memory is available for processing the application code. The need for synchronization of project data may influence the complexity, performance and testability of the system. Supervised and protected memory access may be the basis for project success or failure.

A software design for multicore microcontrollers requires strict system development including an elaborated requirements analysis, a detailed software architecture plan, profound knowledge of multicore microcontroller architectures as well as software test experience.

Authors:

Dipl.-Ing. (FH) **Thomas Batt** has been Trainer and Coach for embedded systems engineering and process consulting at MicroConsult for more than ten years. His main focus has been on establishing and managing the areas embedded systems/software engineering and management as well as project, process and corporate coaching. Before starting his career at MicroConsult, he was an engineer for power electronics, medical electronics, CompactPCI/VME bus and peripheral boards.

Dipl.-Ing. **Ingo Pohle** is co-founder and Managing Director of MicroConsult. He is an internationally renowned specialist for embedded solutions, with a wide range of experience in the field of microcontrollers, bus systems and RTOS.