Timed Automata Templates for Distributed Embedded System Architectures

Simon Perathoner[*,◇] Ernesto Wandeler[*] Lothar Thiele[*]
{psimon,wandeler,thiele}@tik.ee.ethz.ch

TIK-Report No. 233
[*]Computer Engineering and Networks Laboratory,
Swiss Federal Institute of Technology (ETH) Zurich, Switzerland

[◇]Dipartimento di Elettronica e Informazione,
Politecnico di Milano, Italy

Abstract

Networks of timed automata can be used to specify and verify distributed embedded systems. This technical report provides timed automata templates for a few basic components of such systems as a first step towards a library of reusable components which shall facilitate the design and analysis of large distributed embedded systems. In particular timed automata templates for common input event streams and for TDMA-scheduling are provided.

1 Introduction

Formal methods are an important tool for the design of distributed embedded systems. Many different formal approaches can be used to specify a system architecture and verify its correctness. [1] gives an overview of available formalisms for the design of real-time computing systems.

Timed automata [2] are one popular formalism for the specification of real-time systems. The model checker UPPAAL [3], which is used as reference tool for the templates of this report, allows to validate and verify real-time systems modeled as networks of timed automata extended with data types.

The formalism of timed automata itself is simple and comprehensive, but often even for apparently simple systems it turns out to be difficult and time-consuming to find a set of automata which models correctly the desired behavior

of the system and its environment. Moreover, many distributed embedded system architectures consist of the same basic components, as for example computation or communication resources implementing a certain scheduling strategy.

These two reasons encourage the use of prefabricated timed automata templates for the modeling of distributed embedded systems. These templates model 'basic blocks' of embedded system architectures which may be reused in various systems.

This technical report provides some basic timed automata templates for the modeling of distributed embedded systems. In particular timed automata for some input event streams used commonly as environment models and for a resource that implements TDMA scheduling are presented.

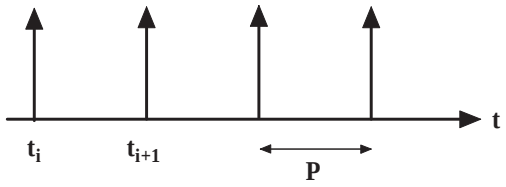
2 Input event stream models

In order to design and verify an embedded system, not only the computing architecture itself, but also the behavior of the system's environment must be modeled. In particular the time occurrence of events triggering the system (denoted as stimuli or input events) must be represented.

Periodic event streams, periodic event streams with jitter and periodic event streams with burst are very common arrival patterns for input events. This section provides timed automata templates which model these three event arrival patterns.

2.1 Periodic event stream

In a periodic event stream with period P the events occur exactly at intervals of P time units, as can be seen in Figure 1.



$$t_{i+1} - t_i = P$$

Figure 1: Periodic event stream

Figure 2 shows a timed automaton that models a periodic event stream.

The increment of the variable *event* models the issue of an event which can be handled by another automaton of the network.

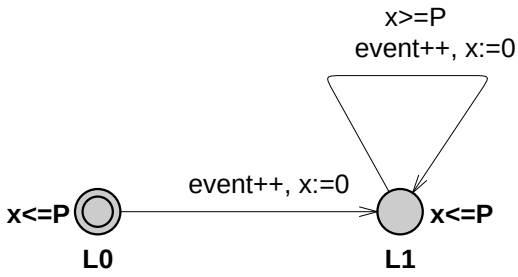


Figure 2: TA model of a periodic event stream

2.2 Periodic event stream with jitter

In a periodic event stream with jitter the events can arrive with a deviation from the ideal periodic arrival time. The deviation is bounded by an interval of length $J \leq P$, i.e. the intervals of admissible arrival times of successive events do not overlap. This can be seen in Figure 3 where the intervals of admissible arrival times are represented as shaded rectangles.

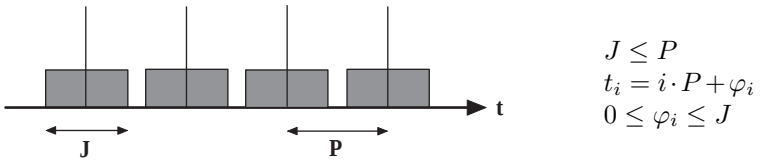


Figure 3: Periodic event stream with jitter

Figure 4 shows a simple timed automaton that models a periodic event stream with jitter.

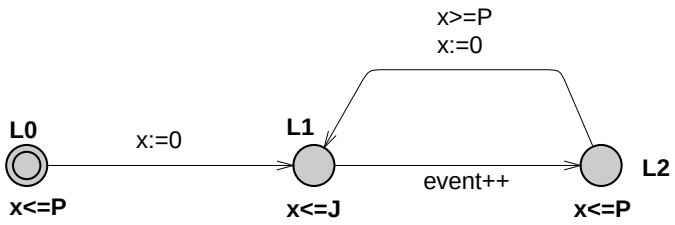
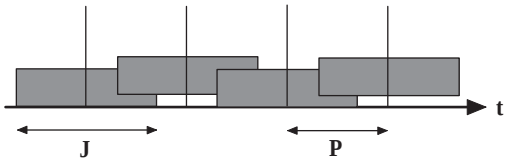


Figure 4: TA model of a periodic event stream with jitter

2.3 Periodic event stream with burst

The events of a stream can arrive in bursts if the deviation from the ideal periodic arrival time is larger than the period, i.e. if the intervals of admissible arrival times of successive events overlap, as shown in Figure 5. The number of immediately succeeding events forming a burst is bounded by the number of overlapping intervals.

Events cannot 'overtake' each other. This means that an event can arrive only after the precedent event has arrived. Moreover, a minimum event interarrival time d may be specified, with $d \leq P$. This constraint ensures that the deviation from the ideal periodic arrival time is still bounded by an interval of length J.



$J > P$
$t_i = i \cdot P + \varphi_i$
$0 \leq \varphi_i \leq J$
$t_{i+1} - t_i \geq d$

Figure 5: Periodic event stream with burst

The timed automaton depicted in Figure 6 is a simple straightforward implementation of the above formulas.

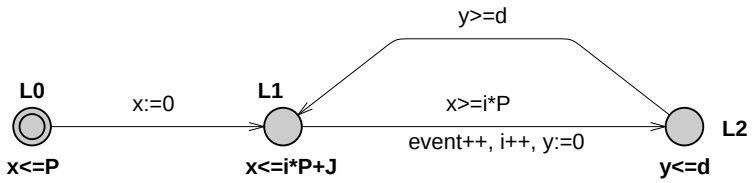


Figure 6: Naive TA model of a periodic event stream with burst

Although this timed automaton is conceptually correct, it is not very useful in practice due to the use of the unbounded variable i: the attempt to verify a system using this timed automaton will lead to an overflow error of the model checker.

An alternative timed automaton for an input event stream with burst is shown in Figure 7. It uses two variables L and W to store the deadline of an event and the earliest possible arrival time of the next event. After generating an event and waiting for the minimum interarrival time d, the automaton checks if the earliest possible arrival time of the next event has already passed. In this case

the next event has been 'disturbed' and this must be memorized by updating L and W consequently. Otherwise L and W are reset to the default values J and P, i.e. the automaton returns to the initial situation.

Unfortunately also this model is problematic for model checking, as the variables L and W can still grow to infinity.

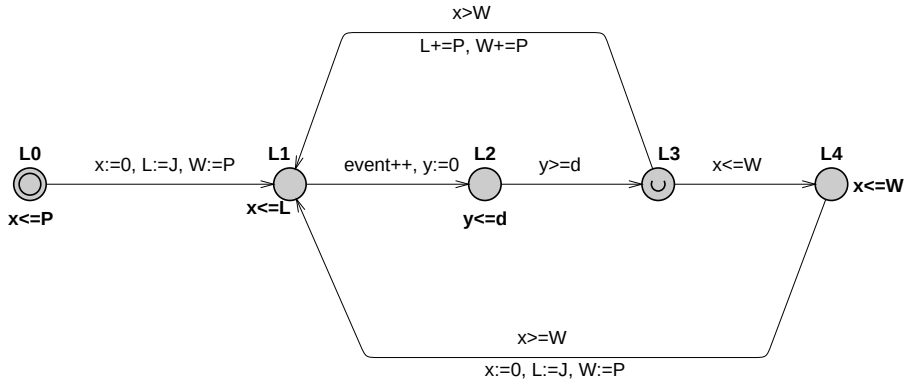


Figure 7: TA model of a periodic event stream with burst

3 A simple TDMA scheduling model

The specification process of a distributed embedded system through a network of timed automata can be accelerated by providing prebuilt timed automata templates also for scheduling algorithms. This section presents a TA model for a resource implementing TDMA scheduling of two concurrent tasks that can be generalized to n tasks.

Figure 8 shows the timed automaton for TDMA scheduling. If the resource is idle, the automaton swaps permanently between the two locations *idle1* and *idle2* which model the fact that the resource is available for task1 or task2. The available slot lengths for the two tasks are *SLOT_T1* and *SLOT_T2*, respectively. If the automaton is in location *idle1* and a processing request for task1 arrives (represented by the fact that the variable *req_T1* is incremented) the automaton passes immediately[1] to the location *occ1* which models the processing of the task. The response time *D1* for the task is initially set to the task's execution time *WCET_T1*. If the next slot change occurs before the execution of the task is completed, the response time *D1* is incremented by the duration of the preemption. When the resource returns available for task1, the flag *T1_running*

[1]The immediate transition is enforced through the use of the *urgent edge* modeling pattern described in [3]. The pattern defines an urgent channel *go* which is synchronized with a permanently active transition in an additional automaton.

reminds the automaton that there is an old execution of task1 to complete before potential new requests for task1 can be served.

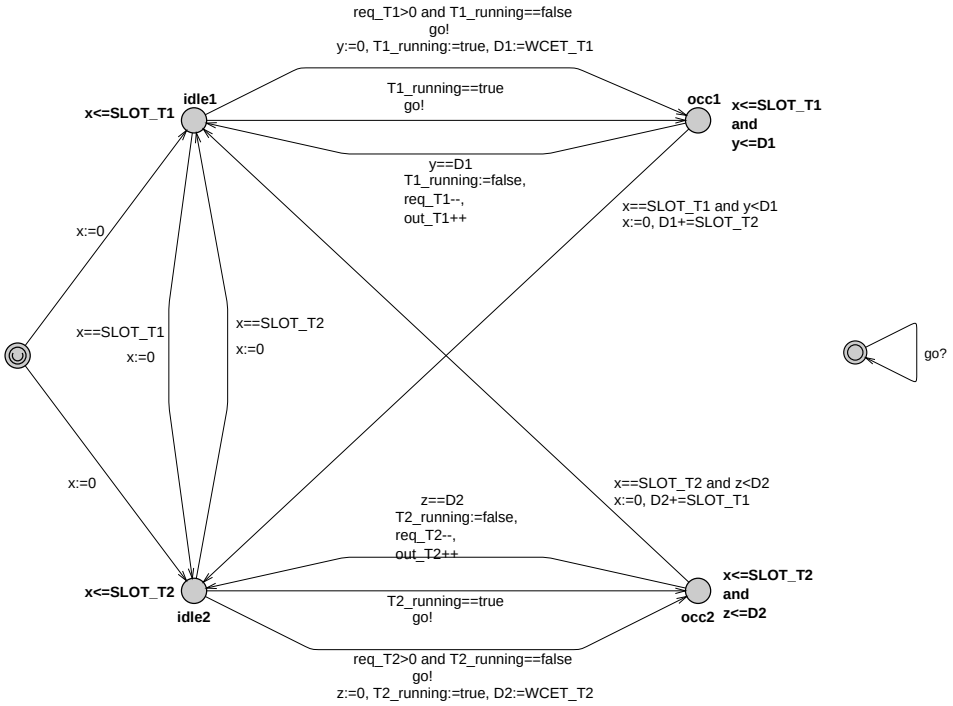


Figure 8: TA model of TDMA scheduling for 2 concurrent tasks

The automaton can be generalized to a model for a TDMA resource with n slots. It is sufficient to add a new 'horizontal layer' consisting of two locations *idle* and *occ* for every new TDMA-slot and to connect the locations with proper transition as suggested by the model for two slots.

4 Conclusions

In this report some timed automata templates have been provided which can be used as prebuilt components for the modeling of distributed embedded systems. The plan for future work is to model other input event streams and scheduling strategies with the goal to achieve a rich library of reusable components.

Furthermore, it is suggested to investigate the scalability of the compositional modeling approach. While systems consisting of only a few components can easily be verified with UPPAAL, the composition of a larger number of modules could lead to a state space explosion and impede rapid model checking for the entire system.

References

[1] Constance Heitmeyer and Dino Mandrioli. *Formal Methods for Real-Time Computing*. John Wiley & Sons, Inc., New York, NY, USA, 1996.

[2] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[3] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A tutorial on UPPAAL. In Marco Bernardo and Flavio Corradini, editors, *Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004*, number 3185 in LNCS, pages 200–236. Springer–Verlag, September 2004.