The ACROSS MPSoC

A New Generation of Multi-Core Processors Designed for Safety-Critical Embedded Systems

Dr. Christian EL SALLOUM
TU Wien
AVL List GmbH
Outline

• ACROSS – project overview
• Motivation to use multi-cores in safety-critical embedded systems
• Problems of existing multi-core architectures
• The ACROSS approach
• Prototype implementation and demonstrators
ACROSS – Objectives

• Develop a cross-domain architecture for embedded systems
• Cover automotive, aerospace & industrial control systems with a single concept
• Project Facts
  • Budget: € 15.998.978
  • 16 partners
ACROSS – Consortium
Why consider multi-cores for dependable embedded systems?

• Scalable with respect to computational power
  – Single-core performance is reaching its limits

• Energy efficiency
  – Multiple cores running on lower frequency consume less power than single high-speed cores

• Heterogeneous cores can be tailored to specific functionalities of an embedded system
  – e.g., encoders and decoders, security cores, RF components, high frequency controllers ...
Problems with existing multi-core architectures
(1) High Complexity

- In the guidelines of the European Aviation Safety Agency (EASA) and the Federal Aviation Administration (FAA), existing multi-core processors are classified as "highly complex microcontrollers"
- This classification has a significant impact on certification issues
- A fallback solution is to simply turn off all other cores and use the multi-core as a single core
  - This is not what one really wants 😞
Problems with existing multi-core architectures

(2) Missing temporal determinism

• Many safety-critical embedded applications are classified as hard real-time
  – Missing a specified deadline can have catastrophic consequences (e.g., drive-by-wire, airbag, avionics ...)
  – Systems have to guarantee worst-case execution times

• Existing multi-core architectures are optimized for maximal average performance (i.e. maximal throughput)
  – Employed mechanism for performance increase make the calculation of worst-case behavior infeasible
  – Abstraction layers hide away details that are relevant for the temporal behavior (e.g., complex implicitly loaded cache architectures, shared memory abstraction on top of physically distributed memory ...)

Dr. Christian El Salloum 21.01.2013
Problems with existing multi-core architectures

(3) Lack of adequate partitioning mechanisms

- IEC61508: “An E/E/PE safety-related system will usually implement more than one safety function. If the safety integrity requirements for these safety functions differ, unless there is sufficient independence of implementation between them, the requirements applicable to the highest relevant safety integrity level shall apply to the entire E/E/PE safety-related system.”

- Typical existing multi-core architectures employ mechanism that compromise strong spatial and temporal partitioning
  - E.g., shared resources without rigorous access control like shared caches or shared I/O, power management techniques
Objective of the ACROSS MPSoC

• Provide a heterogeneous multi-core architecture that enables certification for the highest criticality classes
  – $10^{-9}$ failures/hour $\approx$ probability of 1 failure / 100 000 years
  – We are talking about system failures not component failures!
• Enable construction of systems with temporally deterministic behavior
  – Dependable prediction of WCET
• Enable mixed-criticality integration
  – Prevent error propagation between subsystems
  – Robust partitioning in time and space
The ACROSS Approach

• Increase in the level of abstraction
  – We regard the MPSoC as a multicomputer

• Cores are highly autonomous components
  – Cores have sufficient local memory for the program code
  – Have an explicitly defined message-based linking interface (LIF)
  – Exchange application-level messages (e.g. a speed value of car)

• Communication among cores is restricted to the LIFs
  – No other internal dependencies or sources of interference (e.g., no shared memory)

• Deterministic interconnect
  – Providing temporal and spatial isolation
The ACROSS MPSoC

- Trusted Resource Manager (TRM)
- Gateway System Component Host IP Core
- Video System Component Host IP Core
- Storage System Component Host IP Core
- Local I/O Application Component Host IP Core
- Application Component Host IP Core
- System Component Local I/O
- Application Component Local I/O

- Time-Triggered Network-on-Chip
Time-Triggered Network-on-Chip

• Each core in the MPSoC has access to a consistent chip-wide notion of time (i.e. macro tick)
• A priory defined communication schedule including for each message
  – the sender of the message
  – the set of receivers (multi-cast is supported)
  – the send instance with reference to the macro tick
• The time-triggered schedule is free of conflicts
  – No on-line arbitration required
  – Temporal behavior is “designed”, and not wearisomely analyzed after the implementation
Temporal and spatial partitioning

- The **Trusted Interface Subsystems** (TISSES) act as a guardian for the NoC.
- Host can send only at their allocated time slots $\rightarrow$ Faulty hosts cannot disrupt the communication among other hosts.
- The time-triggered schedule can be exclusively changed by the **Trusted Resource Manager** (TRM).
- The TRM is optional and can be omitted for purely static designs.
Fault Containment Regions in ACROSS

• “A fault containment region (FCR) can be defined as: a set of components that is considered to fail (a) as an atomic unit, and (b) in a statistically independent way with respect to other such FCRs“ [Kopetz]

• In ACROSS we distinguish between two fundamentally different viewpoints on an FCR according to the class of faults
  – Systematic design faults
  – Random hardware faults
Systematic design faults

- Design faults in the hardware or software of a given core
  - e.g., Coding errors in application software or in VHDL
- For these faults, the architecture assures by design that each core constitutes an independent FCR
- The basic assumption is that the TTNoC, the TISSes, and the (optional) TRM is free of design faults
  - These components have the same certification requirements as the most critical application on the MPSoC!
Random hardware faults

• Faults that occur during system operation or the manufacturing process
  – e.g., Single Event Upsets (SEUs) due to electromagnetic interference or radiation or aging

• Common mode failures cannot be avoided on a single die
  – e.g., shared power supply, clock, package, spatial proximity

• For ultra-dependable systems the entire chip has to be considered as a single FCR \(\rightarrow\) HW faults have to be tolerated by hardware redundancy at system level
  – e.g. *Triple Modular Redundancy* (TMR)
Example – TMR in a Traditional Design

- 10 Subsystems
- ECRs with Triple Modular Redundancy (TMR)
- 30 nodes in total required
Example – TMR with the ACROSS MPSoC

- 10 Subsystems
- ECRs with Triple Modular Redundancy (TMR)
- Subsystems share nodes
- Nodes are replicated
- 3 nodes in total required
FPGA-based Prototype

Implemented on ALTERA Stratix IV GX Development Kit

21.01.2013
Dr. Christian El Salloum
ACROSS – Demonstrators

- Automotive
- Avionics
- Industrial Control
Conclusion

• The ACROSS MPSoC is a platform targeted for certifiable safety-critical embedded applications
• Supports temporally deterministic applications via a time-triggered NoC
• Facilitates modular certification by preventing non-intended interference between subsystems
• Supports the integration of subsystems with mixed-criticality levels by reliable partitioning mechanisms
Contact

ACROSS is coordinated by TU Vienna:
Technical Coordinator:
Dr. Christian EL-SALLOUM
phone: +43-1-58801-18225
email: christian.el-salloum@tuwien.ac.at

Administrative Manager:
Sibylle KUSTER, MBA
phone: +43-1-58801-18222
email: kuster @ vmars.tuwien.ac.at

www.across-project.eu
Thank you for your attention!

Any questions?