Formal Verification and Performance Analysis of Embedded Systems

Kim G. Larsen
CISS, Aalborg U., DK

Michael R. Hansen
IMM, Technical U. of Denmark, DK
DaNES Challenges

- Application
  - Stepw. Refinem.
    - SW API / OS
      - HW
        - network

Environment
DaNES Challenges

Selfdiagnosis & -repair

Model Driven Component Based Development

Embedded & Distributed Control

Test & Verification

Application

Informatik og Matematisk Modellering

IO TECHNOLOGIES

ARTIST2 WS: Tool Platforms for ES Modelling, Analysis & Validation

Kim Larsen [3]
Outline

- Validation and Performance Analysis in UPPAAL / Kim G. Larsen
- Formalising the ARTS MPSoC Model in UPPAAL / Michael R. Hansen
- Discussion Points / KGL & MRH
Validation and Performance Analysis in UPPAAL
UPPAAL

Graphical Design Tool

- timed automata =
  - state machines
  +
  - clocks
- communication
- datatypes
- user defined functions
- cost variable
UPPAAL

Graphical Simulator
- visualization and recording
- inexpensive fault detection
- inspection of error traces
- Message Sequence Charts
- (Gantt Charts)
Verifier

- Exhaustive & automatic checking of requirements
- .. including validating, safety, liveness, bounded liveness and response properties
- .. generation of debugging information for visualisation in simulator.
- Optimal scheduling for cost models
“Impact”

Google:

UPPAAL: 134,000
SPIN Verifier: 242,000
nuSMV: 57,700

> 1.500
Google Scholar Citations
(Rhapsody/Esterel < 3.500)
Impact

Academic Courses @

DTU, MCI, IT-U (DK)
Chalmers,
Linköping, Lund,
Chalmers,
Mälardalarn (S)
Nijmegen, Twente, CWI (NL)
Upenn, Northumbria (US)
Braunschweig,
Oldenborg, Marktoberdorf (D)
Tsinghua, Shanghai, ISS, NUS (Asia)
Impact

Company Downloads
Mecel
Jet
Symantec
SRI
Relogic
Realwork
NASA
Verified Systems
Microsoft
ABB
Airbus
PSA
Saab
Siemens
Volvo
Lucent Technologies
Systematic

ARTIST2 WS: Tool Platforms for ES Modelling, Analysis & validation
Kim Larsen [11]
Tutorials Given @
Estonian School (01)
IPA Fall Days (01)
FTRTFT (02)
CPN (02)
SFM (02)
MOVEP (02)
DISC School
MOVEP (04)
PRISE (04)
PDMC (05)
ARTIST2 (05)
EMSOFT (05)
RTSS (05)
TECS week (06)
TAROT (06)
ARTS (06)
GLOBAN (06)
ARTIST ASIAN SCH (07)
......
Timed Automata

[Alur & Dill’89]

Resource

Reset

Invariant

Synchronization

Guard

Idle

use?

x:=0

lnUse

x<=7

done!

x>=4

Resource
Timed Automata

Resource

Synchronization

Reset

Invariant

Guard

Semantics:
( Idle, x=0 )
→ ( Idle, x=2.5 ) d(2.5)
→ ( InUse, x=0 ) use?
→ ( InUse, x=5 ) d(5)
→ ( Idle, x=5 ) done!
→ ( Idle, x=8 ) d(3)
→ ( InUse, x=0 ) use?

[Alur & Dill’89]
Composition

Resource

Idle

use?

x:=0

InUse

x<=B

done!

x>=B

Synchronization

Init

use!

B:=6

Using

done?

Task

Done

Shared variable
Semantics:

\[(\text{Idle}, \text{Init}, B=0, x=0)\]
\[\rightarrow (\text{Idle}, \text{Init}, B=0, x=3.1415) \quad \text{d}(3.1415)\]
\[\rightarrow (\text{InUse}, \text{Using}, B=6, x=0) \quad \text{use}\]
\[\rightarrow (\text{InUse}, \text{Using}, B=6, x=6) \quad \text{d}(6)\]
\[\rightarrow (\text{Idle}, \text{Done}, B=6, x=6) \quad \text{done}\]
Task Graph Scheduling

Optimal Static Task Scheduling

- Task $P = \{P_1, \ldots, P_m\}$
- Machines $M = \{M_1, \ldots, M_n\}$
- Duration $\Delta : (P \times M) \rightarrow \mathbb{N}_\infty$
- $< :$ p.o. on $P$ (pred.)

- A task can be executed only if all predecessors have completed
- Each machine can process at most one task at a time
- Task cannot be preempted.

$P_2$ $P_1$

$P_6$ $P_3$ $P_4$

$P_7$ $P_5$

$M = \{M_1, M_2\}$

$16,10$ $2,3$

$2,3$ $6,6$ $10,16$

$2,2$ $8,2$
Task Graph Scheduling

Optimal Static Task Scheduling

- Task $P = \{P_1, \ldots, P_m\}$
- Machines $M = \{M_1, \ldots, M_n\}$
- Duration $\Delta : (P \times M) \rightarrow \mathbb{N}_\infty$
- $<$: p.o. on $P$ (pred.)

$M = \{M_1, M_2\}$
### Experimental Results

<table>
<thead>
<tr>
<th>name</th>
<th>#tasks</th>
<th>#chains</th>
<th># machines</th>
<th>optimal</th>
<th>TA</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>437</td>
<td>125</td>
<td>4</td>
<td>1178</td>
<td>1182</td>
</tr>
<tr>
<td>000</td>
<td>452</td>
<td>43</td>
<td>20</td>
<td>537</td>
<td>537</td>
</tr>
<tr>
<td>018</td>
<td>730</td>
<td>175</td>
<td>10</td>
<td>700</td>
<td>704</td>
</tr>
<tr>
<td>074</td>
<td>1007</td>
<td>66</td>
<td>12</td>
<td>891</td>
<td>894</td>
</tr>
<tr>
<td>021</td>
<td>1145</td>
<td>88</td>
<td>20</td>
<td>605</td>
<td>612</td>
</tr>
<tr>
<td>228</td>
<td>1187</td>
<td>293</td>
<td>8</td>
<td>1570</td>
<td>1574</td>
</tr>
<tr>
<td>071</td>
<td>1193</td>
<td>124</td>
<td>20</td>
<td>629</td>
<td>634</td>
</tr>
<tr>
<td>271</td>
<td>1348</td>
<td>127</td>
<td>12</td>
<td>1163</td>
<td>1164</td>
</tr>
<tr>
<td>237</td>
<td>1566</td>
<td>152</td>
<td>12</td>
<td>1340</td>
<td>1342</td>
</tr>
<tr>
<td>231</td>
<td>1664</td>
<td>101</td>
<td>16</td>
<td>t.o.</td>
<td>1137</td>
</tr>
<tr>
<td>235</td>
<td>1782</td>
<td>218</td>
<td>16</td>
<td>t.o.</td>
<td>1150</td>
</tr>
<tr>
<td>233</td>
<td>1980</td>
<td>207</td>
<td>19</td>
<td>1118</td>
<td>1121</td>
</tr>
<tr>
<td>294</td>
<td>2014</td>
<td>141</td>
<td>17</td>
<td>1257</td>
<td>1261</td>
</tr>
<tr>
<td>295</td>
<td>2168</td>
<td>965</td>
<td>18</td>
<td>1318</td>
<td>1322</td>
</tr>
<tr>
<td>292</td>
<td>2333</td>
<td>318</td>
<td>3</td>
<td>8009</td>
<td>8009</td>
</tr>
<tr>
<td>298</td>
<td>2399</td>
<td>303</td>
<td>10</td>
<td>2471</td>
<td>2473</td>
</tr>
</tbody>
</table>

Abbeddaïm, Kerbaa, Maler
Optimal Task Graph Scheduling
Power-Optimality

- Energy-rates: \( C : M \rightarrow N \)

![Diagram of task graph scheduling with nodes and edges labeled with numbers and power ratings.](image)
Dynamic Voltage Scaling
Task Scheduling

utilization of CPU

- $P(i)$, $[E(i), L(i)]$, ..: period or earliest/latest arrival or .. for $T_i$
- $C(i)$: execution time for $T_i$
- $D(i)$: deadline for $T_i$

Scheduler

$T_1$, $T_2$, ..,$T_n$ ready done

$T_2$ is running

$\{T_4, T_1, T_3\}$ ready ordered according to some given priority:
(e.g. Fixed Priority, Earliest Deadline,..)
Modeling Task

Scheduler

- $T_1$: ready, done
- $T_2$: stop, run
- $T_n$: stop, run

Graph:
- Idle: $t \leq L[id]
- E[\text{id}]
- Ready: $t > E[\text{id}]
- done!
- Running: $e = \text{id}$, $t=0$, $e = \text{id}$
- Error: $t > D[\text{id}]$
- $ax = C[\text{id}]$

ARTIST2 WS: Tool Platforms for ES Modelling, Analysis & Validation

Kim Larsen [23]
Modeling Scheduler
void add(id_t element)
{
    int i=len;
    int temp=0;
    if(len==0)
    {
        list[len]=e;
        len=1;
    }
    else
    {
        list[len]=e;
        i=len;
        len=len+1;
        while (i>1 && P[list[i]]> P[list[i-1]])
        {
            temp=list[i-1];
            list[i-1]=list[i];
            list[i]=temp;
            i=i-1;
        }
    }
}
Schedulability = Safety Property

\[ \neg (\text{Task0.Error or Task1.Error or ...}) \]

May be extended with preemption

\[ \text{A} \]

Kim Larsen [26]
Energy Optimal Scheduling

Using Priced Timed Automata

Scheduler

"Choose" Scaling/Cost (Freq/Voltage)

F := ?? ; V := ??

T_1
ready
done

T_2
stop
run

T_n

Scheduler

F := ?? ; V := ??
Cost Optimal Scheduling = Optimal Infinite Path

Value of path $\sigma$: $\text{val}(\sigma) = \lim_{n \to \infty} \frac{c_n}{t_n}$

Optimal Schedule $\sigma^*$: $\text{val}(\sigma^*) = \inf_{\sigma} \text{val}(\sigma)$
Cost Optimal Scheduling = Optimal Infinite Path

\[ \text{Value of path } \sigma : \quad \text{val}(\sigma) = \lim_{n \to \infty} \frac{c_n}{t_n} \]

Optimal Schedule \( \sigma^* \): \quad \text{val}(\sigma^*) = \inf_\sigma \text{val}(\sigma)

\( \neg (\text{Task0.Err or Task1.Err or } \ldots) \)

THEOREM: \( \sigma^* \) is computable

Bouyer, Brinksma, Larsen HSCC’04
UPPAAL

Home

UPPAAL is an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata, extended with data types (bounded integers, arrays, etc.).

The tool is developed in collaboration between the Department of Information Technology at Uppsala University, Sweden and the Department of Computer Science at Aalborg University in Denmark.

Download

News: We are proud to officially release UPPAAL 4.0.2 (Aug 7, 2006), available from the Download page. The 4.0 release is the result of over 2.5 years of development, and many new features and improvements are introduced (see also this release note and the web help section new features). To support models created in previous versions of UPPAAL, version 4.0 can convert most old models directly from the GUI (alternatively it can be run in 3.4 compatibility mode by defining the environment variable UPPAAL_OLD_SYNTAX, see also item 2 of the FAQ).

License

The UPPAAL tool is free for non-profit applications. For information about commercial licenses, please email sales(at)uppaal(dot)com.

To find out more about UPPAAL, read this short introduction. Further information may be found at this web site in the pages About, Documentation, Download, and Examples.

Mailing Lists

UPPAAL has an open discussion forum group at YahooGroups intended for users of the tool. To join or post to the forum, please refer to the information at the discussion forum page. Bugs should be reported using the bug tracking system. To email the development team directly, please use uppaal(at)list(dot)as(dot)se.
Formalizing the ARTS MPSoC Model in UPPAAL

Jan Madsen and Michael R. Hansen

Embedded Systems Engineering Group

Informatics and Mathematical Modeling
Technical University of Denmark
Motivation

CELL processor
Motivation

Application model

System platform

System-level design tasks

Model of system implementation

Madsen & Hansen [33]
**ARTS objectives**

- System-level modeling framework
- Bridging,
  - Application
  - RTOS
  - Execution platform
    - Processing elements
    - NoC
- Supporting
  - System-level analysis
  - Early design space exploration

**Cross-layer optimization**

Model of system implementation

Madsen & Hansen [34]
Formalizing ARTS

\[
\text{System} = \text{Application} \ || \ \text{ExecutionPlatform}
\]

\[
\text{Application} = \big|\big|_{i=1}^{n} \tau_i
\]

\[
\text{ExecutionPlatform} = \big|\big|_{j=1}^{m} \text{pe}_j
\]
Timed Automata for a task
MOVES: Hiding UPPAAL!

System-level design tasks

Model of system implementation

Application model

System platform
MOVES

Required specification

\[ E<>\text{missedDeadline} \]
\[ E<>\text{totalCostUsed(Memory)} \geq 23 \]
\[ E<>\text{totalCostUsed(Energy)} \geq 15 \]

Model checking

Model of system implementation

\[ \text{L}_1 \rightarrow \text{R}_1 \rightarrow \text{L}_2 \rightarrow \text{R}_2 \rightarrow \text{L}_3 \rightarrow \text{R}_3 \]

DATE'07 Workshop, April 20, 2007

Madsen & Hansen [38]
Example: MPSoC specification

![Diagram of MPSoC specification]

- Application
  - $\tau_1$, $\tau_2$, $\tau_3$, $\tau_4$

- Task
  - $\pi$, $\delta$, $\alpha$
  - $\tau_1$: $\pi = 10$, $\delta = 10$, $\alpha = 0$
  - $\tau_2$: $\pi = 10$, $\delta = 8$, $\alpha = 0$
  - $\tau_3$: $\pi = 10$, $\delta = 10$, $\alpha = 0$
  - $\tau_4$: $\pi = 10$, $\delta = 5$, $\alpha = 0$

- Execution Platform
  - $os_1$, $os_2$, $os_3$

- $PE_f$, $PE_1$, $PE_2$

- $p_{in}$, $1$, $3$

- $PE_f$: $f = 1$ MHz, $os = RM$

- Task
  - $bcet$, $w cet$, $sm$, $dm$, $pw$
  - $\tau_1$: $bcet = 1$, $w cet = 3$, $sm = 1$, $dm = 3$, $pw = 5$
  - $\tau_2$: $bcet = 2$, $w cet = 2$, $sm = 1$, $dm = 7$, $pw = 5$
  - $\tau_3$: $bcet = 2$, $w cet = 2$, $sm = 1$, $dm = 6$, $pw = 10$
  - $\tau_4$: $bcet = 1$, $w cet = 1$, $sm = 2$, $dm = 9$, $pw = 10$

- $\pi$: period [s]
- $\delta$: deadline [s]
- $\alpha$: offset [s]
- $bcet$: best case execution time [cycles]
- $w cet$: worst case execution time [cycles]
- $sm$: static memory [Byte]
- $dm$: dynamic memory [Byte]
- $pw$: power [nW]
- $f$: frequency [Hz]
- $os$: operating system [{FP, RM, DM, EDF}]

DATE’07 Workshop, April 20, 2007

Madsen & Hansen [39]
Specifying the application

Task $t_1 = \text{new Task}(1, 3, 10, 10, 0, 1);$
Task $t_2 = \text{new Task}(2, 2, 8, 10, 0, 2);$
Task $t_3 = \text{new Task}(2, 2, 10, 10, 0, 3);$
Task $t_4 = \text{new Task}(1, 1, 5, 10, 0, 4);$

<table>
<thead>
<tr>
<th>Task</th>
<th>$\pi$</th>
<th>$\delta$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>10</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>10</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

$\text{PE}_i f = 1 \text{ MHz}, \text{ os } = \text{ RM}$

<table>
<thead>
<tr>
<th>Task</th>
<th>$\text{bcet}$</th>
<th>$\text{w cet}$</th>
<th>$\text{sm}$</th>
<th>$\text{dm}$</th>
<th>$\text{pw}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>
Execution platform

Processor p1 = new Processor(1, Processor.RM);
Processor p2 = new Processor(1, Processor.RM);
Processor p3 = new Processor(1, Processor.RM);
Processor pm = new Processor(1, Processor.RM);
Resource r1 = new Resource();

\[ \text{Execution Platform} \]

\[ \text{PE}_i f = 1 \text{ MHz, os = RM} \]

\[
\begin{array}{c|cccc}
\text{Task} & \text{bcet} & \text{w cet} & \text{sm} & \text{dm} & \text{pw} \\
\hline
\tau_1 & 1 & 3 & 1 & 3 & 5 \\
\tau_2 & 2 & 2 & 1 & 7 & 5 \\
\tau_3 & 2 & 2 & 1 & 6 & 10 \\
\tau_4 & 1 & 1 & 2 & 9 & 10 \\
\end{array}
\]
Task[][] tasks =
    {{t1}, {t2, t3}, {t4}, {tm1, tm2}};

Task tm1 = new Task(3, 3, 10, 5);
Task tm2 = new Task(1, 1, 10, 6);

apps.useResource(tm1, r1);
apps.useResource(tm2, r1);

apps.addDep(t1, tm1);
apps.addDep(tm1, t2);
apps.addDep(t3, tm2);
apps.addDep(tm2, t4);
\[ e(\tau_1) = [1:3] \]
Handling realistic applications?

Smart phone:

[Application from Marcus Schmitz, TU Linkoping]
Smart phone

- Tasks: 114
- Deadlines: [0.02: 0.5] sec
- Execution: [52 : 266.687] cycles
- Platform:
  - 6 processors, 25 MHz
  - 1 bus
- Verified in 2.5 hours!
- Using a granularity of 400 cycles
Acknowledgements

MOVES

- Aske Brekling
- Jens Ellebæk
- Kristian S. Knudsen

ARTS

- Shankar Mahadevan
- Kehuai Wu
- Kashif Virk
- Michael Storgaard
- Mercury Gonzalez
Discussion Points

- **Accuracy versus scalability**
  - UPPAAL & SymTA/S (Network Calculus)

- **Verification versus simulation**
  - UPPAAL & ARTS (System C)
  - UPPAAL & Ptolemy
  - UPPAAL & TrueTime

- Efficient verification methods for **optimal infinite scheduling** for PTA.
  - Negative rates
  - Discounting
Collaborators

@UPPsala
- Wang Yi
- Paul Pettersson
- John Håkansson
- Anders Hessel
- Pavel Krcal
- Leonid Mokrushin
- Shi Xiaochun

@AALborg
- Kim G Larsen
- Gerd Behrman
- Arne Skou
- Brian Nielsen
- Alexandre David
- Jacob I. Rasmussen
- Marius Mikucionis
- Thomas Chatain

@Elsewhere
- Emmanuel Fleury, Didier Lime, Johan Bengtsson, Fredrik Larsson, Kåre J Kristoffersen, Tobias Amnell, Thomas Hune, Oliver Möller, Elena Fersman, Carsten Weise, David Griffioen, Ansgar Fehnker, Frits Vandraager, Theo Ruys, Pedro D’Argenio, J-P Katoen, Jan Tretmans, Judi Romijn, Ed Brinksma, Martijn Hendriks, Klaus Havelund, Franck Cassez, Magnus Lindahl, Francois Laroussinie, Patricia Bouyer, Augusto Burgueno, H. Bowmann, D. Latella, M. Massink, G. Faconti, Kristina Lundqvist, Lars Asplund, Justin Pearson...