

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

- ➊ Introduction; *Distributed Real-time Embedded (DRE)* systems.
- ➋ Formal Modeling; the *ALDERIS Domain-specific Modeling Language (DSML)*.
- ➌ Real-time Verification by Timed Automata (non-preemptive) [3, 7, 8, 9].
- ➍ Performance Estimation by *Discrete Event Simulations (DES)* [5].
- ➎ Conservative Approximation of Preemptive Scheduling by Timed Automata [4].
- ➏ Combining Simulations and Model Checking for MPSoCs [1, 6].
- ➐ Cross-abstraction Analysis of MPSoCs [2].
- ➑ 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.

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.

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.

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

- ➊ Definition of a formal semantic domain for AEDRE systems.
 - Based on timed automata and discrete event systems.
- ➋ A model checking method for the real-time verification of non-preemptive DRE systems by timed automata.
 - Abstract model of dependencies and platform.
- ➌ A performance estimation method for DRE systems by *Discrete Event Simulations (DES)*.
 - CPU-bound real-time analysis method, can achieve 100% coverage.
- ➍ 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

- ➊ Definition of a formal semantic domain for AEDRE systems.
 - Based on timed automata and discrete event systems.
- ➋ A model checking method for the real-time verification of non-preemptive DRE systems by timed automata.
 - Abstract model of dependencies and platform.
- ➌ A performance estimation method for DRE systems by *Discrete Event Simulations (DES)*.
 - CPU-bound real-time analysis method, can achieve 100% coverage.
- ➍ 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.

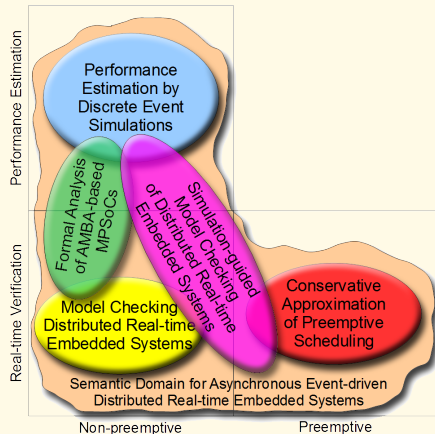
Key Contributions of this Dissertation

- 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.

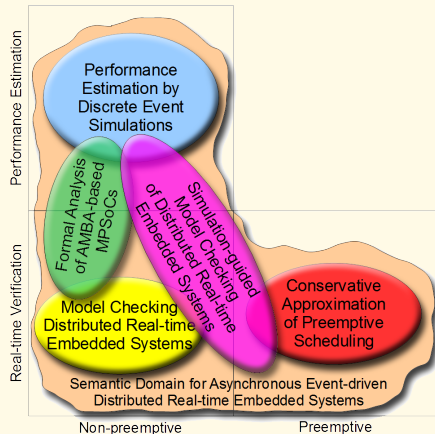
Key Contributions



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.

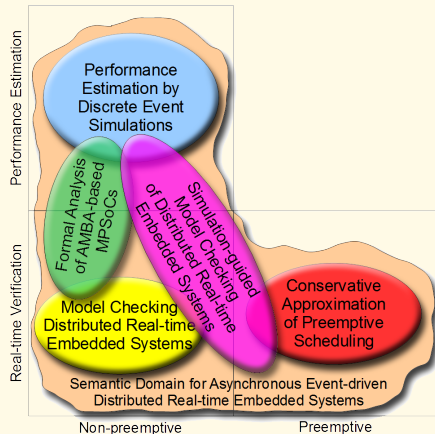
Key Contributions



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.

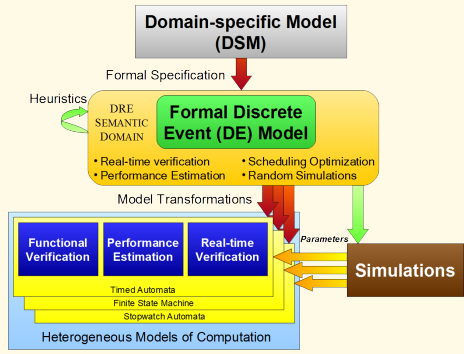
Key Contributions



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.

Model-based Analysis

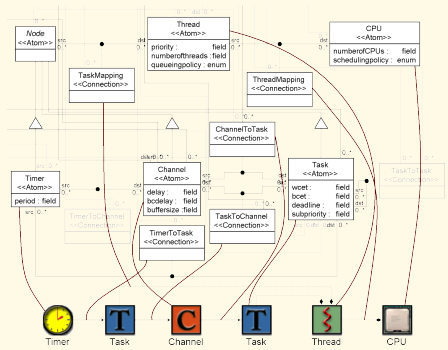


Specifying the ALDERIS DSML by Meta-modeling

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

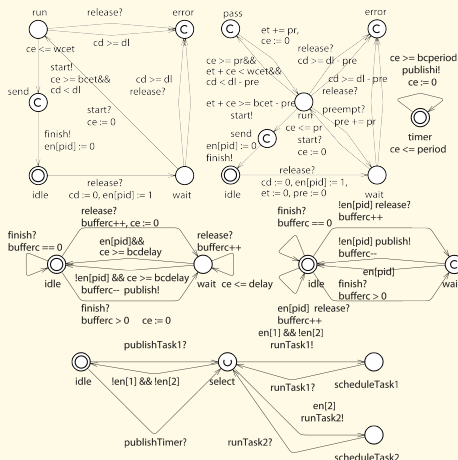


The DRE SEMANTIC DOMAIN

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.

The DRE SEMANTIC DOMAIN



The DRE SEMANTIC DOMAIN

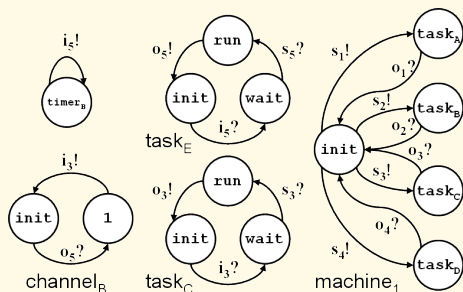
Discrete Event Representation

A discrete event system is a 5-tuple

$$G = (Q, \Sigma, \delta, q_0, Q_m)$$

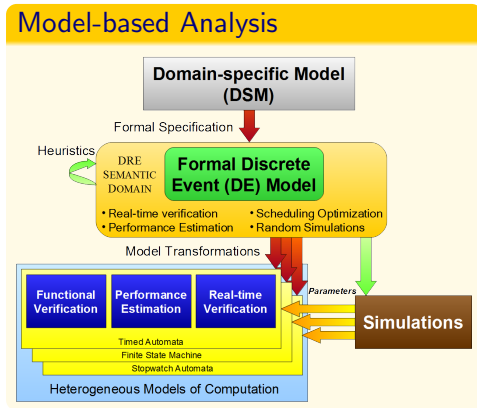
- Q is a finite set of states,
- Σ is a finite alphabet of *event labels*,
- $\delta : Q \times \Sigma \rightarrow Q$ is the transition function,
- q_0 is the initial state,
- and Q_m is the set of *marker states* (exiting states).

The DRE SEMANTIC DOMAIN



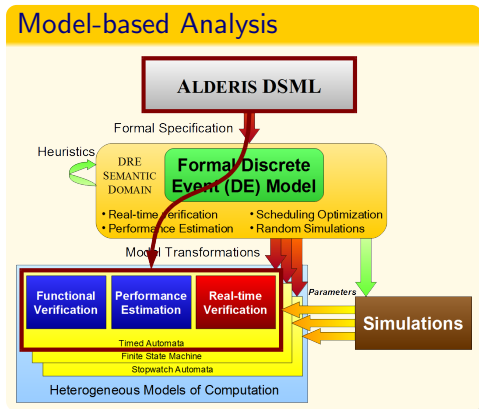
Real-time Verification by Timed Automata

- 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.



Real-time Verification by Timed Automata

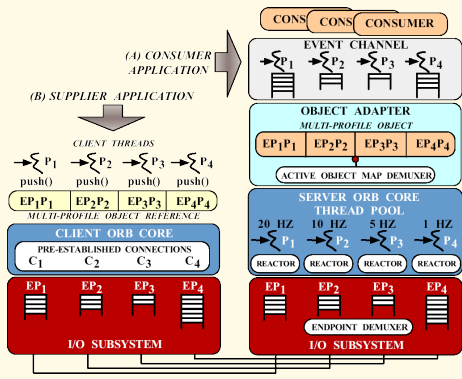
- DSM specified using ALDERIS.
- ALDERIS mapped to timed automata by DREAM.
- Model check timed automata by UPPAAL or Verimag IF.
- Applied to software-intensive DRE systems.



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)*.

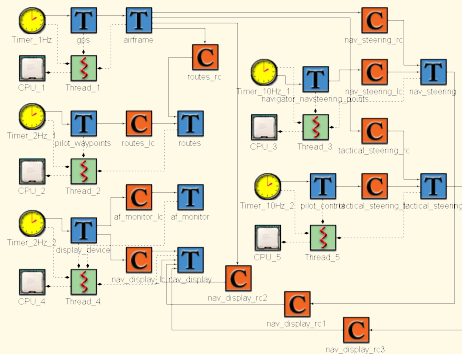
The Bold Stroke Platform



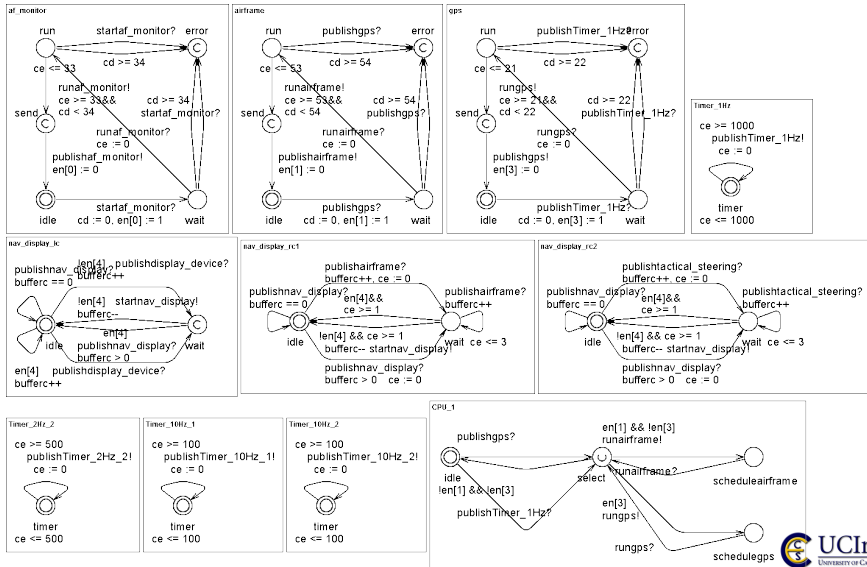
Modeling The DRE Avionics Application by ALDERIS

- Non-preemptive multi-processor platform.
- 5 Timers on 5 CPUs.
- Captures dependencies and event flow.
- Communication delays modeled as Channels.
- Problem: deciding schedulability.

Avionics DRE Application

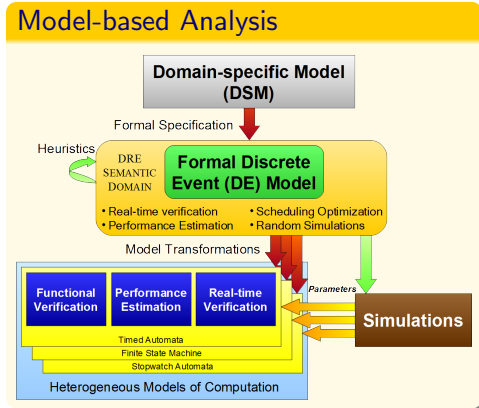


Partial Timed Automata Model of the Application



Performance Estimation by DES

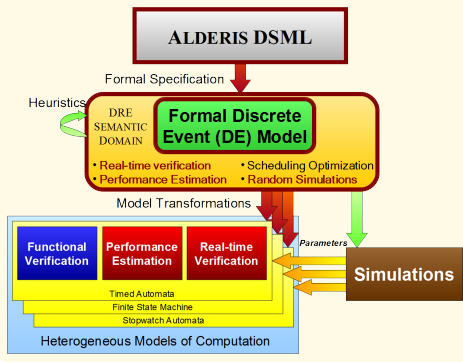
- DSM specified using ALDERIS.
- ALDERIS utilized by DREAM using discrete event semantics.
- Use DREAM for performance estimation or real-time verification.
- Applied to software-intensive DRE systems.



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.

Model-based Analysis



Performance Estimation Problem

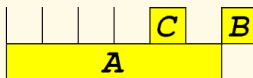
Performance and Schedulability Analysis

- 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.

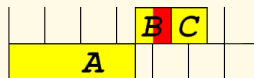
Motivating Example for Non-WCET Deadline Miss



BCET

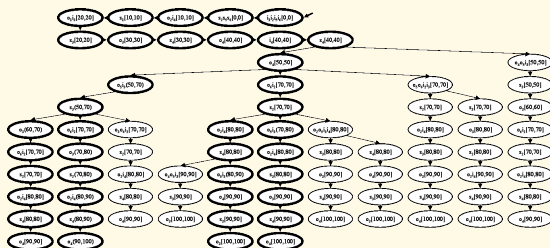



WCET



(BCET, WCET)

Event Order Tree



- 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.

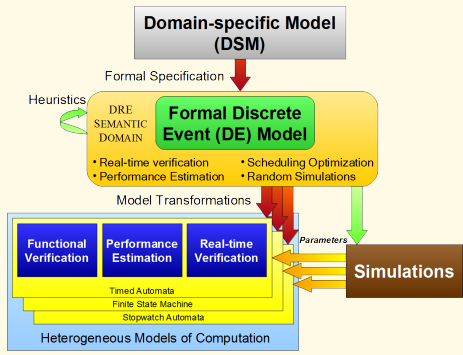
Timed Automata Model Checking

- 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.

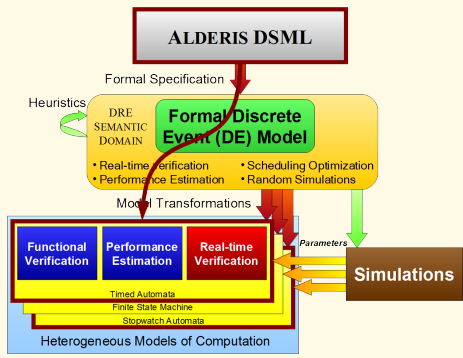
Model-based Analysis



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.

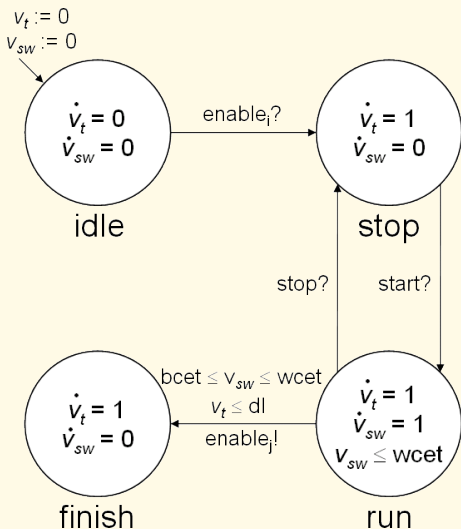
Model-based Analysis



Modeling a Preemptable Task

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

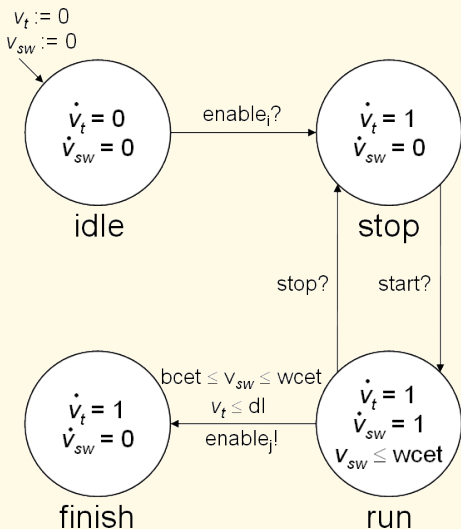
Task Stopwatch Automata



Modeling a Preemptable Task

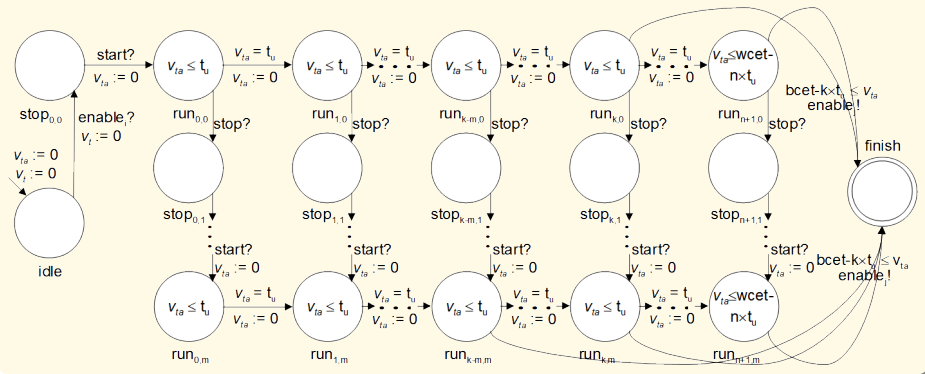
- Stopwatch can be stopped, resumed and reset.
- Clock valuation v_t measures time passed since enabling event.
- Clock valuation v_{sw} measures time spent executing.
- $0 \leq v_{sw} \leq v_t$, $0 \leq v_{sw} \leq \text{wcet}$.
- $\text{en}_i \text{ start (stop start)}^* \text{ en}_j$
- $\sum_{i=1}^e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

Task Stopwatch Automata



Approximating TSA by TTA

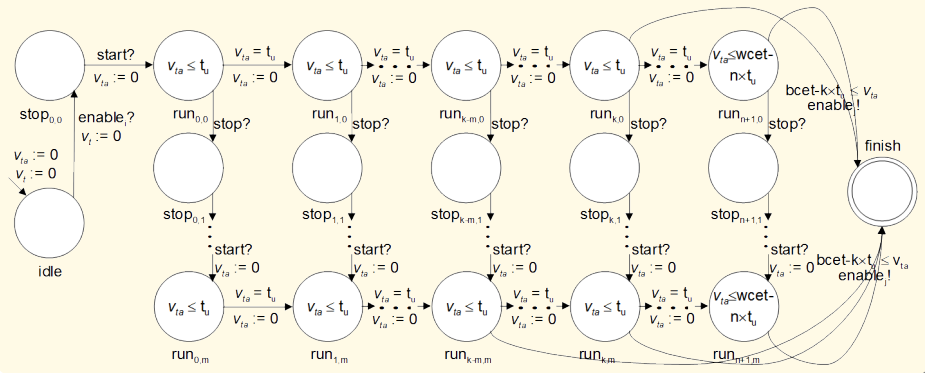
Task Timed Automata



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

Approximating TSA by TTA

Task Timed Automata



- 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

- $S_{L(S)} = \text{enable}_i \text{ start } (\text{stop start})^* \text{ enable}_j$.
- $S_{L(T)} = \text{enable}_i \text{ start } (\text{stop start})^{0\dots m} \text{ enable}_j$.
- TSA: $\sum_{i=1}^{\frac{e}{2}} \tau_{2i} - \tau_{2i-1} \leq \text{dl-wcet}$.
- TTA: $\sum_{i=1}^{\frac{e}{2}} \tau_{2i} - \tau_{2i-1} + t_u \leq \text{dl-wcet}$.
- $L(T) \subseteq L(S)$ iff $L(T) \cap \overline{L(S)} = \emptyset$.
- $t_{\text{stop}} + \sum_{i=1}^{\frac{e}{2}} t_u \leq \text{dl-wcet} \cap t_{\text{stop}} > \text{dl-wcet}$.
- 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.

TTA Schedulable \rightarrow TSA Schedulable

- $S_{L(S)} = \text{enable}_i \text{ start } (\text{stop start})^* \text{ enable}_j$.
- $S_{L(T)} = \text{enable}_i \text{ start } (\text{stop start})^{0\dots m} \text{ enable}_j$.
- TSA: $\sum_{i=1}^{\frac{e}{2}} \tau_{2i} - \tau_{2i-1} \leq \text{dl-wcet}$.
- TTA: $\sum_{i=1}^{\frac{e}{2}} \tau_{2i} - \tau_{2i-1} + t_u \leq \text{dl-wcet}$.
- $L(T) \subseteq L(S)$ iff $L(T) \cap \overline{L(S)} = \emptyset$.
- $t_{\text{stop}} + \sum_{i=1}^{\frac{e}{2}} t_u \leq \text{dl-wcet} \cap t_{\text{stop}} > \text{dl-wcet}$.
- 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.

TTA Schedulable \rightarrow TSA Schedulable

- $S_{L(S)} = \text{enable}_i \text{ start } (\text{stop start})^* \text{ enable}_j$.
- $S_{L(T)} = \text{enable}_i \text{ start } (\text{stop start})^{0\dots m} \text{ enable}_j$.
- TSA: $\sum_{i=1}^{\frac{e}{2}} \tau_{2i} - \tau_{2i-1} \leq \text{dl-wcet}$.
- TTA: $\sum_{i=1}^{\frac{e}{2}} \tau_{2i} - \tau_{2i-1} + t_u \leq \text{dl-wcet}$.
- $L(T) \subseteq L(S)$ iff $L(T) \cap \overline{L(S)} = \emptyset$.
- $t_{\text{stop}} + \sum_{i=1}^{\frac{e}{2}} t_u \leq \text{dl-wcet} \cap t_{\text{stop}} > \text{dl-wcet}$.
- 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.

TTA Schedulable \rightarrow TSA Schedulable

- $S_{L(S)} = \text{enable}_i \text{ start } (\text{stop start})^* \text{ enable}_j$.
- $S_{L(T)} = \text{enable}_i \text{ start } (\text{stop start})^{0\dots m} \text{ enable}_j$.
- TSA: $\sum_{i=1}^{\frac{e}{2}} \tau_{2i} - \tau_{2i-1} \leq \text{dl-wcet}$.
- TTA: $\sum_{i=1}^{\frac{e}{2}} \tau_{2i} - \tau_{2i-1} + t_u \leq \text{dl-wcet}$.
- $L(T) \subseteq L(S)$ iff $L(T) \cap \overline{L(S)} = \emptyset$.
- $t_{\text{stop}} + \sum_{i=1}^{\frac{e}{2}} t_u \leq \text{dl-wcet} \cap t_{\text{stop}} > \text{dl-wcet}$.
- 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.

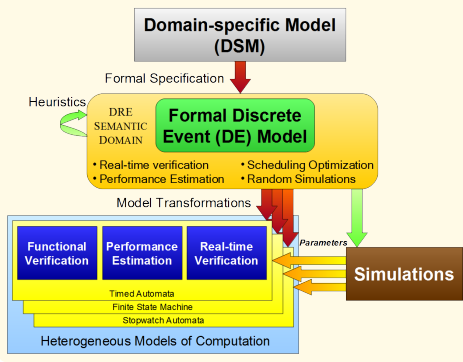
TTA Schedulable \rightarrow TSA Schedulable

- $S_{L(S)} = \text{enable}_i \text{ start } (\text{stop start})^* \text{ enable}_j$.
- $S_{L(T)} = \text{enable}_i \text{ start } (\text{stop start})^{0\dots m} \text{ enable}_j$.
- TSA: $\sum_{i=1}^{\frac{e}{2}} \tau_{2i} - \tau_{2i-1} \leq \text{dl-wcet}$.
- TTA: $\sum_{i=1}^{\frac{e}{2}} \tau_{2i} - \tau_{2i-1} + t_u \leq \text{dl-wcet}$.
- $L(T) \subseteq L(S)$ iff $L(T) \cap \overline{L(S)} = \emptyset$.
- $t_{\text{stop}} + \sum_{i=1}^{\frac{e}{2}} t_u \leq \text{dl-wcet} \cap t_{\text{stop}} > \text{dl-wcet}$.
- 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.

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.

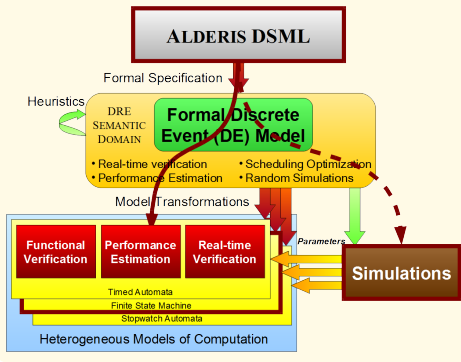
Model-based Analysis



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.

Model-based Analysis



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)*.

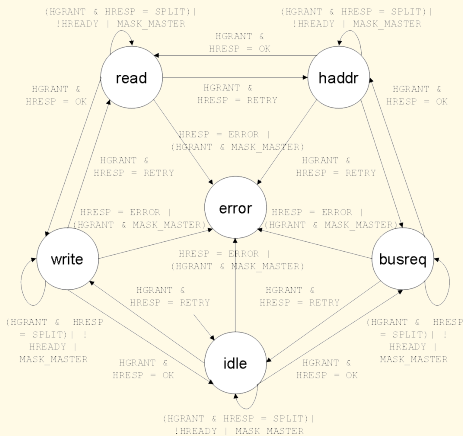
③ Real-time Verification

- Need to ensure real-time schedulability.
- End-to-end real-time constraints.
- Transaction-level *timed automata* model checking.

Modeling AMBA AHB

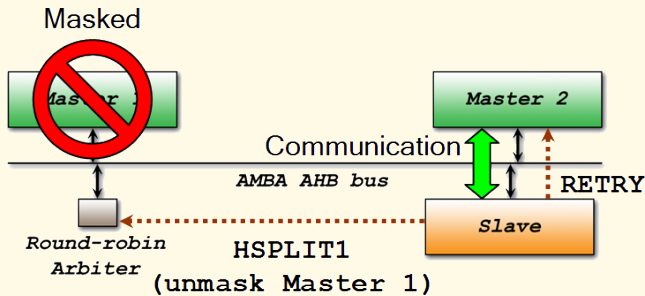
- Cycle-accurate *Finite State Machine (FSM)* model.
- Modeled generic master (6 states), slave (4 slaves), round-robin arbiter.
- Use FSM models for functional verification & performance estimation.
- Uncovered previously undocumented ambiguity.

AMBA AHB Master



Ambiguity in the AMBA AHB Protocol

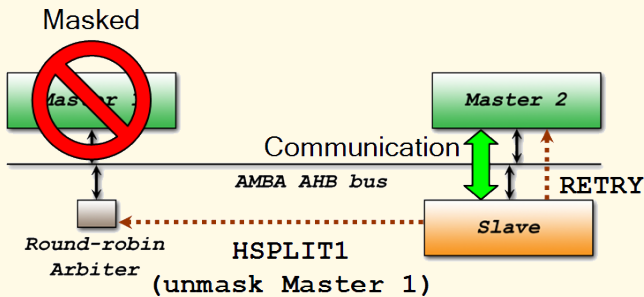
Possible Deadlock



- Slave has previously split Master₁.
- Slave in transaction with Master₂.
- Issue HSPLIT₁ while setting RETRY response.
- Should arbiter hold its state, or unmask Master₁?

Ambiguity in the AMBA AHB Protocol

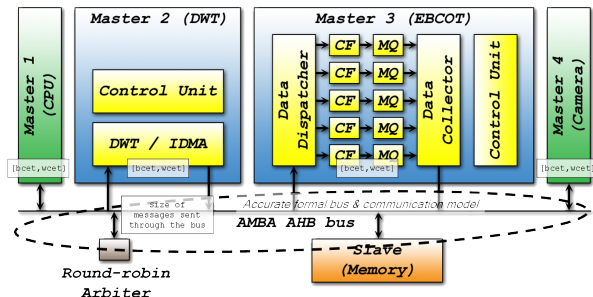
Possible Deadlock



- Slave has previously split Master₁.
- Slave in transaction with Master₂.
- Issue HSPLIT₁ while setting RETRY response.
- Should arbiter hold its state, or unmask Master₁?

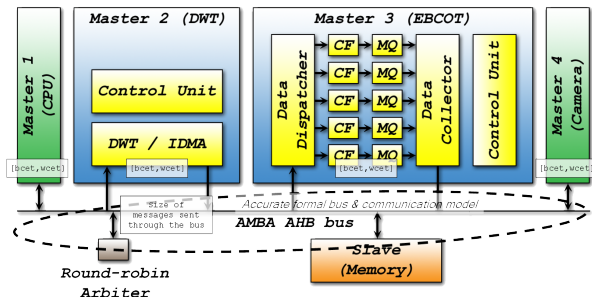
Combining Simulations and Model Checking for MPSoC Evaluation

- 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.



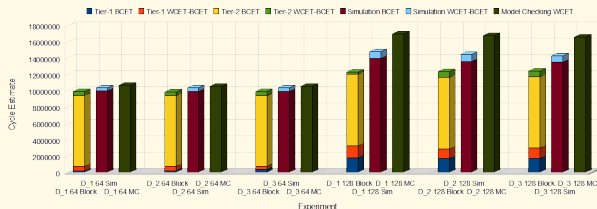
Combining Simulations and Model Checking for MPSoC Evaluation

- 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.

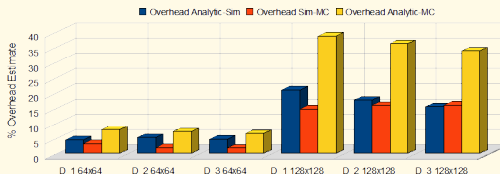


Comparing Simulations and Model Checking

Performance Estimates



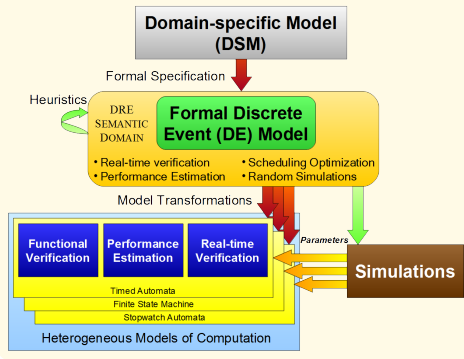
Communication Overhead Estimates



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.

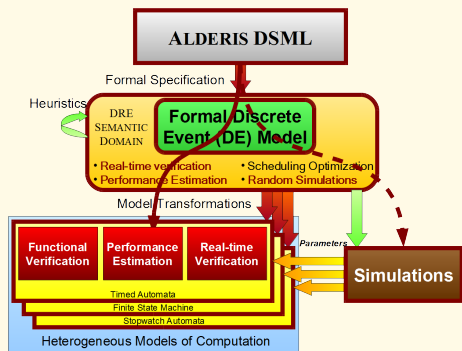
Model-based Analysis



Cross-abstraction Real-time Analysis of MPSoCs

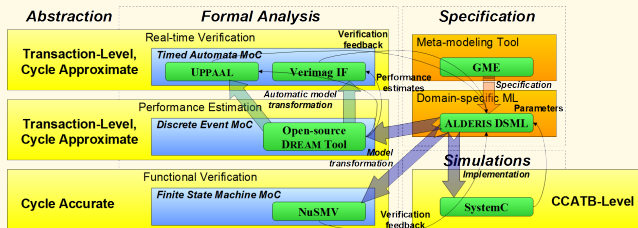
- *SystemC* implementation of the model used to obtain execution parameters.
- Perform functional verification on the FSM model by NuSMV.
- Use DREAM for performance estimation.
- Real-time verification of timed automata by UPPAAL or Verimag IF.
- Applied to networking router design.

Model-based Analysis



Analysis of Bus Matrix MPSoC Designs

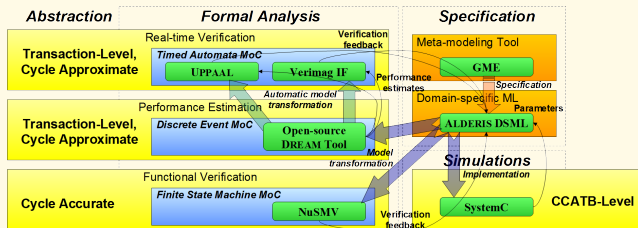
The CARTA Framework



- 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.

Analysis of Bus Matrix MPSoC Designs

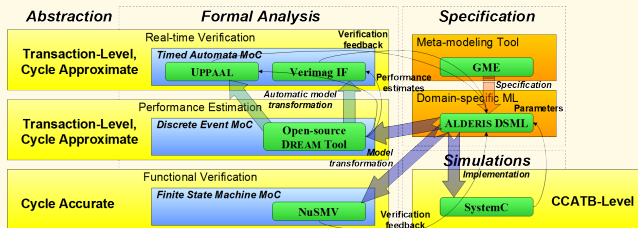
The CARTA Framework



- 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.

Analysis of Bus Matrix MPSoC Designs

The CARTA Framework

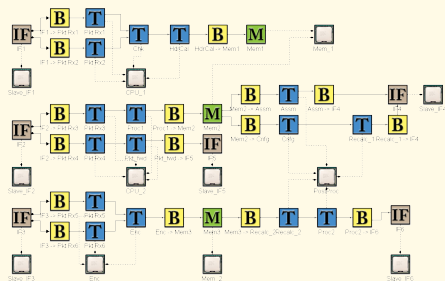


- Transaction-level simulations improve the analysis performance for MPSoCs.
- Could we trade off analysis performance and accuracy for model checking?
- Key contribution; show how analysis methods for DRE systems can be applied to MPSoCs.

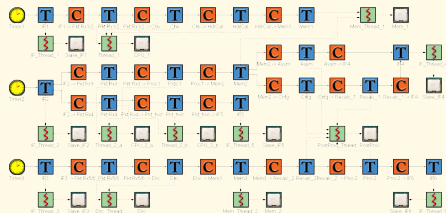
Transaction-level Modeling of Bus Matrix Interconnects

- Formal analysis is simplified in fully connected bus matrices.
- Point arbitration can be expressed as non-preemptive scheduling.

Networking Router Design

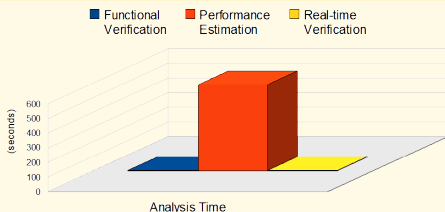
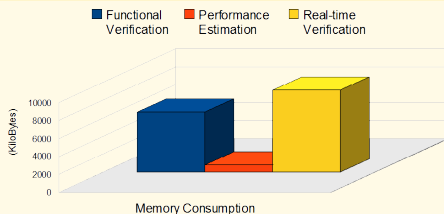


ALDERIS Model



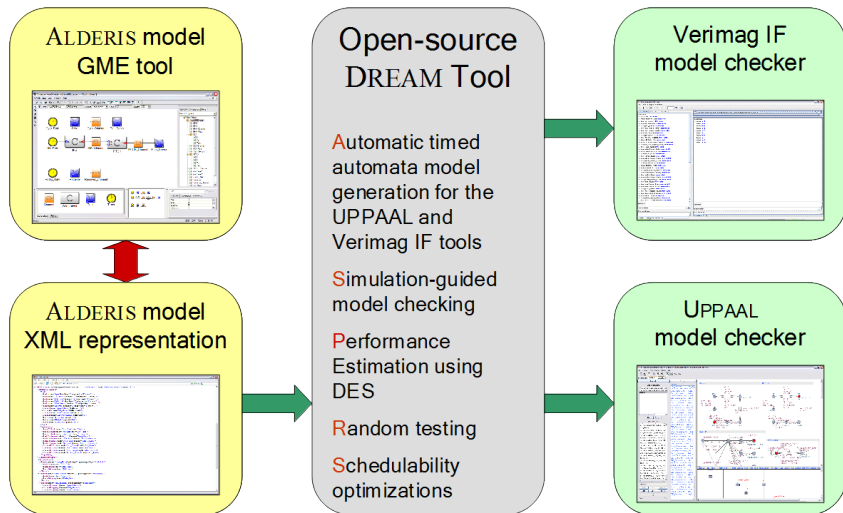
Applying Cross-abstraction Analysis to Networking Router Design

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



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].
- ④ Conservative Approximation of Preemptive Scheduling by Timed Automata [4].
- 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



G. Madl, S. Pasricha, Q. Zhu, L. A. D. Bathen, and N. Dutt.
Combining Transaction-level Simulations and Model Checking for
MPSoC Verification and Performance Evaluation.

ACM Transactions on Design Automation of Electronic Systems
(submitted for publication), 2009.



G. Madl, S. Pasricha, N. Dutt, and S. Abdelwahed.
Cross-abstraction Functional Verification and Performance Analysis of
Chip Multiprocessor Designs.

*IEEE Transactions on Industrial Informatics, Special Section on
Real-time and (Networked) Embedded Systems* (submitted for
publication), 2009.

Related Publications II

 G. Madl, S. Abdelwahed, and D. C. Schmidt.

Verifying Distributed Real-time Properties of Embedded Systems via Graph Transformations and Model Checking.

Real-Time Systems, 33:77–100, Jul 2006.

 G. Madl, N. Dutt, and S. Abdelwahed.

A Conservative Approximation Method for the Verification of Preemptive Scheduling using Timed Automata.

In *Proceedings of RTAS*, pages 255–264, April 2009.

 G. Madl, N. Dutt, and S. Abdelwahed.

Performance Estimation of Distributed Real-time Embedded Systems by Discrete Event Simulations.

In *Proceedings of EMSOFT*, October 2007.

Related Publications III



G. Madl, S. Pasricha, Q. Zhu, L. A. D. Bathen, and N. Dutt.

Formal Performance Evaluation of AMBA-based System-on-Chip Designs.

In *Proceedings of EMSOFT*, pages 311–320, October 2006.



G. Madl and S. Abdelwahed.

Model-based Analysis of Distributed Real-time Embedded System Composition.

In *Proceedings of EMSOFT*, 2005.



G. Madl, S. Abdelwahed, and G. Karsai.

Automatic Verification of Component-Based Real-Time CORBA Applications.

In *Proceedings of RTSS*, pages 231–240, December 2004.

Related Publications IV



G. Madl and N. Dutt.

Domain-specific Modeling of Power Aware Distributed Real-time Embedded Systems.

In Proceedings of the 6th Workshop on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), 2006.



G. Madl and N. Dutt.

Tutorial for the Open-source DREAM Tool.

In CECS Technical Report, 2006.

Questions?

- Open-source DREAM Tool:

`http://dre.sourceforge.net`

- ALDERIS Modeling Language:

`http://alderis.ics.uci.edu`