Summary and Trends

Ingo Sander
ingo@imit.kth.se

Embedded Systems

- Different Requirements are imposed on Embedded Systems, such as
  - Costs
  - Time-To-Market
  - Performance
  - Power Efficiency
  - Safety
- Many Embedded Systems have to interact with several systems at the same time
  - Can be modeled as parallel tasks
  - Can be implemented on a parallel architecture

Specification of Embedded Systems

- Contrary Requirements
  - Abstract specifications lead to a shorter design time (time-to-market)
  - Details are needed for an efficient implementation (performance, power)
- Ideal Design Process
  - Model the system with an abstract language and compile it to an efficient implementation!

Specification of Embedded Systems

Idea (Specification)

Product (Implementation)

Can a Compiler perform the ”magic” we look for?

Compiler

- Embedded System Architectures are so complicated that compilers cannot generate optimal solutions!
- Even in a single-processor environment the performance are hard to predict due to e.g.
  - Caches
  - Pipeline
- For an optimal solution understanding of the hardware and fine-tuning in Assembler is still needed!
- For multi-processor systems, the situation is naturally much worse!

What to do?

- Possible strategy
  1. Design your system in a high-level language, where a ”good” compiler exists
  2. Check, if the implementation meets the design requirements
  3. If not, improve the critical parts of your design by fine-tuning it in assembler and using the underlying features of the hardware
  4. If all that fails, think of another architecture
     - Change cache size
     - Change processor
     - Investigate a parallel solution

Polling vs Interrupt

- Polling and Interrupt
  - In polling, the processor has to wait for a resource to get ready and cannot do anything useful while waiting
  - Interrupt allows the processor to work with other things, but includes an overhead

Caches

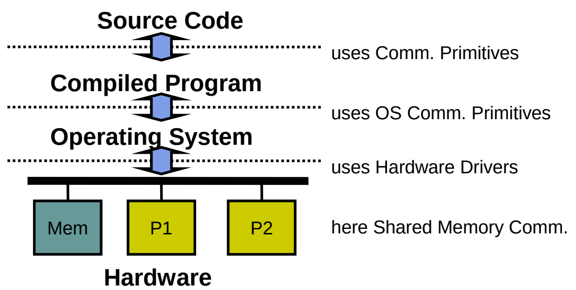
- Caches
  - Caches shorten the memory access time and lead to much better performance
  - Caches introduce uncertainties, which make it difficult to use all benefits for hard real-time systems, where the requirements have even to be fulfilled in the worst case situation

Real-Time Operating Systems

- Many embedded systems fit in a parallel process model
- Real-Time Operating Systems help to design real-time systems which are described as parallel tasks
- Interrupts initiate the scheduling of a new task
- For periodic tasks scheduling methods with static (RMS) and dynamic (EDF) priority exist
- The RTOS has to work for the worst case. Thus the effects of caches cannot be used to their full advantage

Abstraction through Layered Software Model

- A layered model allows to abstract from the implementation detail of the underlying levels
- The operating system defines services and hides the hardware
- Thus the source code does not need to reflect the architecture at all
- Challenge: To develop tools at several levels to translate the source into an implementation on the underlying architecture



Future Work

- May result in
  - Better compilers that allow to design at a higher level
  - Better operating systems that can use caches efficiently
  - Power-efficient high-performance processors at a low cost which can be used for all design problems...
- ... But until this happens
  - Know your compiler well
  - Know your architecture well
  - Reflect over architecture alternatives
  - Design your systems in a methodic way

A look on the next course...

- Embedded Systems (2B1445) has looked at
  - The microprocessor as a component for embedded system design
- SoC Architectures will focus on the communication problem in heterogeneous parallel architectures
  - Bus Architectures with Shared Memory
  - Hardware Acceleration
  - Network on Chip Architectures
  - The component Digital Signal Processor