Formal Model-Driven Validation of Deployed Architectures with a Focus on Power Consumption

— Power Trace Testing (PTT) —

Kai Lampka kai.lampka@it.uu.se
Ass. Prof. for Embedded Systems
Dep. of Information Technology
Uppsala University, Sweden

Joint work with Matthias Woehrle and Lothar Thiele (ETH Zurich)


3. Matthias Woehrle, Kai Lampka and Lothar Thiele: Exploiting...
Scenario

Wireless Sensor Network (WSN)
- battery operated and
- located in a remote area (PermaSense project)

Power efficiency:
Guarantee of long lasting operation is important:
- data collection of an individual node, and
- functionality of the network
TinyNode

- Battery operated systems
- Real-time requirements, data sampling, communication

- Energy-efficiency
  - Use low power modes of HW
  - Return to a low power mode as soon as possible to save energy

- Efficiency and correctness left to the implementation engineer
Examples of (real-time) constraints

• Monitor data sampling, e.g. send buffered data every 2 minutes to the sink 4 packets with 40 Bytes.

• Wake-up period requires additional time for initializing the system’s operational mode.

• Wake up every sec. to check for radio activity (channel polling for incoming traffic)
Energy-aware programming of real-time systems

• HW-oriented programming: explicitly exploit power modes; e. g., switch radio off

• Leakage of power: not all HW modes are employable at the same time

• Strong RT-requirements to be fulfilled by I/O-triggered activities of the nodes;
  – data sampling,
  – forwarding of incoming traffic, etc.

• No back-up like exception handling possible; e. g. reboot may help but may also cause troubles

• Environmental conditions severely affect system behavior (clock oscillator, power consumption / temperature, reloading of batteries)
Our aim: Formal conformance test

Test environment

Inputs

Implementation

Conformance test

YES or NO

Specification model

Inputs

Measured power consumption (Power Trace)

Model for describing system behavior, i.e., time-wise, function-wise, and w.r.t. the power consumption
Model Checker (requires formal model)

Model of expected behavior (Sys)

Model of observed behavior (PT)

\[ \exists \pi \in \prod_{Sys} | PT : s \xrightarrow{\pi} t \iff PT \models Sys \]

Power trace

Manually provided

System in operation

Deployed Architecture

Manually provided

Expected behavior

System Specification
A timed automaton extended with variables is a tuple $TA = (Loc, Loc_0, Act, C, V, \rightarrow, I, AP, L)$ where:

- $Loc$ is a finite set of locations
- $Loc_0 \subseteq Loc$ is a set of initial locations
- $Act$ is a set of actions
- $C$ is a finite set of clocks
- $V$ is a finite set of (discrete) variables
- $\rightarrow \subseteq Loc \times Cons(C) \cup Cons(V) \times Act \times 2^C \times 2^V \times Loc$ is a transition relation
- $I: Loc \rightarrow Cons(C) \cup Cons(V)$ is an invariant-assignment function
- $AP$ is a finite set of atomic propositions, and
- $L: Loc \rightarrow 2^{AP}$ is a labeling function for the locations

![Timed automaton diagram](image)
Operational Semantics

- **delay transition**

\[ \langle l, u, v \rangle \xrightarrow{d} \langle l, u + d, v \rangle \quad \text{if } \forall d' \in \mathbb{R}_{\geq 0} : 0 \leq d' \leq d \Rightarrow u + d' \in I(l) \]  

(1)

- **discrete transition**

\[ \langle l, u, v \rangle \xrightarrow{\alpha} \langle l', u', v' \rangle \quad \text{if } l \xrightarrow{g, \alpha_{r, a}} \ x' \quad \text{and } u \in g \text{ and } u' = [r \rightarrow 0]u \text{ and} \]

\[ u' \in I(l') \text{ and } v' \in I(l') \text{ and } v' = \begin{cases} c \in \mathbb{R}_{\geq 0} \leftrightarrow v' \in a, a \subset V \\ v \text{ else} \end{cases} \]  

(2)
Auto-generated trace model (TM)

1.79204827
2.32205325
0.288717749
0.0842842821
1.57702206
2.0505207
2.25958169
2.8585387
2.94728409
2.70120291
2.55394464
2.59723726

Fine-grained measurements yields large number of equidistant entries
Compression: Greatest Common Intervals (GCI)
**Benefit of “smart” discretization**

<table>
<thead>
<tr>
<th>Model</th>
<th>Samples</th>
<th>Intervals</th>
<th>100μA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wake-up</td>
<td>1000000</td>
<td>2089</td>
<td>13812</td>
</tr>
<tr>
<td>Inject</td>
<td>990000</td>
<td>2040</td>
<td>13666</td>
</tr>
<tr>
<td>Complex Trace</td>
<td>310000</td>
<td>21811</td>
<td>89678</td>
</tr>
<tr>
<td>MC state</td>
<td>1000000</td>
<td>1086</td>
<td>5539</td>
</tr>
<tr>
<td>Specification</td>
<td>1000000</td>
<td>1478</td>
<td>4578</td>
</tr>
</tbody>
</table>

Where do these intervals come from?
Obtaining a TA-based model form outputs

The compressed power trace is converted into a sequential TA, the labelled locations (stateTrace) model the duration of power consumptions.

Transitions:
1. update the power consumption (p)
2. reset the clock for triggering change of power measurement (y)
TA-based model of system: the radio
TA-based modeling of SW module

For complexity reasons we also restrict the behavior of the SW to some use-cases, which somehow take the interaction with the environment into account.
Annotating the system model with power values

For keeping things simple and understandable we emphasize a compositional modeling style, where components communicate via binary synchronization and shared variables, e.g., as implemented in the timed model checker Uppaal.

- Locations are annotated with upper and lower power values consumed by the system while residing in the resp. “mode”.
  These bounds can be obtained from manufacturer's data sheets or characterization measurements.

- For computing the power consumption of the overall model, we sum over lower, and upper power values of those locations the system is currently residing in. These are the intervals for compressing the power trace.
Example

Idle

RadioUp := 4000
RadioLow := 200

WarmUp

Idle

μCUp := 1800
μCLow := 100

Awake
Example

Idle

RadioUp := 4000
RadioLow := 200

WarmUp

SysUp := 4000
SysLow := 200

Idle

µCUp := 1800
µCLow := 100

Awake
Conformance Test

• Joint execution of system model and model of power measurement yields our conformance test.
• Reachability of the PT's final location gives the acceptance condition:
\[ \exists \pi \in \prod_{\text{Sys}} \parallel PT : s \xrightarrow{\pi} t \iff PT \models Sys \]
• In case of a negative answer one may execute a binary search for detecting the last state, where power measurement is continued in the system behavior (deadlock in the execution of the formal model). This might give evidence where expected and observed behavior diverge; not modeled or implementation error
• May produce false positives;
Example

\[ p := 120, y := 0 \]
\[ p := 2000, y := 0 \]

\[ y \leq 0 \quad y = 0 \quad y \leq 100 \quad y = 100 \quad y \leq 250 \quad \text{final} \]

Trajectory at time \( 0 \leq y < 100 \)

\[ \text{MC\textunderscore up} := 1800 \]
\[ \text{MC\textunderscore low} := 100 \]

idle

\[ \text{warm-up} \quad \text{MC\textunderscore low} \leq p \leq \text{MC\textunderscore up} \]
Example

<table>
<thead>
<tr>
<th>y ≤ 0</th>
<th>p := 120, y := 0</th>
<th>y ≤ 100</th>
<th>p := 2000, y := 0</th>
<th>y ≤ 250</th>
<th>final</th>
</tr>
</thead>
</table>

Trajectory at time y = 100

<table>
<thead>
<tr>
<th>idle</th>
<th>MC_up := 1800</th>
<th>deadlock!</th>
<th>MC_low := 100</th>
<th>warm-up</th>
<th>MC_low ≤ p ≤ MC_up</th>
</tr>
</thead>
</table>

MC_low := 100
MC_up := 1800
Method
Empirical evaluation

<table>
<thead>
<tr>
<th>Model</th>
<th>Samples</th>
<th>Compression</th>
<th>Runtime</th>
<th>TRON</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Intervals</td>
<td>100uA steps</td>
<td>Intervals</td>
</tr>
<tr>
<td>Wake-up</td>
<td>1000000</td>
<td>2089</td>
<td>13812</td>
<td>8s</td>
</tr>
<tr>
<td>Inject</td>
<td>990000</td>
<td>2040</td>
<td>13666</td>
<td>5s</td>
</tr>
<tr>
<td>Complex Trace</td>
<td>310000</td>
<td>21811</td>
<td>89678</td>
<td>15 min</td>
</tr>
<tr>
<td>MC state</td>
<td>1000000</td>
<td>1086</td>
<td>5539</td>
<td>4s</td>
</tr>
<tr>
<td>Specification</td>
<td>1000000</td>
<td>1478</td>
<td>4578</td>
<td>7s</td>
</tr>
</tbody>
</table>
Contributions

• **Power Trace Testing**: Conformance testing of power measurements using a timed automata representation.

• The feasibility of the approach is validated by a realistic case study based on commercial embedded devices and a standard wireless sensor network application (Harvester).

• The method has been improved towards scalability in order to be able to deal with realistic scenarios (FORMATS 2011).
BACKUP
Conformance relation - rtioco

\[ s \text{ rtioco}_e t \quad \text{iff} \quad \sigma \in TTr(e). \text{Out}((s, e) \text{ After } \sigma) \supset \text{Out}((t, e) \text{ After } \sigma) \]

**Uppaal (offline)**
- Automata based formulation
- Composition of trace and specification
- Reachability of final trace location

**TRON (online)**
- Power trace as timed trace
- Specification automaton
- Iterative trace inclusion
HW (Component) power consumption

![Graph showing current consumption over time with specific points and intervals.]

- Current (mA): 0 to 20
- Time (s): 11.608 to 11.618
- 1ms ⇒ 50 samples

Points:
1. 11.608
2. 11.609
3. 11.61
4. 11.611

Consumption peaks and intervals are indicated with arrows and labels.
Related Work

• Quanto [Fonseca2008]
  – fine-grained energy consumption measurements
  – attribution to software modules

• Model-based Testing
  – ioco [Tretmans1999]
  – rtioco [Larsen2004]
Input representation

\[ h_{HW,j} = [h_{HW,j}^{low}, h_{HW,j}^{up}] \]
Quick (Informal) Uppaal Cheat Sheet

- Timed Automata, i.e. timing annotations, e.g. for invariants
- Data variables (ints)
- Composition with synchronization
  - send! -> send?
- Queries (reachability)
  - Is there an execution, which might satisfy...?
  - $E<>\text{traceTerminal}$
SW Model

- Individual software aspects
- Integrated with the environment
- Testcase elaborates on specific behavior