Formal analysis and optimization of heterogeneous networks in industrial practice
- from networked systems to MpSoC

Marek Jersak, Symtavision

Rolf Ernst, TU Braunschweig

Overview

• Part 1:
  Formal analysis in industrial practice
  Marek Jersak, Symtavision

• Part 2:
  From networked systems to MpSoC
  Rolf Ernst, TU Braunschweig
Formal analysis and optimization of heterogeneous networks in industrial practice

Marek Jersak, Symtavision

Integration Challenges

**Complexity**
- Hundreds of functions, many safety-critical
- 50+ ECUs
- Networked
- Many suppliers

**Integration challenges**
- Reliability, quality, liability
- Meeting SOP target
- Development and production cost
Integration Challenges: 5 buses, 55 ECUs

Typical Automotive Architecture

- Bus protocols (and arbitration)
  - CAN – static-priority non-preemptive
  - FlexRay – TDMA + SPNP
  - Lin
  - MOST
  - Proprietary

- ECU (electronic control unit)
  - OSEK RTOS (different flavours exist) – static priority preemptive

End-to-end timing is important
Many functional problems are in fact *timing* problems

- ECU (temporarily) overloaded
- tasks not always schedulable
- deadlines are missed
- network (temporarily) overloaded
- messages arrive "too late" or with "too large" jitter
- messages are lost (buffer overflow)
- end-to-end deadlines of car function are missed
- stability of distributed control is compromised

➔ Carefully monitor performance and timing during design and integration

**SymTA/S Tool Suite:**
Scheduling Analysis and Optimization

![Diagram of SymTA/S Tool Suite](image)

**Plug-Ins**
- Exploration
- Sensitivity
- ...

**Component libraries**
- OSEK variants
- AUTOSAR OS
- CAN
- FlexRay*
- Gateway*

**Open interfaces**
- FIBEX*
- XML
- K-Matrix
- Trace tools
- WCET tools
- Mircosoft® EXCEL
- OIL*
- TraceGURU, WinRTM, ...
- aT, ...

* = work in progress

**Import / export interfaces**
- AUTOSAR*
Design-Phases

Requirements
System Design
- Architecture Exploration
- Network Timing Estimation
Module Design
- ECU Timing Estimation
Function Design
Module Test
- ECU Timing Verification
Function Test
System Test
- Network Timing Verification
- System Timing Verification

SW-Integration on ECUs
Verification of SIL-3 Project
Verify Critical ECU Timing

Active Front Steering

- ECU trace data import
- Timing analysis using SymTA/S including sensitivity analysis
- Result: reliable performance, cost savings (use of smaller CPU)

Integration: Tracing + Scheduling Analyse

- Single function execution times
- Interrupt Frequency
Measurement vs. SymTA/S Analysis

- Measured, 10ms task, Response Time 6.9ms
  - 4 CAN, 8 SPI Interrupts, 7 preemptions by 1ms task

- SymTA/S Analysis, 10ms task, Response Time 9ms
  - 10 CAN, 8 SPI Interrupts, 9 preemptions by 1ms task, blocking

CAN Bus Optimization:
CAN IDs (== message priorities),
Message offsets (traffic shaping)
Reliable CAN Bus Extension

- Problem: CAN bus load high but needs to carry more frames
- SymTA/S: Offset /CAN Id optimization → room for new frames
- Result: Increased utilization and reliable, safe extension

Detailed Sensitivity Analysis

\[ \delta \text{response time} \]
\[ \delta \text{offset} \]
CAN bus load under varying dynamic load situations

Analyzing Several Dynamic Load Profiles

- all triggered signals considered (in direct and mixed frames) ➔ full dynamic CAN load
- subset of dynamic signals ➔ representative CAN load
- no dynamic / triggered signals ➔ only periodic CAN load
Comparing Dynamic Load Profiles

Absolute Response Times for different dynamic load (AR) without Offsets

Architecture Use Cases
(today and tomorrow)
Software Component Integration (in AUTOSAR context)

- Evaluate alternatives („what if?“)
  - Mapping
  - Scheduling
  - Communication

Gated Network Analysis and Optimization

- Architecture questions
  - Topology
  - Dimensioning of ECUs and buses
  - Gateway Design
  - Function → ECU Mapping
  - Signal → Communication Mapping
End-to-end Timing Analysis is Key

- Verification and visualization of end-to-end timing
- Optimization (adding synthesis capabilities) (work in progress)

Black-box System Integration
Component Dependencies and Interfaces

- Local requirements
- Local analysis
- Black box integration
- IP protection

Restricted to components or domains

Global analysis
Full knowledge about system behavior

Between components or domains

Timing contracts

Interfaces for the Supply Chain

- Integrators: distribute the available time, bandwidth and flexibility among the suppliers
  - No details about subsystem needed

- Suppliers: work locally with available time, bandwidth and flexibility
  - Optimally fulfill requirements
  - No details about "rest of system" needed
Black-Box integration example (OEM view)

send offsets & dynamic patterns
required by OEM

guaranteed by supplier

receive offsets & jitter
required by OEM

guaranteed by supplier

Message Response Times
0 10 20 30 40 50 60 70 80 90
0 5 10 15 20 25 30
Message Number (ordered by priority)
Response Time (ms)

utilization gain
overload management time-out

Contact

Symtavision GmbH
Frankfurter Str. 3b
38122 Braunschweig
Germany

info@symtavision.com
Tel +49 886 179-0

sales@symtavision.com
Tel +49 886 179-25

www.symtavision.com
Fax +49 886 179-29
Formal analysis and optimization of heterogeneous networks in industrial practice
- from networked systems to MpSoC

Rolf Ernst, TU Braunschweig

Lessons learned

• formal performance analysis and optimization have proven to be practical in industrial design
  – in different phases of a design
  – even in very complex architectures and design processes such in automotive electronics with many players
  – greatly improving predictability and lowering design risk

• main practical challenges are
  – acquisition of basic model data
    • obtained by a variety of techniques from estimation to tracing to WCET analysis
  – design process enhancement
    • tool coupling, method coupling
    • feasible even in conservative industries
  – libraries
    • simplified in case of standardization
But – MpSoC needs different model

• distributed embedded systems (automotive)
  – local computation and memory resources
  – network mainly used for process communication

• MpSoC
  – process communication and global memory accesses are superimposed
  – more complex traffic with feedback to process execution

Process timing analysis

• traditional: task „execution“ times are combined to calculate response times
• need to include memory accesses for MPSoCs!
MpSoC process execution

- Classical process model
- Process model w/ memory access

Interference during transactions

Network arbitration

Memory arbitration

MpSoC process performance analysis

- Interference has highly dynamic behavior!
- Assuming worst case transaction is too conservative

Solution:

- Derive upper total interference bound using formal analysis
- Superimpose and continue with classical analysis

Core execution time

Core communication time + total netw. interference

Core memory access time + total mem. interference
Enhanced SymTA/S analysis engine

- so far: fixed point solution over local analysis

**environment model**

**input traffic description**

**local analysis** (WCET + WCRT)

**output traffic description**

until convergence or non-schedulability

Shared resource transaction analysis analysis

- enhanced by nested loop for process execution time determination with transaction and interference

**Data needed for enhanced analysis**

- Task behavior on CPU
  - process execution time
  - number and target of transactions

- Bus / Network
  - latency and arbitration policy

- Shared Memory
  - timing
  - arbitration and scheduling policy
  - similar for coprocessors
How to acquire these data?

- **task behavior on CPU**
  - execution time
    - simulation-traces
    - WCET analysis where applicable/available
  - transactions
    - simulation traces
    - cache traces
    - task communication analysis where applicable

- **bus / network**
  - scheduling model
  - formal system level analysis using SymTA/S

- **memory**
  - word-level timing (from datasheet)
  - arbitration model

---

Example architecture (STMicroelectronics)

![Architecture Diagram](image-url)
Application: MPEG 4 Contour Detection

- contour detection algorithm from École Polytechnique de Montreal (Gabriela Nicolesu)
- 2 – 4 processor architecture
- 2 threads per processor
  - round-robin scheduling
- StepNP simulator available

Application model

CPU 1

CPU 2

CPU 3

bus w/ communication tasks

shared memory

task
Execution Timing - Example

- system worst case task response time for task T1 based on single task simulation data
- shows little worst case interference, bus is sufficient in this simple example

Results of STMicroelectronics Example

- simulation
  - system simulation completely performed with StepNP simulator
- analysis
  - single task simulation
  - SymTA/S analysis based on single task simulation data
System Level Prediction

Worst Simulated Behavior

Predicted Worst-Case

Concurrent Behavior

• Very fast prediction of worst case behavior considering
  – Bus/Network Congestion (if any)
  – Memory Congestion
  – coprocessors ...

• Possible investigation of processor sharing, degree of parallelism / pipelining, ...

Conclusion

• new approaches to formal performance analysis have found their way into industrial practice

• worst case design successfully used for predictable and robust systems integration – supported by tools

• more complex behavior of MpSoC due to conflicting task communication and memory access

• new technique for MpSoC presented and demonstrated with a practical experiment
Acknowledgement - Literature

• more info can be found
  – www.ida.ing.tu-bs.de

• the experiments and part of the slide figures have been provided by Simon Schliecker and Mircea Negrean