

Formal and Informal Methods for Multi-Core Design Space Exploration

Jean-Francois Kempf Olivier Lebeltel Oded Maler

QAPL, April 13, 2014

Introduction

Context

A motivating example

DeSpEx

Overview

DeSpEx: The Tool

Case Study

Conclusion

Context

Minalogic project ATHOLE

- ▶ Low-power multi-processors platform for embedded systems.
- ▶ Partners: STMicroelectronics, CEA Leti, Thales, CWS, Verimag.
- ▶ Verimag: High level modeling and analysis.

Contribution

- ▶ Development of a framework for modeling and analysis of embedded systems.

Motivation

Embedded Systems Design

- ▶ Several design choices both in hardware and software

Motivation

Embedded Systems Design

- ▶ Several design choices both in hardware and software
- ▶ Each has advantages according to different criteria:
 - ▶ Timing performance
 - ▶ Power consumption
 - ▶ Platform cost
 - ▶ ...

Motivation

Embedded Systems Design

- ▶ Several design choices both in hardware and software
- ▶ Each has advantages according to different criteria:
 - ▶ Timing performance
 - ▶ Power consumption
 - ▶ Platform cost
 - ▶ ...

Needs

- ▶ **Performance estimation as soon as possible**
 - ▶ evaluate quickly different trade-offs
- ▶ **Exploration and Analysis on a high level of abstraction.**

High-Level Performance Evaluation

Advantage

- ▶ Works at virtual level:
 - ▶ No need for a physical platform
 - ▶ No need for a complete implementation
- ▶ Models are simplified:
 - ▶ Performance analysis is tractable
 - ▶ Simulation and analysis are fast
- ▶ Evaluation of different alternatives can be done easily

High-Level Performance Evaluation

Advantage

- ▶ Works at virtual level:
 - ▶ No need for a physical platform
 - ▶ No need for a complete implementation
- ▶ Models are simplified:
 - ▶ Performance analysis is tractable
 - ▶ Simulation and analysis are fast
- ▶ Evaluation of different alternatives can be done easily

To compensate the lack of precision at this level of description:

- ▶ Increase the uncertainty margins
- ▶ Consider this uncertainty in the analysis

Uncertainty

Modeling uncertainty with timed automata

- ▶ Timing informations are modeled as intervals
- ▶ Exhaustive reachability analysis
- ▶ Analysis is **worst-case oriented** and sometimes intractable.

Uncertainty

Modeling uncertainty with timed automata

- ▶ Timing informations are modeled as intervals
- ▶ Exhaustive reachability analysis
- ▶ Analysis is **worst-case oriented** and sometimes intractable.

We may be more concerned about the **average performance**.

Uncertainty

Modeling uncertainty with timed automata

- ▶ Timing informations are modeled as intervals
- ▶ Exhaustive reachability analysis
- ▶ Analysis is **worst-case oriented** and sometimes intractable.

We may be more concerned about the **average performance**.

Modeling uncertainty probabilistically

- ▶ Duration Probabilistic Automata:
 - ▶ Timed automata with probabilistic durations
 - ▶ Discrete event simulation and statistical analysis
 - ▶ *Exact computation of expected termination time*

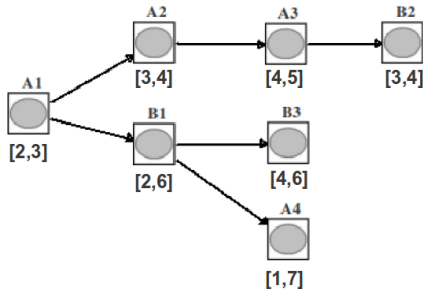
A motivating example

We show, with this example, the importance of considering the *uncertainty* in the analysis.

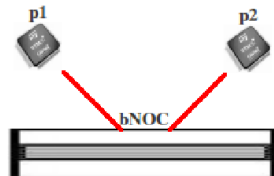
Outcome

- ▶ Timing analysis based exclusively on **worst case execution times** might not catch the **worst behavior**

Abstract Model

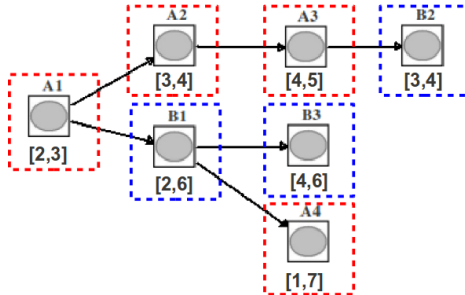


Task Graph

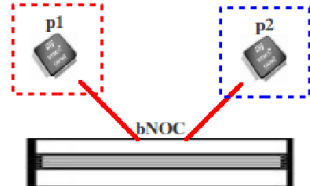


Architecture

Abstract Model



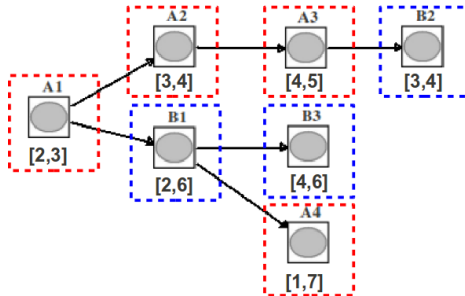
Task Graph



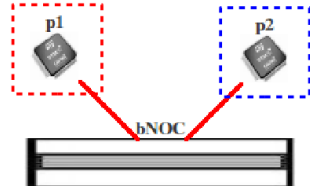
Architecture

- FIFO scheduling (non preemptive)

Abstract Model



Task Graph



Architecture

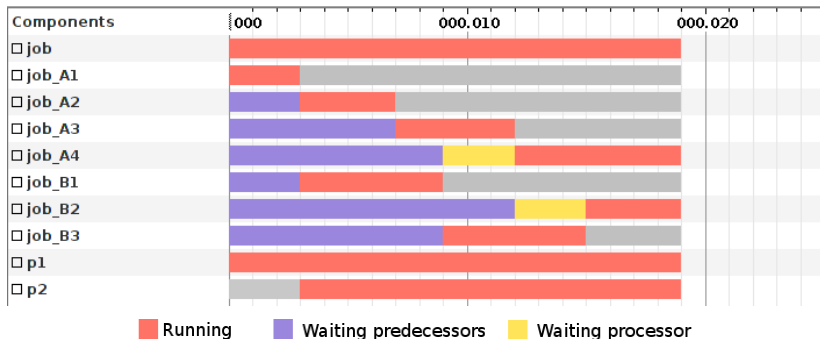
- ▶ FIFO scheduling (non preemptive)
- ▶ Question:
 - ▶ **What is the maximal response time of the job ?**

Corner-Case Analysis

- ▶ Naively, to get the maximal response time, one might do an analysis based on *worst-case execution time* for all tasks.

Corner-Case Analysis

- ▶ Naively, to get the maximal response time, one might do an analysis based on *worst-case execution time* for all tasks.



- ▶ Analysis gives a response time of 19 timeunits

Reachability Analysis with Uncertainty

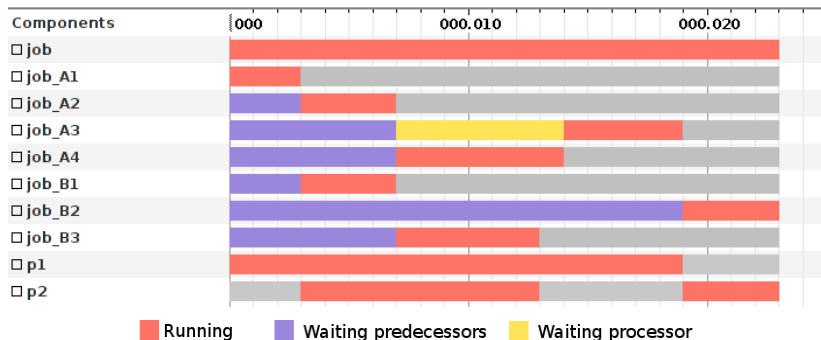
We use now timed automata reachability analysis:

- ▶ Explore all possible behaviors.
- ▶ Retrieve the execution trace leading to the worst response time.

Reachability Analysis with Uncertainty

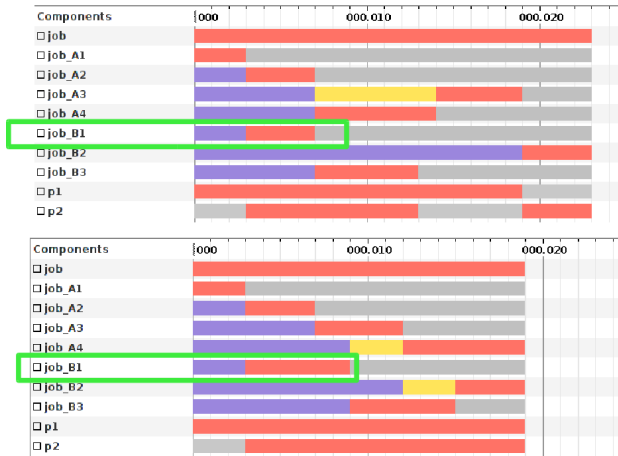
We use now timed automata reachability analysis:

- Explore all possible behaviors.
- Retrieve the execution trace leading to the worst response time.



- Analysis gives a response time of 23 timeunits

Explanations



B1 takes less time \Rightarrow A4 start before A3 (on critical path).

Quantitative Estimation

Uncertainty plays also an important role when we care more about the **average performance**

Quantitative Estimation

Uncertainty plays also an important role when we care more about the **average performance**

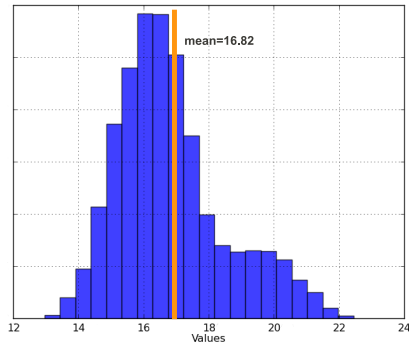
- ▶ Assumption: execution times are distributed uniformly.

Quantitative Estimation

Uncertainty plays also an important role when we care more about the **average performance**

- Assumption: execution times are distributed uniformly.

With simulation we get more quantitative information:



Motivation for combining formal and informal methods

- ▶ Analysis based on deterministic values (lower and upper) might give incorrect bounds on the global response time.

Motivation for combining formal and informal methods

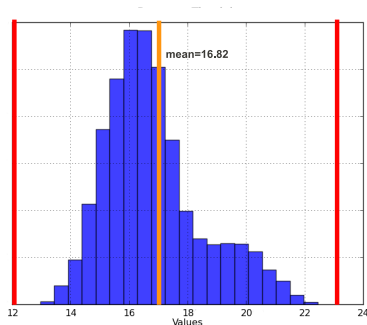
- ▶ Analysis based on deterministic values (lower and upper) might give incorrect bounds on the global response time.
- ▶ Timed automata reachability analysis gives us correct bounds but no quantitative information.

Motivation for combining formal and informal methods

- ▶ Analysis based on deterministic values (lower and upper) might give incorrect bounds on the global response time.
- ▶ Timed automata reachability analysis gives us correct bounds but no quantitative information.
- ▶ Stochastic simulation does not catch tight bounds but gives more quantitative information about average performance.

Motivation for combining formal and informal methods

- ▶ Analysis based on deterministic values (lower and upper) might give incorrect bounds on the global response time.
- ▶ Timed automata reachability analysis gives us correct bounds but no quantitative information.
- ▶ Stochastic simulation does not catch tight bounds but gives more quantitative information about average performance.



Introduction

Context

A motivating example

DeSpEX

Overview

DeSpEx: The Tool

Case Study

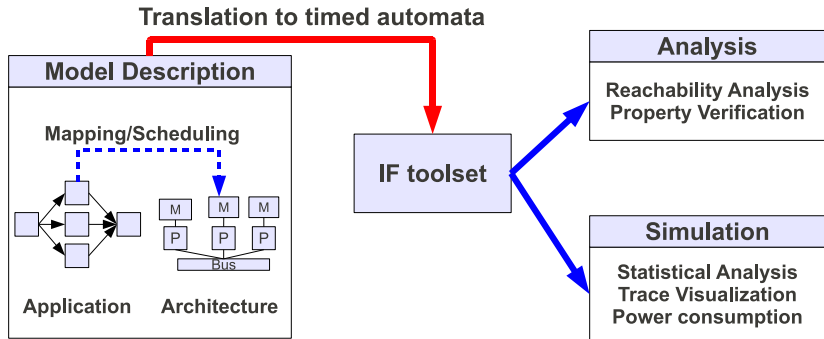
Conclusion

DeSpEx

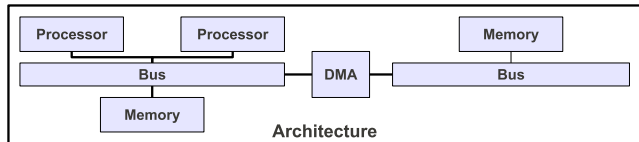
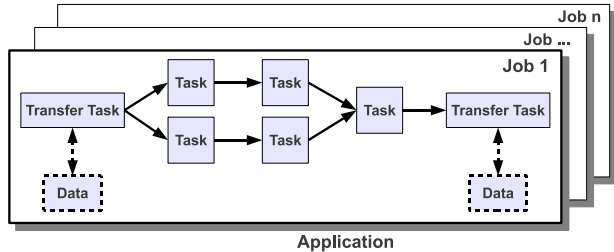
A framework for high level modeling and analysis

- ▶ Provide HW/SW designers with a framework for rapid design space exploration
- ▶ High level language for model description
- ▶ Formal semantics provided by *timed automata*
- ▶ Performance evaluation using formal methods and stochastic simulation

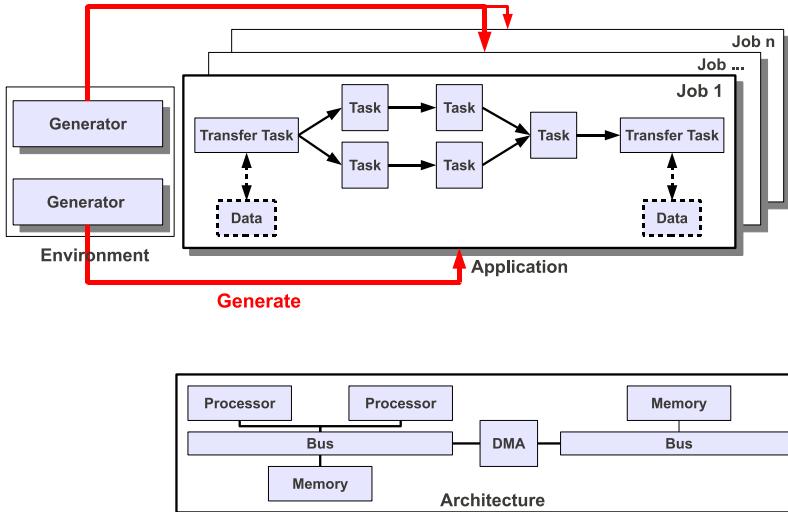
Framework Overview



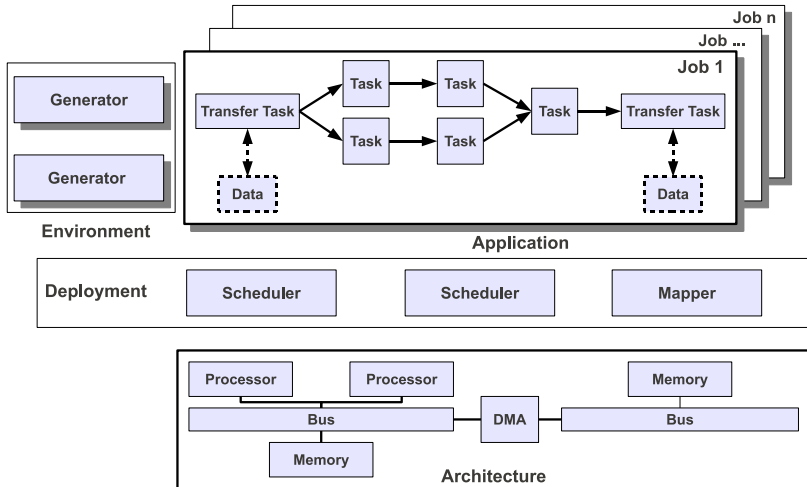
Model overview



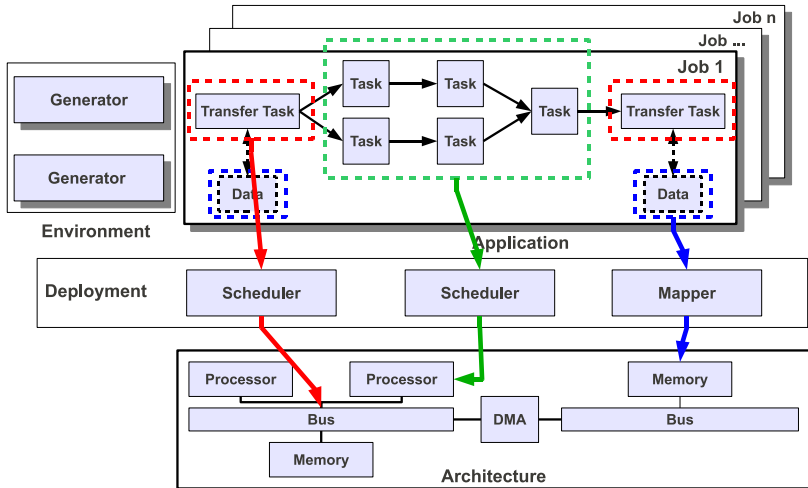
Model overview



Model overview



Model overview



Evaluation

- ▶ The aim of this modeling framework is to provide design space exploration for **performance evaluation**

Evaluation

- ▶ The aim of this modeling framework is to provide design space exploration for **performance evaluation**
- ▶ For each component we generate a corresponding timed automaton model

Evaluation

- ▶ The aim of this modeling framework is to provide design space exploration for **performance evaluation**
- ▶ For each component we generate a corresponding timed automaton model
- ▶ The composition of all automata yields a global timed automaton which captures the semantics of the system

Evaluation

- ▶ The aim of this modeling framework is to provide design space exploration for **performance evaluation**
- ▶ For each component we generate a corresponding timed automaton model
- ▶ The composition of all automata yields a global timed automaton which captures the semantics of the system
- ▶ All our analysis methods are based on a **unified semantic model** provided by timed automata

Introduction

Context

A motivating example

DeSpEX

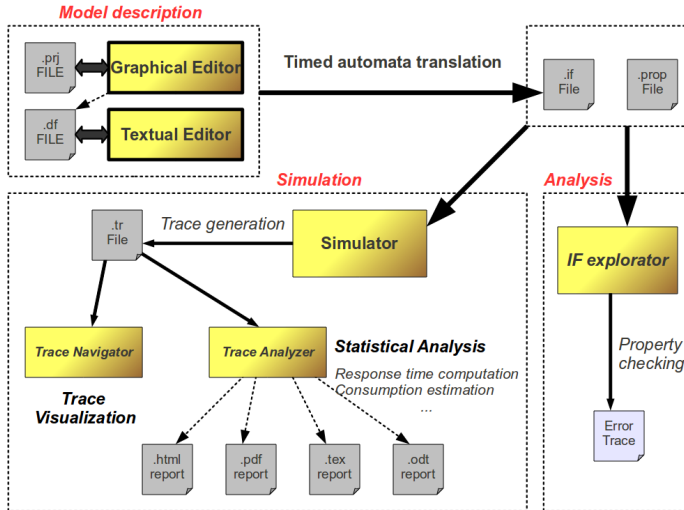
Overview

DeSpEX: The Tool

Case Study

Conclusion

Tool Overview



DeSpEx: Graphical Model Editor

The screenshot displays the DeSpEx Graphical Model Editor interface. The main workspace shows a task graph with a central task 'task_1Split' connected to a grid of 18 tasks (s2_1.1 to s2_5.6) which then converge into a 'task_Join'. The left palette lists components like Application, Job, Data, Environment (Generators), Architecture (Processors, Memory, DMA, Link, Bus), and System. The right side features a Parameter Editor for 'Component Task' (s2_4_3) and a Project Overview tree showing the hierarchy from Model to Application to Job to Architecture to Processors.

Parameter Editor

Name	Value
Component Task	
Name	s2_4_3
Weight	
Min	2482
Max	2584
Deadline	1000000

Project Overview

Type	Name
Model	thalesNoMem
Application	application
Job	job_0
Architecture	architecture
Processor	processor_0
Processor	processor_1
Processor	processor_2
Processor	processor_3
Processor	processor_4
Processor	processor_5
Processor	processor_6

Tool output
Project saved

DeSpEX: Graphical Model Editor

Architecture inspired by P2012 platform

The screenshot displays the DeSpEX Graphical Model Editor interface. The main workspace shows a system architecture diagram with the following components and connections:

- sharedMemory**: A memory component at the top.
- pe_0, pe_1, pe_2, pe_3**: Four processor components arranged in a row.
- externalMemory**: A memory component on the right.
- externalBus**: A bus component at the bottom right.
- dma_0**: A DMA component in the center.
- InterConnect**: A bus component at the bottom left.

Connections are shown as red lines: **sharedMemory** is connected to **pe_0, pe_1, pe_2, pe_3**. **pe_0, pe_1, pe_2, pe_3** are connected to **InterConnect**. **InterConnect** is connected to **dma_0**. **dma_0** is connected to **externalBus**. **externalBus** is connected to **externalMemory**.

The interface includes a **Palette** on the left with categories: Application, Environment, Architecture, Processor, ProcessorMF, Memory, DMA, Link, Bus, and System. The **Parameter Editor** on the right shows the configuration for the selected **Processor** component (**pe_3**):

Name	Value
Component	Processor
Name	pe_3
Speed	600
+ ChangeSpeedLatency	
+ TurnOffLatency	
+ TurnOnLatency	
Consumption	0
ConsumptionIdle	12
ConsumptionBusy	86

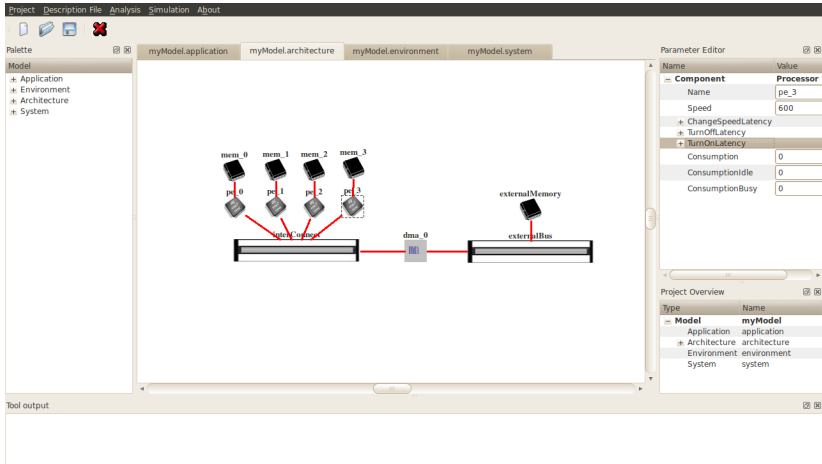
The **Project Overview** panel on the right shows the hierarchy of the model:

Type	Name
Model	myModel
Application	application
Architecture	architecture
+ DMA	dma_0
+ Bus	externalBus
+ Memory	externalMemory
+ Bus	interConnect
+ Link	link_0
+ Link	link_1
+ Link	link_10
+ Link	link_2
+ Link	link_3
+ Link	link_4

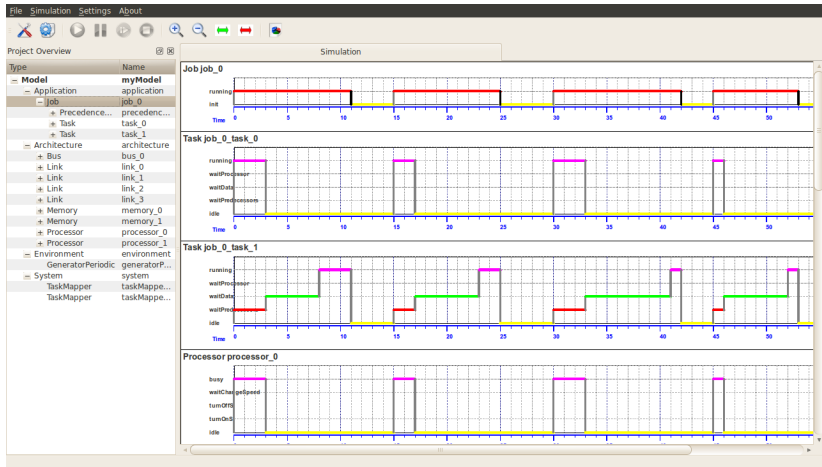
The **Tool output** panel at the bottom is currently empty.

DeSpEX: Graphical Model Editor

Architecture inspired by Cell platform



DeSpEx: Simulation and Trace Visualization



DeSpEX: Trace Visualization (Gantt Chart)



Performance Evaluation

Reachability Analysis

- ▶ Check whether some properties are satisfied or not
- ▶ In case of property violation generates an error trace, viewable with the GUI

Performance Evaluation

Reachability Analysis

- ▶ Check whether some properties are satisfied or not
- ▶ In case of property violation generates an error trace, viewable with the GUI

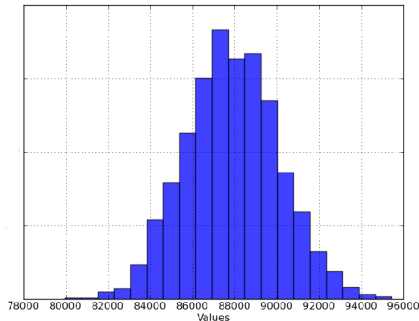
Stochastic Simulation

- ▶ Depending on the size of the model it may be difficult (sometimes impossible) to perform reachability analysis
- ▶ Timed automata are used to perform discrete event simulation:
 - ▶ **Semantic model is the same** as for reachability analysis
 - ▶ Randomized reachability exploration
 - ▶ We restrict ourself to bounded uncertainty

Performance Evaluation

Trace Analysis

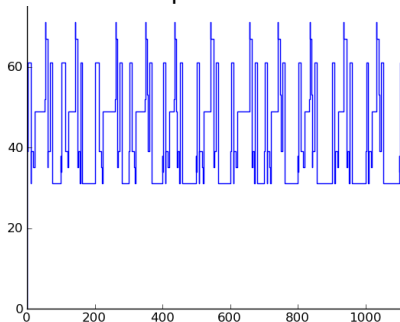
- ▶ Stochastic simulation generates timed traces
- ▶ We can retrieve quantitative informations with trace analysis:
 - ▶ Response time distribution of job



Performance Evaluation

Trace Analysis

- ▶ Stochastic simulation generates timed traces
- ▶ We can retrieve quantitative informations with trace analysis:
 - ▶ Response time distribution of job
 - ▶ Power consumption estimation



Introduction

Context

A motivating example

DESpEX

Overview

DeSpEx: The Tool

Case Study

Conclusion

Case Study

Video Processing on P2012

Goal

- Demonstrate how DeSpEx can be used to solve realistic problems in design space exploration

Case Study

Video Processing on P2012

Goal

- ▶ Demonstrate how DeSpEx can be used to solve realistic problems in design space exploration
- ▶ Quantify the performance differences between different design choices

Case Study

Video Processing on P2012

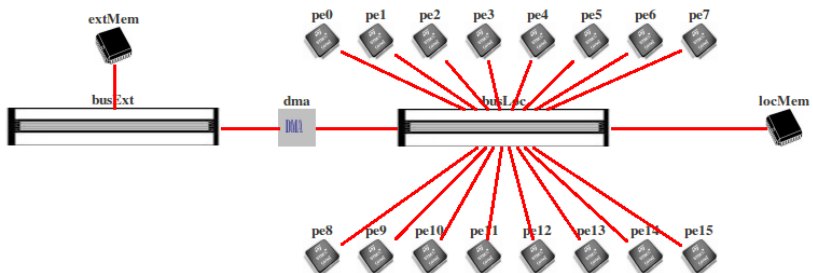
Goal

- ▶ Demonstrate how DeSpEx can be used to solve realistic problems in design space exploration
- ▶ Quantify the performance differences between different design choices
- ▶ Represent available cost/performance trade-offs.

Video Processing on P2012

Architecture Abstraction

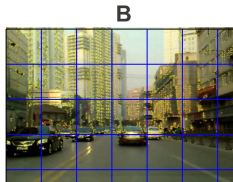
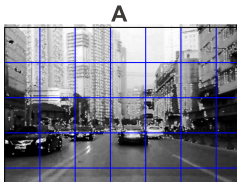
- ▶ P2012 is a many-core computing fabric based on multiple clusters
- ▶ We restrict our models to one cluster



Video Processing on P2012

Application

- ▶ Augmented reality application called FAST (Features from Accelerated Segment Test)
- ▶ Corner detection method
- ▶ Algorithm consists in computing the detection on a chunk of an image



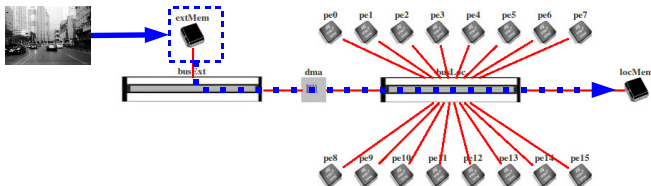
Algorithm

```
ForEach A[i][j]  
Do  
    B[i][j] := compute(A[i][j])  
EndDo
```

Video Processing on P2012

From an architectural point of view:

- ▶ The image resides initially in the off-chip memory
- ▶ Needs to be brought to local memory
- ▶ Then dispatched to processors for execution



Constraints:

- ▶ The whole image does not fit into the local memory
- ▶ Several alternatives for its splitting and transfer to local memories

Video Processing on P2012

At early design stage:

- ▶ no physical platform
- ▶ no complete implementation

Video Processing on P2012

At early design stage:

- ▶ no physical platform
- ▶ no complete implementation

Compare different design alternatives with DeSpEx

- ▶ How many processors should we use ? 1,2,4,8 or 16
- ▶ At which processor frequency? 200, 400 or 600 MHz

Video Processing on P2012

At early design stage:

- ▶ no physical platform
- ▶ no complete implementation

Compare different design alternatives with DeSpEx

- ▶ How many processors should we use ? 1,2,4,8 or **16**
- ▶ At which processor frequency? 200, 400 or **600** MHz

Depends on different criteria for processing an image:

- ▶ Response time

Video Processing on P2012

At early design stage:

- ▶ no physical platform
- ▶ no complete implementation

Compare different design alternatives with DeSpEx

- ▶ How many processors should we use ? **1**,2,4,8 or 16
- ▶ At which processor frequency? **200**, 400 or 600 MHz

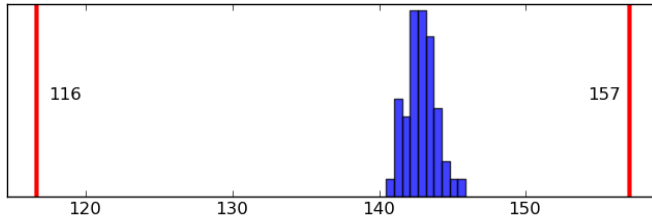
Depends on different criteria for processing an image:

- ▶ Response time
- ▶ Power consumption of the platform

Video Processing on P2012: Evaluation

Worst-Case vs Statistics

- ▶ Compare results from simulation and reachability analysis:



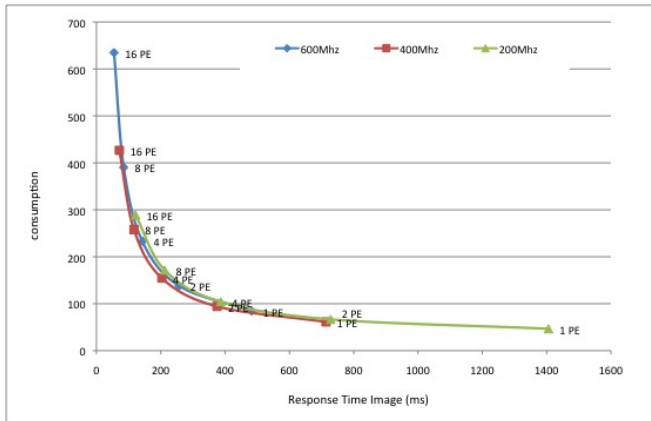
Response time of processing one image.

- ▶ Bands treatment
- ▶ 4 processors (600 MHz)

Video Processing on P2012: Evaluation

Power Consumption

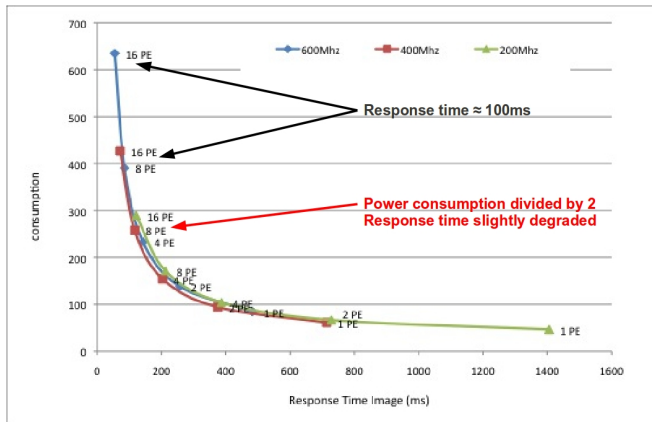
We compare different configuration of the platform to get the trade-offs between response time and power consumption.



Video Processing on P2012: Evaluation

Power Consumption

We compare different configuration of the platform to get the trade-offs between response time and power consumption.



Introduction

Context

A motivating example

DeSpEx

Overview

DeSpEx: The Tool

Case Study

Conclusion

Conclusion and future work

- ▶ We provide a tool-supported framework for Design-Space Exploration based on abstract model
- ▶ The framework and analysis techniques have been implemented into an extensible toolset: DeSpEx

Conclusion and future work

- ▶ We provide a tool-supported framework for Design-Space Exploration based on abstract model
- ▶ The framework and analysis techniques have been implemented into an extensible toolset: DeSpEx

Future work:

- ▶ Enrich the component library
- ▶ Integration with other formalism such as Synchronous Data-Flow
- ▶ Integration of a module for piecewise analytic computation of expected performance

Conclusion and future work

- ▶ We provide a tool-supported framework for Design-Space Exploration based on abstract model
- ▶ The framework and analysis techniques have been implemented into an extensible toolset: DeSpEx

Future work:

- ▶ Enrich the component library
- ▶ Integration with other formalism such as Synchronous Data-Flow
- ▶ Integration of a module for piecewise analytic computation of expected performance

Thank you for your attention