From High-Level Models to Time-Triggered Implementations
Hela GUESMI(1,3), Belgacem BEN HEDIA(1), Simon BLIUDZE(2), Saddek BENSELEM(3)
(1)CEA, LIST, Embedded Real Time Systems Laboratory, 91199 Gif-sur-Yvette, France
(2)EPFL IC INF/COM Risd, Station 14, 1015 Lausanne, Switzerland
(3)Verimag, Université Joseph Fourier, 38031 Grenoble, France

A. Introduction

Developing embedded real-time systems based on the Time-Triggered (TT) paradigm is a challenging task due to the increasing complexity of such systems and the necessity to manage, already in the programming model, the fine-grained temporal constraints and the low-level communication primitives imposed by the temporal firewall abstraction.

1. State of the art:

- Component-based frameworks allowing designers to:
  - think on a higher level of abstraction,
  - tackle the complexity of critical real-time systems.
- RTOS implementing TT execution model

2. Challenge:

Establish a link between High-level design framework and a RTOS based on the TT approach.

B. Related Work

<table>
<thead>
<tr>
<th>Approach</th>
<th>Transformation</th>
<th>Correctness proofs</th>
<th>Components-based modeling</th>
<th>Target platform based on TT paradigm</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>From SCADIC model to PharOS</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[2]</td>
<td>From BIP to Distributed implementation</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>[3]</td>
<td>From AADL model to TT architecture</td>
<td>-</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>[4]</td>
<td>From BIP to real HW/SW systems</td>
<td>x</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>[5]</td>
<td>From QML model to TT architecture</td>
<td>-</td>
<td>-</td>
<td>x</td>
</tr>
</tbody>
</table>

Our approach: From RT-BIP to PharOS

C. Background

1. RT-BIP Framework:

- BIP is a component framework for constructing systems by the superposition of three layers: Behavior, Interaction, and Priority.
- Structure of a real-time BIP model:
  - Priorities
  - Interactions
  - Behavior
  - Connector

2. TCA computation model and PharOS platform:

- Time-Constrained Automata (TCA) is a formal computation model of TT tasks in PharOS. The temporal behavior of a task is specified using a directed graph, where arcs represent the successive jobs of the task to be executed (one at a time), and the nodes bear the temporal constraints of the jobs.

D. From RT-BIP to TT implementation

1. Global approach

- High-Level Model
- BIP Model
- Step 1 Transformation
- Physical Model
- TT-BIP Model
- Step 2 Transformation
- Computation Model
- TCA Model

2. Step 1[8]:

From RT-BIP to TT-BIP. The first step transforms the initial model, in order to comply with the TT paradigm. Components of initial model are grouped into tasks components following a user-defined task mapping. Each inter-task interaction is transformed in order to be handled by a dedicated component [8].

3. Step 2[9]:

From TT-BIP to TCA. This step transforms each TT-BIP automaton into a TCA automaton. Thus it aims at transforming timing constraints of the initial model into nodes of TCA.

- Communication mapping Challenge:
  - Interactions: data transfer and synchronization between sending and receiving actions of the sender and the receiver components.
  - Communication model: temporal variables.
  - A new value of a temporal variable is provided at each synchronization point of the sender.
  - Receivers can consult new values when their current time is equal to or higher than the visibility dates of consulted temporal variables.

- Proposed solution:
  - Finer grained clock: g= 4*c
  - Initial synchronization instant: advancing(advance(H) with g)

4. Proof of correctness of each step:

- Identifying OS service patterns potentially existing in the initial model. If detected, behavior of some components can be mapped to one of the OS services.
  - In that case, the transformation process should transform that part into a system call. And only the remaining components are transformed into TCA automata.
  - What about a generic transformation process? In order generalize the transformation to other RTOS-based implementations with TT execution model, we need to express its rules on the basis of a common formalism.

E. Open Challenges

- Identification of OS service patterns potentially existing in the initial model. If detected, behavior of some components can be mapped to one of the OS services.