Towards a Modelling and Design Framework for Mixed-Criticality SoCs and Systems-of-Systems

F. Herrera, H. Attarzadeh and I. Sander

KTH Royal Institute of Technology

DSD'13, Santander (Spain), Sept. 26th, 2013
Introduction

Design Disciplines related to MCS design

Proposed core MCS ontology

Open aspects and features for MCS design

Conclusions
Mixed-Criticality Systems (MCSs)

- **Integrated suite of hardware, operating system and middleware services and application software that supports the execution of safety-critical, mission-critical, and non-critical software within a single, secure compute platform** [Barhorst et al., 2009]
- **Core foundational concept in Cyber-Physical Systems** [Baruah et al., 2010]
Mixed Criticality Applications

**Mixed-Criticality Application**

- **FCS** (Safety-Critical) and **Best-Effort**

**Diagram:**
- Criticality axes: Safety-Critical (Mixed) vs. Best-Effort (Single)
- Scale axes: Embedded vs. Distributed

**Text:**

F. Herrera, H. Attarzadeh and I. Sander

Towards a Modelling and Design Framework for Mixed-Criticality Systems
Mixed Criticality Systems

Mixed-Criticality Application

Safety-Critical (FCS)  Best-Effort

Predictability - Efficiency trade-off

Platform

Shared Resources:
- computation
- communication
- memory

Towards a Modelling and Design Framework for Mixed-Criticality Systems
Mixed Criticality System Scales

Mixed-Criticality Application

- Safety-Critical
- Best-Effort

Criticality

- Mixed
- Single

FCS

Predictability - Efficiency trade-off

Platform

Shared Resources:
- computation
- communication
- memory

Towards a Modelling and Design Framework for Mixed-Criticality Systems
Mixed-Criticality Application

FCS
Safety-Critical
Best-Effort

Criticality

Predictability - Efficiency trade-off

Platform

Shared Resources:
- computation
- communication
- memory

F. Herrera, H. Attarzadeh and I. Sander
Towards a Modelling and Design Framework for Mixed-Criticality Systems and Systems-of-Systems
Towards a Wider Approach

Mixed-Criticality Application

FCS
Safety-Critical
Best-Effort

Criticality

Mixed

MCSoC

MCSoS

Single

NoC
HRT
EDS

Predictability - Efficiency trade-off

Platform

Shared Resources:
- computation
- communication
- memory

F.Herrera, H. Attarzadeh and I. Sander
Towards a Modelling and Design Framework for Mixed-Criticality SoCs and Systems-of-Systems
For Multi-Processors: from [Liu & Lailand, 73] to [David & Burns, 11]

Consideration of impact of communication resources of the:
- NoC [Shi & Burns, 10][Pellizoni et al., RTSSS’09]
- Embedded Distributed network, e.g. ECU networks [Rajeev et al, 10], MAST2 [Harbour et al. & Burns, 12]
Extension of Hard Real Time theory for MCS

- Priority-based, Reservation-based (→ Criticality inversion)
- New scheduling theory [Baruah et al, 2010], e.g. OCPB
  - criticality ≠ priority
  - workloads depend on criticality, i.e. \( WCET = f(\chi) \ | \ \chi \in \mathbb{N}^+ \)
  - scheduling algorithms: OCPB, CAPA
- Standard IEC 61508, SIL
Other disciplines which should be involved

- **Model-driven technologies, MDE, MDA** (OMG)
  - Metamodel, Graphical and standard front-end
  - M2T (Correctness-by-construction in SW development), M2M
  - Views (Separation-of-Concerns)

- **System-Level modelling/specification**
  - Abstraction, Concurrency, Heterogeneity
  - Models of Computation
  - Modelling constraints for Properties (Determinism, Deadlock protection, Boundeness, etc)
  - Ptolemy II, Metropolis II, ForSyDe, HetSC, SysteMoC, SystemC-H, ...
Other disciplines which should be involved

- **Design Space Exploration**
  - Analytical Techniques
  - Simulation-based Techniques
  - *Joint analytical and simulation-based (JAS) techniques*

- Simulation-based Performance Estimation
  - ISS, cycle-accurate ISS, RTL simulator
  - Virtualization
  - *Native Simulation*
Communication modelling and analysis

- Variety of taxonomies:
  - NoC vs Distributed system
  - Switched vs Packetized
- Variety of standards, domains and architectures
  - Standards: WiDom, CAN, Spacewire, Flexray, TTEthernet, AFDX, etc
- Predictable networks: Main parameter: WCCL
- Other properties: Scalability, Segregation
- Variety of tools
  - NoC simulators: TOPAZ, Nostrum, Noxim
  - DE: MAST2, OMNET, etc.
A bunch of integrating work already done!

- MoC theory and DSE
- MoC and NoC
- MDA and MoC
- MDA and DSE
- ...
Core Ontology: SoC
Core Ontology: SoC/SoS
Core Ontology: SoC/SoS (Environment)
Core Ontology: MCSoC/SoS
Multi-level approach

For computation nodes
- abstract level: traffic generators, and statistic collectors
- detailed level: SoC model

For the interconnection
- abstract-level: matrix of P2P links with specific attributes
- detailed-level: specific NoC and network models
Open Aspects and Features

- Agreed core terminology for modelling elements, e.g. . . .
  - task = a system-level concurrent behaviour? . . .
Open Aspects and Features

- Agreed core terminology for modelling elements, e.g. . . .
  - *task* = a system-level concurrent behaviour? . . .
  - . . . or a software-level concurrent behaviour (thread/process)?
Open Aspects and Features

- Agreed core terminology for modelling elements, e.g. . . .
  - task = a system-level concurrent behaviour? . . .
  - . . . or a software-level concurrent behaviour (thread/process)?
- . . . only for modelling elements?
Open Aspects and Features

- Agreed core terminology for modelling elements, e.g. ...
  - task = a system-level concurrent behaviour? ...
  - ... or a software-level concurrent behaviour (thread/process)?
- ... only for modelling elements?
  - actor **mapping** and actor **scheduling** [Kumar et al., 12]
Open Aspects and Features

- Agreed core terminology for modelling elements, e.g. . . .
  - *task* = a system-level concurrent behaviour? . . .
  - . . . or a software-level concurrent behaviour (thread/process)?
- . . . only for modelling elements?
  - actor **mapping** and actor **scheduling** [Kumar et al., 12]
  - Multiprocessor **scheduling** algorithm $\rightarrow$ scheduling = allocation + job ordering [David and Burns, 11]
Open Aspects and Features

Composability of models and techniques

- specifically, hard-real time models and MoCs
- combination of constraints and assumptions → properties-by construction and analizability
Multi-level (platform-model) approach

Seamless swap of computation nodes and networks

- at different levels of abstraction (enables gradual refinement and segregation of analysis)
- of different types of physical platforms without having to change sw-level platforms (reuse of RTOS modelling engines, facilitates automated exploration)
Enhanced DSE techniques

- criticality-aware exploration (and optimization)
- Combined static (analytical) and dynamic (simulation-based) DSE techniques
- Combine time constraints e.g., throughput and deadlines

[Herrera&Sander, FDL’13]
Enhanced DSE techniques

- criticality-aware exploration (and optimization)
- Combined static (analytical) and dynamic (simulation-based) DSE techniques
- Combine time constraints e.g., throughput and deadlines
- DSE enabling exploration of scheduling policies plus computation and communication infrastructures

[Herrera&Sander, FDL'13]
Integration of MDA/MDE techniques I

- MCSoC/MCSoS metamodel
  ← MCS ontology

- standard and graphical front-end,
Integration of MDA/MDE techniques I

- MCSoC/MCSoS metamodel ← MCS ontology
- standard and graphical front-end,
- integration and automatic generation of executable models...
Integration of MDA/MDE techniques I

- MCSoC/MCSoS metamodel
  - MCS ontology
- standard and graphical front-end,
- integration and automatic generation of executable models...
- ...for SoC (e.g., COMPLEX),
Integration of MDA/MDE techniques I

- MCSoC/MCSoS metamodel ← MCS ontology
- standard and graphical front-end,
- integration and automatic generation of executable models...
- ...for SoC (e.g., COMPLEX),
- ...for MCSoC/SoS (e.g., [Ebeid et al., UKSim2013])
Integration of MDA/MDE techniques: Separation of Concerns

Views (for independent and concurrent development)
- at SoC-level (e.g., UML/MARTE COMPLEX)
Integration of MDA/MDE techniques: Separation of Concerns

Views (for independent and concurrent development)

- at SoC-level (e.g., UML/MARTE COMPLEX)
- at NoC/SoS-level, communication centric,
Integration of MDA/MDE techniques: Separation of Concerns

Views (for independent and concurrent development)

- at SoC-level (e.g., UML/MARTE COMPLEX)
- at NoC/SoS-level, communication centric,
- analysis-based (MoC vs HRT)
Integration of MDA/MDE techniques: Separation of Concerns

Views (for independent and concurrent development)

- at SoC-level (e.g., UML/MARTE COMPLEX)
- at NoC/SoS-level, communication centric,
- analysis-based (MoC vs HRT)
- Criticality-aware perspectives
Integration of MDA/MDE techniques: Separation of Concerns

Views (for independent and concurrent development)

- at SoC-level (e.g., UML/MARTE COMPLEX)
- at NoC/SoS-level, communication centric,
- analysis-based (MoC vs HRT)
- Criticality-aware perspectives
- and how to combine them?
Open Aspects and Features

- Tunable platform in terms of resources for predictability and for average-optimization
- Techniques for fast assessment of platform requirements in terms of the aforementioned resources (criticality profile)
- Criticality regarding performance metrics and properties
Conclusions

- Mixed Criticality (MC)
  - a logic consequence of complexity and efficiency,
  - present at different system scales

- MC System (MCS) Design requires:
  - A broader, more interdisciplinary, perspective
  - an important integration effort of existing methodologies
  - developing novel aspects and features

- This paper:
  - has provided a view of the main disciplines to be integrated,
  - proposed a core ontology for MCSoC and MCSoS design,
  - identified novel aspects and features for MCS design regarding MDA integration, criticality-aware DSE, etc
Thanks to:

» the reviewers,
» attendees,
» KTH/ICT Excellence Postdoc position grant I-2011-0646
» We are willing to collaborate!

Contact:
» fernanhc@kth.se
» ingo@kth.se