Model-based Analysis of Event-driven Distributed Real-time Embedded Systems

Gabor Madl

University of California, Irvine
Bren School of Computer Science

Committee

- Chancellor’s Professor Nikil Dutt (Chair)
- Professor Tony Givargis
- Professor Ian Harris

Ph.D. Dissertation Defense, 2009
Outline

1. Introduction; Distributed Real-time Embedded (DRE) systems.
2. Formal Modeling; the ALDERIS Domain-specific Modeling Language (DSML).
3. Real-time Verification by Timed Automata (non-preemptive) [3, 7, 8, 9].
4. Performance Estimation by Discrete Event Simulations (DES) [5].
5. Conservative Approximation of Preemptive Scheduling by Timed Automata [4].
6. Combining Simulations and Model Checking for MPSoCs [1, 6].
7. Cross-abstraction Analysis of MPSoCs [2].
8. The Open-source DREAM Framework [10].
Distributed Real-time Embedded Systems
The Platform for the Implementation of Cyber-Physical Systems

- As embedded systems become increasingly networked, and interact with the physical environment, *Cyber-Physical Systems (CPS)* emerge.
  - Run in open-environments, in less predictable conditions than previous generations of embedded systems.

- *Distributed Real-time Embedded (DRE)* systems provide a highly adaptive and flexible infrastructure for reusable resource management services.
  - Provide a platform for the implementation of CPS.
  - Range from small-scale Multi-processor Systems-on-Chip (MPSoCs) to complex software-intensive systems applied to avionics, shipboard computing, power grids.

- Component-based design is an emerging design paradigm.
  - Shifts focus to a build-by-composition methodology.
As embedded systems become increasingly networked, and interact with the physical environment, *Cyber-Physical Systems (CPS)* emerge.

- Run in open-environments, in less predictable conditions than previous generations of embedded systems.

*Distributed Real-time Embedded (DRE)* systems provide a highly adaptive and flexible infrastructure for reusable resource management services.

- Provide a platform for the implementation of CPS.
- Range from small-scale Multi-processor Systems-on-Chip (MPSoCs) to complex software-intensive systems applied to avionics, shipboard computing, power grids.

Component-based design is an emerging design paradigm.

- Shifts focus to a build-by-composition methodology.
As embedded systems become increasingly networked, and interact with the physical environment, *Cyber-Physical Systems (CPS)* emerge.

- Run in open-environments, in less predictable conditions than previous generations of embedded systems.

*Distributed Real-time Embedded (DRE)* systems provide a highly adaptive and flexible infrastructure for reusable resource management services.

- Provide a platform for the implementation of CPS.
- Range from small-scale Multi-processor Systems-on-Chip (MPSoCs) to complex software-intensive systems applied to avionics, shipboard computing, power grids.

Component-based design is an emerging design paradigm.

- Shifts focus to a build-by-composition methodology.
Composing Time- and Event-Driven DRE Systems

Time-triggered Distributed Real-time Embedded (TTDRE)
- Extend the concepts of the *time-triggered architecture*.
- Time is synchronized to a global clock, high *predictability*.
- Strong in real-time aspect, weak in distributed and embedded.

Asynchronous Event-driven Distributed Real-time Embedded (AEDRE)
- Reactive, event-driven communication paradigm.
- Better utilization, energy consumption than TTDRE.
- Strong in distributed and embedded aspect.
Composing Time- and Event-Driven DRE Systems

Time-triggered Distributed Real-time Embedded (TTDRE)
- Extend the concepts of the *time-triggered architecture*.
- Time is synchronized to a global clock, high *predictability*.
- Strong in real-time aspect, weak in distributed and embedded.

Asynchronous Event-driven Distributed Real-time Embedded (AEDRE)
- Reactive, event-driven communication paradigm.
- Better utilization, energy consumption than TTDRE.
- Strong in distributed and embedded aspect.
Key Contributions of this Dissertation

1. Definition of a formal semantic domain for AEDRE systems.
   - Based on timed automata and discrete event systems.

2. A model checking method for the real-time verification of non-preemptive DRE systems by timed automata.
   - Abstract model of dependencies and platform.

3. A performance estimation method for DRE systems by *Discrete Event Simulations (DES).*
   - CPU-bound real-time analysis method, can achieve 100% coverage.

4. A conservative approximation method for the verification of preemptive DRE systems by timed automata.
   - First decidable method for the real-time model checking of AEDRE systems.
Key Contributions of this Dissertation

1. Definition of a formal semantic domain for AEDRE systems. Based on timed automata and discrete event systems.
3. A performance estimation method for DRE systems by Discrete Event Simulations (DES). CPU-bound real-time analysis method, can achieve 100% coverage.
4. A conservative approximation method for the verification of preemptive DRE systems by timed automata. First decidable method for the real-time model checking of AEDRE systems.
Key Contributions of this Dissertation

1. Definition of a formal semantic domain for AEDRE systems.
   - Based on timed automata and discrete event systems.

2. A model checking method for the real-time verification of non-preemptive DRE systems by timed automata.
   - Abstract model of dependencies and platform.

   - CPU-bound real-time analysis method, can achieve 100% coverage.

4. A conservative approximation method for the verification of preemptive DRE systems by timed automata.
   - First decidable method for the real-time model checking of AEDRE systems.
Key Contributions of this Dissertation

1. Definition of a formal semantic domain for AEDRE systems.
   - Based on timed automata and discrete event systems.

2. A model checking method for the real-time verification of non-preemptive DRE systems by timed automata.
   - Abstract model of dependencies and platform.

3. A performance estimation method for DRE systems by *Discrete Event Simulations (DES)*.
   - CPU-bound real-time analysis method, can achieve 100% coverage.

4. A conservative approximation method for the verification of preemptive DRE systems by timed automata.
   - First decidable method for the real-time model checking of AEDRE systems.
Application Domains Investigated in this Dissertation

1. The domain of software-intensive DRE systems, in the context of the *Boeing Bold Stroke* execution platform.

2. The cross-abstraction analysis of multimedia *Multi-Processor Systems-on-Chip (MPSoCs)*.

3. The novelty of our approach:
   - Combining formal methods and symbolic simulations.
   - Utilizing multiple abstractions to trade off analysis accuracy in scalability.
Application Domains Investigated in this Dissertation

1. The domain of software-intensive DRE systems, in the context of the Boeing Bold Stroke execution platform.

2. The cross-abstraction analysis of multimedia Multi-Processor Systems-on-Chip (MPSoCs).

3. The novelty of our approach:
   - Combining formal methods and symbolic simulations.
   - Utilizing multiple abstractions to trade off analysis accuracy in scalability.

Key Contributions

- Performance Estimation by Discrete Event Simulations
- Model Checking Distributed Real-time Embedded Systems
- Semantic Domain for Asynchronous Event-driven Distributed Real-time Embedded Systems
- Formal Analysis of AMBA-based MPSoCs
- Conservative Approximation of Preemptive Scheduling
- Real-time Verification
The domain of software-intensive DRE systems, in the context of the *Boeing Bold Stroke* execution platform.

2. The cross-abstraction analysis of multimedia *Multi-Processor Systems-on-Chip (MPSoCs)*.

3. The novelty of our approach:
   - Combining formal methods and symbolic simulations.
   - Utilizing multiple abstractions to trade off analysis accuracy in scalability.
Model-based Design and Analysis of DRE Systems

- Design flow driven by DSM, a high-level specification that captures key properties.
- DSM mapped to formal executable model for verification and evaluation.
- Formal models drive functional verification.
- Combine simulations and formal methods.
The Alderis DSML is a 6-tuple $A = \{ T, C, TR, TH, M, D \}$ where:

- $T$ is a set of tasks,
- $C$ is a set of communication channels: $C \subseteq T$,
- $TR$ is a set of timers: $TR \subseteq T$,
- $TH$ is a set of execution threads,
- $M$ is a set of machines.
- $D$ is the task dependency relationship: $D \subseteq T \times T$. 

**Specifying DSMLs by Meta-modeling**

[Diagram showing Alderis DSML concepts]
The DRE Semantic Domain
Timed Automata Representation

- **Timer:** publishes events at periodic time intervals.
- **Nonp Task:** models computation time.
- **Task:** approximates a preemptable task.
- **Buffer:** models FIFO, allows non-blocking communication.
- **Channel:** Buffer with delay.
- **Scheduler:** specifies the scheduling policy.
A discrete event system is a 5-tuple $G = (Q, \Sigma, \delta, q_0, Q_m)$

- $Q$ is a finite set of states,
- $\Sigma$ is a finite alphabet of event labels,
- $\delta : Q \times \Sigma \rightarrow Q$ is the transition function,
- $q_0$ is the initial state,
- and $Q_m$ is the set of marker states (exiting states).
Real-time Verification by Timed Automata

- DSM specified using Alderis.
- Alderis mapped to timed automata by Dream.
- Model check timed automata by UPPAAL or Verimag IF.
- Applied to software-intensive DRE systems.
DSM specified using Alderis.
Alderis mapped to timed automata by Dream.
Model check timed automata by Uppaal or Verimag IF.
Applied to software-intensive DRE systems.
The Boeing Bold Stroke Execution Platform

- Real-time CORBA avionics application.
- Software timers, asynchronous event propagation.
- Thread pool policy: *half-sync half-async*.
- Event channels have their own thread, *asynchronous message invocation (AMI)*.
Non-preemptive multi-processor platform.

5 Timers on 5 CPUs.

Captures dependencies and event flow.

Communication delays modeled as Channels.

Problem: deciding schedulability.
Partial Timed Automata Model of the Application

- **af_monitor**
- **airframe**
- **gps**

**Model-based Analysis of DRE Systems**
Performance Estimation by DES

- DSM specified using Alderis.
- Alderis utilized by Dream using discrete event semantics.
- Use Dream for performance estimation or real-time verification.
- Applied to software-intensive DRE systems.
Performance Estimation by DES

- DSM specified using Alderis.
- Alderis utilized by Dream using discrete event semantics.
- Use Dream for performance estimation or real-time verification.
- Applied to software-intensive DRE systems.
Performance Estimation Problem

Performance and Schedulability Analysis

- In AEDRE systems WCET analysis is not sufficient.
- The product of local WCET times does not necessarily correspond to the global WCET.
- Analysis has to capture execution intervals in continuous time.

Motivating Example for Non-WCET Deadline Miss

BCET  WCET  (BCET, WCET)
DE semantics, “jump” to event with the smallest timestamp.

A *run* is the chronological sequence of events occurred.

2 execution traces are equivalent, iff they contain the same events in the same order.

Enumerate equivalent execution traces.
Evaluating the Proposed Method

Random/Directed Simulations
- Proposed method achieves 100% coverage.
- Continuously increases coverage.

Static Analysis Methods
- Captures dynamic effects such as race conditions, congestions on bus.
- More accurate.

Timed Automata Model Checking
- Calculates WCET instead of answering yes/no question.
- CPU-bound; it can return partial results on large models.
Evaluating the Proposed Method

Random/Directed Simulations
- Proposed method achieves 100% coverage.
- Continuously increases coverage.

Static Analysis Methods
- Captures dynamic effects such as race conditions, congestions on bus.
- More accurate.

Timed Automata Model Checking
- Calculates WCET instead of answering yes/no question.
- CPU-bound; it can return partial results on large models.
### Evaluating the Proposed Method

#### Random/Directed Simulations
- Proposed method achieves 100% coverage.
- Continuously increases coverage.

#### Static Analysis Methods
- Captures dynamic effects such as race conditions, congestions on bus.
- More accurate.

#### Timed Automata Model Checking
- Calculates WCET instead of answering yes/no question.
- CPU-bound; it can return partial results on large models.
Conservative Approximation of Preemptive Scheduling

- DSM specified using Alderis.
- Alderis mapped to stopwatch automata – decidability issues.
- Alderis mapped to timed automata approximation by Dream.
- Model check timed automata by UPPAAL or Verimag IF.
- Applied to software-intensive DRE systems.
DSM specified using \textit{Alderis}.

\textit{Alderis} mapped to stopwatch automata – decidability issues.

\textit{Alderis} mapped to timed automata approximation by \textit{Dream}.

Model check timed automata by \textit{Uppaal} or Verimag IF.

Applied to software-intensive DRE systems.
Modeling a Preemptable Task

- Stopwatch can be stopped, resumed and resetted.
- Clock valuation $v_t$ measures time passed since enabling event.
- Clock valuation $v_{sw}$ measures time spent executing.
  - $0 \leq v_{sw} \leq v_t$, $0 \leq v_{sw} \leq wcet$.
  - $en_i$ start (stop start)$^*$ $en_j$
  - $\sum_{i=1}^{e} \tau_{2i} - \tau_{2i-1} \leq dl - wcet$.

Task Stopwatch Automata

- $v_t := 0$
- $v_{sw} := 0$
- $\dot{v}_t = 0$, $\dot{v}_{sw} = 0$
- enable$_i$?
- $\dot{v}_t = 1$, $\dot{v}_{sw} = 0$
- stop?
- $\dot{v}_t = 1$, $\dot{v}_{sw} = 1$
- $bcet \leq v_{sw} \leq wcet$
- $v_t \leq dl$
- enable$_j$!
- $v_{sw} \leq wcet$
- finish
- run
Modeling a Preemptable Task

- Stopwatch can be stopped, resumed and resetted.
- Clock valuation $v_t$ measures time passed since enabling event.
- Clock valuation $v_{sw}$ measures time spent executing.
- $0 \leq v_{sw} \leq v_t$, $0 \leq v_{sw} \leq wcet$.
- $en_i$ start (stop start) * $en_j$
- $\sum_{i=1}^{e} \tau_{2i} - \tau_{2i-1} \leq dl - wcet$.

**Task Stopwatch Automata**

- $v_t := 0$
- $v_{sw} := 0$

**States:**
- **idle**
  - $\dot{v}_t = 0$
  - $\dot{v}_{sw} = 0$
- **stop**
  - $\dot{v}_t = 1$
  - $\dot{v}_{sw} = 0$
- **run**
  - $\dot{v}_t = 1$
  - $\dot{v}_{sw} = 1$
- **finish**
  - $\dot{v}_t = 1$
  - $\dot{v}_{sw} = 0$

**Transitions:**
- $enable_i$?
- $stop$?
- $start$?
- $bcet \leq v_{sw} \leq wcet$
- $\dot{v}_t \leq dl$
- $enable_j!$
- $v_{sw} \leq wcet$
Conservative Approximation of Preemptive Scheduling

Task Timed Automata

Approximating TSA by TTA

- Reachability analysis undecidable on stopwatch automata in general.
- Approximate stopwatch automata by timed automata.
- If TTA schedulable $\Rightarrow$ TSA schedulable.

Gabor Madl (UC Irvine)
Approximating TSA by TTA

Reachability analysis undecidable on stopwatch automata in general.
Approximate stopwatch automata by timed automata.
If TTA schedulable $\rightarrow$ TSA schedulable.
TTA Schedulable $\rightarrow$ TSA Schedulable

- $S_{L(S)} = \text{enable}_i \text{ start (stop start)}^* \text{ enable}_j$.
- $S_{L(T)} = \text{enable}_i \text{ start (stop start)}^0...^m \text{ enable}_j$.

- TSA: $\sum_{i=1}^{e} \tau_{2i} - \tau_{2i-1} \leq dl - wcet$.
- TTA: $\sum_{i=1}^{e} \tau_{2i} - \tau_{2i-1} + t_u \leq dl - wcet$.

- $L(T) \subseteq L(S)$ iff $L(T) \cap \overline{L(S)} = \emptyset$.
- $t_{stop} + \sum_{i=1}^{e} t_u \leq \text{dl-wcet} \cap t_{stop} > \text{dl-wcet}$.
- Since $t_u \in \mathbb{R}_{\geq 0}$, therefore $0 \leq \sum_{i=1}^{e} t_u$.
- Contradiction.
- If TTA schedulable $\rightarrow$ TSA schedulable.
TTA Schedulable $\rightarrow$ TSA Schedulable

- $S_{L(S)} = \text{enable}_i \text{ start (stop start)}^* \text{ enable}_j$.
- $S_{L(T)} = \text{enable}_i \text{ start (stop start)}^0...^m \text{ enable}_j$.
- \[ \text{TSA: } \sum_{i=1}^{e} \tau_{2i} - \tau_{2i-1} \leq \text{dl} - \text{wcet}. \]
- \[ \text{TTA: } \sum_{i=1}^{e} \tau_{2i} - \tau_{2i-1} + t_u \leq \text{dl} - \text{wcet}. \]
- \[ L(T) \subseteq L(S) \text{ iff } L(T) \cap \overline{L(S)} = \emptyset. \]
- \[ t_{stop} + \sum_{i=1}^{e} t_u \leq \text{dl} - \text{wcet} \cap t_{stop} > \text{dl} - \text{wcet}. \]
- Since $t_u \in \mathbb{R}_{\geq 0}$, therefore $0 \leq \sum_{i=1}^{e} t_u$.
- Contradiction.
- If TTA schedulable $\rightarrow$ TSA schedulable.
TTA Schedulable → TSA Schedulable

- $S_L(S) = \text{enable}_i \text{ start (stop start)}^* \text{ enable}_j$.
- $S_L(T) = \text{enable}_i \text{ start (stop start)}^0 \ldots^m \text{ enable}_j$.
- TSA: $\sum_{i=1}^{\frac{e}{2}} \tau_{2i} - \tau_{2i-1} \leq \text{dl} - \text{wcet}$.
- TTA: $\sum_{i=1}^{\frac{e}{2}} \tau_{2i} - \tau_{2i-1} + t_u \leq \text{dl} - \text{wcet}$.
- $L(T) \subseteq L(S)$ iff $L(T) \cap \overline{L(S)} = \emptyset$.
- $t_{\text{stop}} + \sum_{i=1}^{\frac{e}{2}} t_u \leq \text{dl} - \text{wcet} \cap t_{\text{stop}} > \text{dl} - \text{wcet}$.
- Since $t_u \in \mathbb{R}_{\geq 0}$, therefore $0 \leq \sum_{i=1}^{\frac{e}{2}} t_u$.
- Contradiction.
- If TTA schedulable $\rightarrow$ TSA schedulable.
TTA Schedulable $\rightarrow$ TSA Schedulable

- $S_{L(S)} = \text{enable}_i \text{ start (stop start)}^* \text{ enable}_j$.
- $S_{L(T)} = \text{enable}_i \text{ start (stop start)}^{0\ldots m} \text{ enable}_j$.
- TSA: $\sum_{i=1}^{e} \tau_{2i} - \tau_{2i-1} \leq dl - wcet$.
- TTA: $\sum_{i=1}^{e} \tau_{2i} - \tau_{2i-1} + t_u \leq dl - wcet$.
- $L(T) \subseteq L(S)$ iff $L(T) \cap \overline{L(S)} = \emptyset$.
- $t_{stop} + \sum_{i=1}^{e} t_u \leq dl - wcet \cap t_{stop} > dl - wcet$.
- Since $t_u \in \mathbb{R}_{\geq 0}$, therefore $0 \leq \sum_{i=1}^{e} t_u$.
- Contradiction.
- If TTA schedulable $\rightarrow$ TSA schedulable.
**TTA Schedulable → TSA Schedulable**

- $S_{L(S)} = \text{enable}_i \text{ start (stop start)}^* \text{ enable}_j$.
- $S_{L(T)} = \text{enable}_i \text{ start (stop start)}^{0...m} \text{ enable}_j$.
- **TSA:** $\sum_{i=1}^{\frac{e}{2}} \tau_{2i} - \tau_{2i-1} \leq dl - wcet$.
- **TTA:** $\sum_{i=1}^{\frac{e}{2}} \tau_{2i} - \tau_{2i-1} + t_u \leq dl - wcet$.
- $L(T) \subseteq L(S)$ iff $L(T) \cap \overline{L(S)} = \emptyset$.
- $t_{stop} + \sum_{i=1}^{\frac{e}{2}} t_u \leq dl - wcet \cap t_{stop} > dl - wcet$.
- Since $t_u \in \mathbb{R}_{\geq 0}$, therefore $0 \leq \sum_{i=1}^{\frac{e}{2}} t_u$.
- Contradiction.
- If TTA schedulable $\rightarrow$ TSA schedulable.
• DSM specified using ALDERIS.
• ALDERIS mapped to finite state machine model – manually using templates.
• SystemC implementation of the model used to obtain execution parameters.
• Annotate formal models by simulation data.
• Perform model checking on the FSM model by NuSMV.
• Applied to JPEG 2000 multimedia design.
Combining Simulations and Model Checking for MPSoCs

- DSM specified using Alderis.
- Alderis mapped to finite state machine model – manually using templates.
- SystemC implementation of the model used to obtain execution parameters.
- Annotate formal models by simulation data.
- Perform model checking on the FSM model by NuSMV.
- Applied to JPEG 2000 multimedia design.
Challenges in MPSoC Design

1. Functional Verification
   - Deadlock-freedom and livelock-freedom not guaranteed.
   - Access to the bus is managed by an arbiter (or several arbiters).
   - Communication subsystem has a major impact.
   - Cycle-accurate *Finite State Machine (FSM)* model.

2. Performance Estimation
   - MPSoCs can be viewed as DRE systems.
   - Point arbitration in fully connected bus matrices modeled as non-preemptive scheduling.
   - Transaction-level *Discrete Event Simulations (DES)*.

3. Real-time Verification
   - Need to ensure real-time schedulability.
   - End-to-end real-time constraints.
   - Transaction-level *timed automata* model checking.
Challenges in MPSoC Design

1. Functional Verification
   - Deadlock-freedom and livelock-freedom not guaranteed.
   - Access to the bus is managed by an arbiter (or several arbiters).
   - Communication subsystem has a major impact.
   - Cycle-accurate *Finite State Machine (FSM)* model.

2. Performance Estimation
   - MPSoCs can be viewed as DRE systems.
   - Point arbitration in fully connected bus matrices modeled as non-preemptive scheduling.
   - Transaction-level *Discrete Event Simulations (DES)*.

3. Real-time Verification
   - Need to ensure real-time schedulability.
   - End-to-end real-time constraints.
   - Transaction-level *timed automata* model checking.
Challenges in MPSoC Design

1. Functional Verification
   - Deadlock-freedom and livelock-freedom not guaranteed.
   - Access to the bus is managed by an arbiter (or several arbiters).
   - Communication subsystem has a major impact.
   - Cycle-accurate *Finite State Machine (FSM)* model.

2. Performance Estimation
   - MPSoCs can be viewed as DRE systems.
   - Point arbitration in fully connected bus matrices modeled as non-preemptive scheduling.
   - Transaction-level *Discrete Event Simulations (DES)*.

3. Real-time Verification
   - Need to ensure real-time schedulability.
   - End-to-end real-time constraints.
   - Transaction-level *timed automata* model checking.
Combining Simulations and Model Checking for MPSoCs

Functional Verification

Modeling AMBA AHB

- Cycle-accurate *Finite State Machine (FSM)* model.
- Modeled generic master (6 states), slave (4 slaves), round-robin arbiter.
- Use FSM models for functional verification & performance estimation.
- Uncovered previously undocumented ambiguity.

UCIrvine
Gabor Madl (UC Irvine)
Ambiguity in the AMBA AHB Protocol

Possible Deadlock

- Slave has previously split Master_1.
- Slave in transaction with Master_2.
- Issue HSPLIT_1 while setting RETRY response.
- Should arbiter hold its state, or unmask Master_1?
Ambiguity in the AMBA AHB Protocol

Possible Deadlock

- Slave has previously split Master_1.
- Slave in transaction with Master_2.
- Issue HSPLIT\_1 while setting RETRY response.
- Should arbiter hold its state, or unmask Master_1?
Combining Simulations and Model Checking for MPSoCs

Application to JPEG 2000 Encoder

Combining Simulations and Model Checking for MPSoC Evaluation

- Obtain execution intervals for components by SystemC simulations.
- Calculate size of messages on bus.
- Annotate formal models with parameters from simulations.
- Model check to find actual end-to-end WCET times.
Combining Simulations and Model Checking for MPSoC Evaluation

- Obtain execution intervals for components by *SystemC* simulations.
- Calculate size of messages on bus.
- Annotate formal models with parameters from simulations.
- Model check to find actual end-to-end WCET times.
Comparing Simulations and Model Checking

Performance Estimates

Communication Overhead Estimates
Cross-abstraction Real-time Analysis of MPSoCs

- **SystemC** implementation of the model used to obtain execution parameters.
- Perform functional verification on the FSM model by NuSMV.
- Use **DREAM** for performance estimation.
- Real-time verification of timed automata by **UPPAAL** or Verimag IF.
- Applied to networking router design.
Cross-abstraction Real-time Analysis of MPSoCs

- **SystemC** implementation of the model used to obtain execution parameters.
- Perform functional verification on the FSM model by NuSMV.
- Use **DREAM** for performance estimation.
- Real-time verification of timed automata by **UPPAAL** or Verimag IF.
- Applied to networking router design.
Transaction-level simulations improve the analysis performance for MPSoCs.

Could we trade off analysis performance and accuracy for model checking?

Key contribution; show how analysis methods for DRE systems can be applied to MPSoCs.
Cross-abstraction Real-time Analysis of MPSoCs

The CARTA Framework

Analysis of Bus Matrix MPSoC Designs

The CARTA Framework

- Transaction-level simulations improve the analysis performance for MPSoCs.
- Could we trade off analysis performance and accuracy for model checking?
- Key contribution; show how analysis methods for DRE systems can be applied to MPSoCs.
Transaction-level simulations improve the analysis performance for MPSoCs.

Could we trade off analysis performance and accuracy for model checking?

Key contribution; show how analysis methods for DRE systems can be applied to MPSoCs.
Formal analysis is simplified in fully connected bus matrices.

Point arbitration can be expressed as non-preemptive scheduling.

Transaction-level Modeling of Bus Matrix Interconnects

Networking Router Design

ALDERIS Model
Applying Cross-abstraction Analysis to Networking Router Design

Analysis Time and Memory Consumption

- Functional verification at cycle-accurate abstraction, by FSM.
- Performance estimation at transaction-level, by DES.
- Real-time verification at transaction-level, by timed automata.
The Open-source DREAM Framework

**ALDERIS model**
- GME tool

**Open-source DREAM Tool**
- Automatic timed automata model generation for the UPPAAL and Verimag IF tools
- Simulation-guided model checking
- Performance estimation using DES
- Random testing
- Schedulability optimizations

**VERIMAG IF**
- Model checker

**UPPAAL**
- Model checker

Gabor Madl (UC Irvine)  
Model-based Analysis of DRE Systems  
Final Defense 33 / 40
Concluding Remarks

1. Formal Modeling; the Alderis Domain-specific Modeling Language (DSML).
2. Real-time Verification by Timed Automata (non-preemptive) [3, 7, 8, 9].
3. Performance Estimation by Discrete Event Simulations (DES) [5].
5. Combining Simulations and Model Checking for MPSoCs [1, 6].
6. Cross-abstraction Analysis of MPSoCs [2].
7. The Open-source Dream Framework [10].
Future Work

- Hierarchical (compositional) approach to verification.
- Energy (and power) verification.
- Multi-core implementation of the DES-based real-time analysis method.
- Integration with real-time calculus.
- Real-time kernel based on the open-source Dream tool.
- Run-time analysis.
- Integration with Model Predictive Control/Supervisory Control.
Related Publications I


Related Publications III


G. Madl and N. Dutt.

G. Madl and N. Dutt.
Tutorial for the Open-source DREAM Tool.
Questions?

- Open-source **DREAM** Tool:
  
  http://dre.sourceforge.net

- **ALDERIS** Modeling Language:
  
  http://alderis.ics.uci.edu