Energy-optimization for Dataflow Applications using Timed Automata

Jaco van de Pol, 18 Aug 2016

Joint work with
Waheed Ahmad and Mariëlle Stoelinga
- Over 10 years history of multi-core architectures: PC, mobile
- HD video playback, web browsing, 3D gaming, 3D interfaces
- By 2015, video causes 2/3 of mobile data traffic (Cisco, 2011)
Over 10 years history of multi-core architectures: PC, mobile
HD video playback, web browsing, 3D gaming, 3D interfaces
By 2015, video causes 2/3 of mobile data traffic (Cisco, 2011)

Multimedia services are energy-hungry
How to get quality & performance within tight energy bounds?
FP7 Sensation: Self-energy-supporting autonomous computing
# Table of Contents

1. **Scheduling and Mapping on Multi-core Hardware**
   - Modeling: Synchronous Dataflow and Hardware Platform
   - Analysis: Timed Automata and **Uppaal**

2. **Energy Optimization by Power Management**
   - Modeling: DPM, DVFS, VFI in Hardware Platform Model
   - Analysis: Priced Timed Automata and **Uppaal Cora**

3. **Extension 1: Battery Management**
   - Modeling: Kinetic Battery Model
   - Analysis: Hybrid Automata and **Uppaal SMC**

4. **Extension 2: Handling Uncertainty by Adaptation**
   - Modeling: Execution Time is Stochastic
   - Analysis: Stochastic Hybrid Games and **Uppaal Stratego**

5. **Conclusion**
   - Tool Support
   - Discussion
<table>
<thead>
<tr>
<th></th>
<th>Scheduling and Mapping on Multi-core Hardware</th>
<th>Energy Optimization by Power Management</th>
<th>Extension 1: Battery Management</th>
<th>Extension 2: Handling Uncertainty by Adaptation</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Modeling: Synchronous Dataflow and Hardware Platform</td>
<td>Analysis: Timed Automata and UPPAAL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Energy Optimization by Power Management</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Extension 1: Battery Management</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Extension 2: Handling Uncertainty by Adaptation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Conclusion</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Synchronous Data Flow graphs

An SDF Graph is a tuple $G = (A, D, Tok_0, \tau)$ where:

- $A$ is a finite set of actors
An SDF Graph is a tuple $G = (A, D, \text{Tok}_0, \tau)$ where:

- $A$ is a finite set of actors
- $D$ is a finite set of channels $D \subseteq A^2 \times \mathbb{N}^2$
Synchronous Data Flow graphs (Lee, 1986)

An SDF Graph is a tuple $G = (A, D, Tok_0, \tau)$ where:

- $A$ is a finite set of actors
- $D$ is a finite set of channels $D \subseteq A^2 \times \mathbb{N}^2$
- $Tok_0 : D \rightarrow \mathbb{N}$ denotes initial tokens in each buffered channel

![SDF Graph Example](image_url)
Synchronous Data Flow graphs (Lee, 1986)

An SDF Graph is a tuple \( G = (A, D, \text{Tok}_0, \tau) \) where:

- \( A \) is a finite set of actors
- \( D \) is a finite set of channels \( D \subseteq A^2 \times \mathbb{N}^2 \)
- \( \text{Tok}_0 : D \rightarrow \mathbb{N} \) denotes initial tokens in each buffered channel
- \( \tau : A \rightarrow \mathbb{N}_{\geq 1} \) assigns an execution time to each actor

![Diagram of Synchronous Data Flow graph](image-url)
Synchronous Data Flow graphs (Lee, 1986)

An SDF Graph is a tuple $G = (A, D, \text{Tok}_0, \tau)$ where:

- $A$ is a finite set of actors
- $D$ is a finite set of channels $D \subseteq A^2 \times \mathbb{N}^2$
- $\text{Tok}_0 : D \rightarrow \mathbb{N}$ denotes initial tokens in each buffered channel
- $\tau : A \rightarrow \mathbb{N}_{\geq 1}$ assigns an execution time to each actor

- Actor firing: consume and produce tokens through channels
- Actors fire concurrently; even auto-concurrency is possible
Some SDF Examples

(Wiggers’09, Theelen’12)

MP3 Playback

MPEG-4 Decoder
Some SDF Examples (Wiggers’09, Theelen’12)

MP3 Playback

MPEG-4 Decoder

Face Recognition (Recore)
To get to a stable periodic execution:

- Repetition vector \( \gamma : A \rightarrow \mathbb{N}_{\geq 1} \), such that \( p.\gamma(a) = q.\gamma(b) \)
- For this example: \( \gamma\langle u, v, w \rangle = \langle 4, 2, 3 \rangle \)
To get to a stable periodic execution:

- Repetition vector $\gamma : A \rightarrow \mathbb{N}_{\geq 1}$, such that $p.\gamma(a) = q.\gamma(b)$
- For this example: $\gamma\langle u, v, w \rangle = \langle 4, 2, 3 \rangle$
- The self-timed execution guarantees maximum throughput:
SDF for parallel, self-timed execution

Advantages of SDF with self-timed execution

- SDF fits streaming applications (no data dependencies, SADF)
- Guarantees maximal throughput, transient + periodic phase
- Simple, polynomial algorithms (implemented in SDF3)
SDF for parallel, self-timed execution

Advantages of SDF with self-timed execution

- SDF fits streaming applications (no data dependencies, SADF)
- Guarantees maximal throughput, transient + periodic phase
- Simple, polynomial algorithms (implemented in SDF3)

Limitations of SDF with self-timed execution

- Leads to “maximal parallelism”, which is expensive
- So far, a homogeneous processor model is assumed
- Worst-case assumptions lead to over-dimensioning
Some actors can be mapped to particular processors only
- floating point, analog/digital, . . .

Definition

A hardware platform model is a tuple \((P, \zeta)\) consisting of
- a finite set \(P\) of processors; and
- a function \(\zeta: P \times A \rightarrow \{0, 1\}\): actor \(A\) can run on processor \(P\)
Table of Contents

1. Scheduling and Mapping on Multi-core Hardware
   - Modeling: Synchronous Dataflow and Hardware Platform
   - Analysis: Timed Automata and UPPAAL

2. Energy Optimization by Power Management

3. Extension 1: Battery Management

4. Extension 2: Handling Uncertainty by Adaptation

5. Conclusion
Approach using Timed Automata
Ahmad, De Groote, Hölzenspies, Stoelinga, van de Pol

Application SDF Graph

Translation to TA

Mapping & Scheduling by UPPAAL

Model-Checking

Optimal Schedule

Architecture

Translation to TA

Energy-optimization with Timed Automata
Ingredients of Timed Automata

- States and Transitions as in finite automata
- Real valued clock variables (here $y$)
- Clock constraints, invariants and resets ($y \geq 5$, $y := 0$)
- Synchronisation between automata via action labels
Ingredients of Timed Automata

- States and Transitions as in finite automata
- Real valued clock variables (here $y$)
- Clock constraints, invariants and resets ($y \geq 5$, $y := 0$)
- Synchronisation between automata via action labels
- The UPPAAL model-checker can then check:
  - reachability of states (also safety, liveness properties)
  - It can synthesize shortest or fastest traces
Translation of Processor Model and SDF graph

- A processor can be occupied by at most one task at the time
- Clock $x$ specifies the duration of the task

Model for each Processor $p_{id}$:

![Timed Automata Diagram]

- InUse_a
- InUse_b
- Idle

- $x:=0$
- $x:=0$
- $x:=3$
- $x:=3$
- fire[p_id][c]?
- fire[p_id][b]?
- end[p_id][c]!
- end[p_id][b]!
- end[p_id][a]!
Translation of Processor Model and SDF graph

- A processor can be occupied by at most one task at the time
- Clock $x$ specifies the duration of the task
- **On firing:** check for required tokens, and consume them
- **On ending:** produce the required tokens

Model for each Processor $p_{id}$:

Model for actors $a$, $b$, $c$:

---

UNIVERSITY OF TWENTE.
Synthesizing a schedule

- Actors are mapped on Processors non-deterministically
- We query UPPAAL for the fastest trace to a full iteration.
- In addition, the model checker UPPAAL could check for absence of deadlocks, buffer bounds, safety, liveness, ...
Synthesizing a schedule

- Actors are mapped on Processors non-deterministically
- We query UPPAAL for the fastest trace to a full iteration.
- In addition, the model checker UPPAAL could check for absence of deadlocks, buffer bounds, safety, liveness, ...
Self-timed execution uses 4 processors: throughput 1/9
Results on Running Example: throughput versus processors

Self-timed execution uses 4 processors: throughput 1/9

Restrict to 3 processors: still the same throughput 1/9
Results on Running Example: throughput versus processors

Self-timed execution uses 4 processors: throughput 1/9

Restrict to 3 processors: still the same throughput 1/9

Restrict to 2 processors: small penalty, throughput 1/11
Assume: $p_0, p_1$ can execute task $u, v$; and $p_2, p_3$ only execute $w$

Self-timed execution uses 4 processors: throughput 1/9

Using 4 heterogeneous processors, still same throughput 1/9
Table of Contents

1. Scheduling and Mapping on Multi-core Hardware
   - Modeling: Synchronous Dataflow and Hardware Platform
   - Analysis: Timed Automata and UPPAAL

2. Energy Optimization by Power Management
   - Modeling: DPM, DVFS, VFI in Hardware Platform Model
   - Analysis: Priced Timed Automata and UPPAAL CORA

3. Extension 1: Battery Management
   - Modeling: Kinetic Battery Model
   - Analysis: Hybrid Automata and UPPAAL SMC

4. Extension 2: Handling Uncertainty by Adaptation
   - Modeling: Execution Time is Stochastic
   - Analysis: Stochastic Hybrid Games and UPPAAL STRATEGO

5. Conclusion
   - Tool Support
   - Discussion
Dynamic Power Management (DPM)

- Idle processors can be switched off to a low power (sleep) state
- This reduces energy usage in idle time
  - Don’t ignore it static power consumption
  - Switching consumes some energy as well

![Diagram of power states and transitions]

- LCD: ON → OFF
- Backlight: ON → OFF
Voltage and Frequency Scaling (DFVS) (Zhuravlev'13)

- Lower the voltage and frequency dynamically
- This reduces energy during active time
  - At the expense of increasing the execution time
  - Again, switching frequency levels has some cost
Voltage and Frequency Islands (VFI) (Ogras, DAC’07)

- Local versus Global DVFS

- VFI: a group of processors clustered together
- Common clock frequency/voltage per island
- Examples: Intel i7, IBM Power 7 series, Samsung S5 octa core
We extend the Hardware Platform Model to the following tuple:

\[
(P, \zeta, F, Time_{act}, Pow_{idle}, Pow_{occ}, Pow_{tr})
\]

- \(P\): set of processors, partitioned in voltage frequency islands
- \(\zeta: P \times A \rightarrow \{0, 1\}\): maps actors on heterogeneous processors
- \(F\): finite set of frequency levels \(f_1 < \cdots < f_m\)
- \(Time_{act}: A \times F \rightarrow \mathbb{N}_{\geq 1}\): execution time of \(A\) at frequency \(F\)
- \(Pow_{idle}: P \times F \rightarrow \mathbb{R}^+\): power consumption in idle state
- \(Pow_{occ}: P \times F \rightarrow \mathbb{R}^+\): power consumption in busy state
- \(Pow_{tr}: P \times F^2 \rightarrow \mathbb{R}^+\): overhead of switching frequency level
Example: Samsung Exynos 4210  
(Park, TCAD’13)

<table>
<thead>
<tr>
<th>Level</th>
<th>Voltage (V)</th>
<th>Frequency (MHz)</th>
<th>Exper. $P_{\text{idle}}$(W)</th>
<th>Exper. $P_{\text{occ}}$(W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.2</td>
<td>1400</td>
<td>0.1</td>
<td>4.6</td>
</tr>
<tr>
<td>2</td>
<td>1.15</td>
<td>1312.2</td>
<td>0.1</td>
<td>3.9</td>
</tr>
<tr>
<td>3</td>
<td>1.10</td>
<td>1221.8</td>
<td>0.1</td>
<td>3.2</td>
</tr>
<tr>
<td>4</td>
<td>1.05</td>
<td>1128.7</td>
<td>0.1</td>
<td>2.5</td>
</tr>
<tr>
<td>5</td>
<td>1.00</td>
<td>1032.7</td>
<td>0.1</td>
<td>1.8</td>
</tr>
</tbody>
</table>
Table of Contents

1  Scheduling and Mapping on Multi-core Hardware

2  Energy Optimization by Power Management
   - Modeling: DPM, DVFS, VFI in Hardware Platform Model
   - Analysis: Priced Timed Automata and UPPAAL CORA

3  Extension 1: Battery Management

4  Extension 2: Handling Uncertainty by Adaptation

5  Conclusion
Scheduling & Mapping

Power Optimization

Batteries

Adaptation

Conclusion

Approach with Priced Timed Automata – **UPPAAL CORA**
Ahmad, Hölzenspies, Stoelinga, van de Pol

**Application SDF Graph**

**Architecture**

Translation to PTA

**UPPAAL CORA**

**Model-Checking**

**Energy Efficient Schedule**

**Throughput Requirement**

Energy-optimization with Timed Automata

18 Aug 2016 24 / 44
Specification of a lamp:

- Costs accumulate over transitions and while residing in states
- **Uppaal Cora** can synthesize fastest or cheapest traces

![Diagram of a lamp specification with states and transitions.](image-url)
Translation of Hardware Platform Model

- InUse_FD_f1
  - fire[p_id][FD]?
  - x:=0, freq_lev[vfi_id]+=1
  - x:=3 && cost'==18
  - freq_lev[vfi_id]-=1
  - end[p_id][FD]!

- InUse_FD_f2
  - fire[p_id][FD]?
  - x:=2 && cost'==46
  - freq_lev[vfi_id]-=1
  - end[p_id][FD]!

- Idle_f1
  - cost'==1

- fjump_12[vfi_id]?
  - cost+=1, curr_freq[vfi_id]=2

- Idle_f2
  - cost'==1

- fjump_21[vfi_id]?
  - cost+=4, curr_freq[vfi_id]=1
Using **Uppaal Cora**

**Uppaal Cora** supports computing **fastest** and **cheapest** traces

- Obtain the **completion time** via fastest trace as before
- Incorporate the found completion time in the model
- Obtain **Energy Efficient Schedule** via cheapest trace
Using **Uppaal Cora**

**Uppaal Cora** supports computing fastest and cheapest traces

- Obtain the completion time via fastest trace as before
- Incorporate the found completion time in the model
- Obtain Energy Efficient Schedule via cheapest trace

**Design Exploration**

Let’s analyse the energy usage when we vary

- The required throughput
- The number of processors and frequency islands
Results of Power Management Optimization for MPEG-4

Energy/-frame versus Frames/second.
Legend: number of processors/VFIs.
Energy/frame versus Frames/second.
Legend: number of processors/VFIs.
High throughput: using more processors is cheaper (run at decent speed)
Results of Power Management Optimization for MPEG-4

Energy/Frame versus Frames/second.
Legend: number of processors/VFIs.
Lower throughput: using less processors is cheaper (can switch off others)
Results of Power Management Optimization for MPEG-4

Energy consumption (mWs) versus Frames per second.

Legend: number of processors/VFIs.

Very low throughput: uses more energy per frame (idle time dominates)
# Table of Contents

1. **Scheduling and Mapping on Multi-core Hardware**  
   - Modeling: Synchronous Dataflow and Hardware Platform  
   - Analysis: Timed Automata and UPPAAL

2. **Energy Optimization by Power Management**  
   - Modeling: DPM, DVFS, VFI in Hardware Platform Model  
   - Analysis: Priced Timed Automata and UPPAAL CORA

3. **Extension 1: Battery Management**  
   - Modeling: Kinetic Battery Model  
   - Analysis: Hybrid Automata and UPPAAL SMC

4. **Extension 2: Handling Uncertainty by Adaptation**  
   - Modeling: Execution Time is Stochastic  
   - Analysis: Stochastic Hybrid Games and UPPAAL STRATEGO

5. **Conclusion**  
   - Tool Support  
   - Discussion
Kinetic Battery Model (KiBaM) [Manwell, McGowan, 1993]

Include battery scheduling considerations. Needs a model.

- This model explains the apparent “recovery effect” of batteries
- KiBaM distinguishes two wells: bound and available charge
- Two differential equations describe the evolution over time

\[
\begin{align*}
\dot{a}(t) &= -i(t) + k(h_b - h_a) \\
\dot{b}(t) &= -k(h_b - h_a) \\
h_a &= \frac{a(t)}{c} \\
h_b &= \frac{b(t)}{1-c}
\end{align*}
\]
Approach using Hybrid Automata – *Uppaal SMC*
Ahmad, Jongerden, Stoelinga, van de Pol

HA for Thermostat

Simulation of HA evolution

Hybrid Automata [Henzinger, 1996]

- Basis: states and transitions as in automata
- Clock constraints, invariants, synchronisation via action labels
- Real-time variables evolve with a rate governed by ODEs
The *UPPAAL SMC* model checker

- Supports (Stochastic) Hybrid automata
- Statistical Model Checking (cf. Monte Carlo simulations)
- Visualisation of simulated runs with line or bar plots

We (and others) have used this to investigate:

- Various *battery scheduling policies* [Wognsen, Haverkort, et al. '15]
- Variation in number of batteries, processors
- *Survivability* of the system under various circumstances

Scales better than previous approach using discretisation and PTA
Table of Contents

1. Scheduling and Mapping on Multi-core Hardware
   - Modeling: Synchronous Dataflow and Hardware Platform
   - Analysis: Timed Automata and UPPAAL

2. Energy Optimization by Power Management
   - Modeling: DPM, DVFS, VFI in Hardware Platform Model
   - Analysis: Priced Timed Automata and UPPAAL CORA

3. Extension 1: Battery Management
   - Modeling: Kinetic Battery Model
   - Analysis: Hybrid Automata and UPPAAL SMC

4. Extension 2: Handling Uncertainty by Adaptation
   - Modeling: Execution Time is Stochastic
   - Analysis: Stochastic Hybrid Games and UPPAAL STRATEGO

5. Conclusion
   - Tool Support
   - Discussion
Can we avoid over-dimensioning due to worst-case execution time?

Strategy can be adaptive: If task A happens fast, we can relax for task B

Model: replace WCET by lower/upper bound

Goal: Guarantee required throughput, minimize expected energy

10 simulations: Safe Strategy

10 simulations: Optimized Strategy
**UPPAAL STRATEGO** (David, Jensen, Larsen, Mikučionis, Taankvist, ’15)

**Stochastic Hybrid Automata**

- Extends Hybrid Automata
- Distinguish controllable transitions (scheduling decisions) from non-controllable transitions (stochastic execution times)
- Synthesize safe strategies, guaranteeing a required throughput
- Refine to near-optimal strategies, optimizing energy usage
- **Means:** game strategy synthesis, statistical model checking, reinforcement learning
# Table of Contents

1. **Scheduling and Mapping on Multi-core Hardware**
   - Modeling: Synchronous Dataflow and Hardware Platform
   - Analysis: Timed Automata and **Uppaal**

2. **Energy Optimization by Power Management**
   - Modeling: DPM, DVFS, VFI in Hardware Platform Model
   - Analysis: Priced Timed Automata and **Uppaal Cora**

3. **Extension 1: Battery Management**
   - Modeling: Kinetic Battery Model
   - Analysis: Hybrid Automata and **Uppaal SMC**

4. **Extension 2: Handling Uncertainty by Adaptation**
   - Modeling: Execution Time is Stochastic
   - Analysis: Stochastic Hybrid Games and **Uppaal Stratego**

5. **Conclusion**
   - **Tool Support**
   - **Discussion**
Meta Models, Transformations, Graphical Editors

Similar, we designed (or adapted):

- Meta-models for SDF, compatible with SDF3 tool suite
- Meta-models for PTA, compatible with UPPAAL tool suite
- Model2Model Transformations: SDF + HPM → PTA
Tool Support

- Tool Chain is compatible with SDF3 and UPPAAL -family
- Mappings are compositional model transformations in Epsilon

<table>
<thead>
<tr>
<th>Self-timed execution</th>
<th>SDF graphs</th>
<th>SDF3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapping &amp; Scheduling</td>
<td>Timed Automata</td>
<td>UPPAAL</td>
</tr>
<tr>
<td>Power Management</td>
<td>Priced Timed Automata</td>
<td>UPPAAL CORA</td>
</tr>
<tr>
<td>Battery Management</td>
<td>Hybrid Automata</td>
<td>UPPAAL SMC</td>
</tr>
<tr>
<td>Handling Uncertainty</td>
<td>Hybrid Stochastic Games</td>
<td>UPPAAL STRATEGO</td>
</tr>
</tbody>
</table>

Main challenge: scalability.

- Better Algorithms (heuristics, statistical methods)
- More clever modeling
- Leverage High-performance Model Checking
Visualizing the result: http://www.mudcrab.eu/SDF-Fish/
**Table of Contents**

1. Scheduling and Mapping on Multi-core Hardware
2. Energy Optimization by Power Management
3. Extension 1: Battery Management
4. Extension 2: Handling Uncertainty by Adaptation
5. Conclusion
   - Tool Support
   - Discussion
### Summary

**Scheduling and Mapping on Heterogeneous Multi-core**

- Schedule maximal throughput on *small number of processors*
- Handling *heterogeneous* processor application models
Summary

Scheduling and Mapping on Heterogeneous Multi-core

- Schedule maximal throughput on small number of processors
- Handling heterogeneous processor application models

Power Optimization

- Exploit and combine modern power management strategies
- Use model checking PTA for optimal mapping & scheduling, trading off throughput versus energy
Summary

Scheduling and Mapping on Heterogeneous Multi-core

- Schedule maximal throughput on small number of processors
- Handling heterogeneous processor application models

Power Optimization

- Exploit and combine modern power management strategies
- Use model checking PTA for optimal mapping & scheduling, trading off throughput versus energy

Extensible method

- Battery Management
- Adaptation to Uncertainty

Hybrid Automata
Stochastic Hybrid Games
Applications and Future Work

- FP7 SENSATION had several Cool and Hot applications:
  - Face Recognition (Recore, Twente)
  - Nano-satellites (Gomspace, Aalborg)
  - E-bikes (EnergyBus consortium)
Applications and Future Work

- FP7 SENSATION had several Cool and Hot applications:
  - Face Recognition (Recore, Twente)
  - Nano-satellites (Gomspace, Aalborg)
  - E-bikes (EnergyBus consortium)

- Get closer to engineering
  - More extensive experimental validation is still required
  - Generate SDF + HPM from instrumented experiments

- Extensions
  - Include: communication costs, processor pinning/affinity, ...
  - Include: battery charging: fully energy-self-aware systems

- Scalability (heuristic algorithms + parallel implementation)
This presentation is based on four papers with PhD student Waheed Ahmad and co-supervisor Mariëlle Stoelinga:

- Ahmad, De Groote, Hölzengies, Stoelinga, van de Pol, *Resource-constrained optimal scheduling of synchronous dataflow graphs via timed automata*. ACSD’14

- Ahmad, Hölzengies, Stoelinga, van de Pol, *Green computing: Power optimisation of VFI-based real-time multiprocessor dataflow applications*. DSD’15

- Ahmad, Jongerden, Stoelinga, van de Pol, *Model checking and evaluating QoS of batteries in MPSoC dataflow applications via hybrid automata*. ACSD’16

- Waheed Ahmad and Jaco van de Pol, *Synthesizing Energy-Optimal Controllers for Multiprocessor Dataflow Applications with Uppaal Stratego*. ISOLA’16