# Modeling and Analyzing Asynchronous Real-Time Systems[*]

Victor Pollex, Steffen Kollmann, Frank Slomka
Department of Embedded Systems / Real-Time Systems
Ulm University
{firstname}.{lastname}@uni-ulm.de

## Abstract

*Distributed embedded hard real-time systems consist of various components where each of them may operate on a different time base. For the response-time analysis it is important to model the stimulation of a task correctly. So far the response-time analysis does not consider the different time bases explicitly. In this paper we introduce an abstract clock model to consider the different time bases. Furthermore we will show how to adapt the response-time analysis to include the introduced clock model.*

## 1 Introduction

The size of networked electronic control units (ECU) used in the automotive industry grew rapidly in recent years. Not only the number of ECUs increased but also the number of buses used e.g. CAN, LIN, FlexRay. In particular the latter one specifies its own time-base independent from the other time-bases that might be used in the ECUs. Such a network constitutes an asynchronous distributed real-time system.

Assuming constant frequencies of the clocks the time-triggered protocol [3] proposes the usage of the fault tolerant average algorithm to establish a global time-base for a system. Although the precision of the global time-base is known, the clocks in the system will always deviate [4]. Even if it is possible to choose resynchronization periods short enough that the deviations of the clocks are negligible, there are cases where establishing a global time-base is not possible. This is the case when clocks with variable frequencies are used. For instance a combustion engine where ignition times have to be computed for each revolution. The task that computes the ignition times is triggered several times per revolution depending on the amount of cylinders the task is responsible for. In case that one engine control unit is not enough, all engine control units have to able to communicate with each other. Is this done through FlexRay, an additional time-base is added to the system and an acceptable global-time base can no longer be established, because the engine as a mechanical system cannot be synchronized.

Münzenberger et al. [6] introduced a time model which was designed to model the behaviour of actual implemented clocks. This model allows to determine the time of a clock at a given point in time of the reference time-base. In contrast to this model, we need to be able to model the number of clock cycles that have elapsed for a clock in a certain time interval. Therefore we propose a new abstract clock model.

During the design stage of a distributed real-time system its real-time requirements have to be verified. With a simulation the system's real-time behaviour can be observed. Due to the rarity of corner cases a simulation can only verify the real-time requirements with a certain probability. A formal analysis however is performed on guaranteed bounds. This leads to guaranteed assertions about the real-time behaviour of the system.

A widely known formal analysis is the response-time analysis as described by Joseph and Pandya [2]. Lehoczky [5] extended this analysis for the case of arbitrary deadlines. Tindell and Clark introduced the holistic schedulability analysis [8] to analyze distributed systems with the response time analysis. We will adapt this response-time analysis to include our proposed clock model to verify asynchronous systems.

We start by introducing the different models which are used. Section 3 shows how to adapt the response-time analysis in order to include the proposed clock model. We finish with a conclusion in section 4.

## 2 Models

### 2.1 Clock Model

Assume that every component operates on a clock or something that can be considered as a clock e.g. an engine. This clock represents the time base the component is working on. A cycle of the clock is the granularity of the time base. The clock model establishes the link between the real time and the time base of the clock.

**Definition 1** (Clock Function)**.** Let $\omega(t, \Delta t)$ denote the number of clock cycles that have ended in the interval $[t, t + \Delta t)$ of the reference time-base.

$$\omega : \mathbb{R} \times \mathbb{R}_0^+ \to \mathbb{N}_0, (t, \Delta t) \mapsto \omega(t, \Delta t)$$

To be able to use the clock model in the response-time analysis, we define the following bound:

**Definition 2** (Clock Function Bound)**.** Let $\omega^+(\Delta t)$ denote a subadditive upper bound for the number of clock

[*]Supported by the Carl Zeiss Foundation

cycles that have ended in any interval of length $\Delta t$.

$$\omega(t, \Delta t) \leq \omega^+(\Delta t) \tag{1}$$

holds for all $t \in \mathbb{R}, \Delta t \in \mathbb{R}_0^+$ and

$$\omega^+(\Delta t_1 + \Delta t_2) \leq \omega^+(\Delta t_1) + \omega^+(\Delta t_2) \tag{2}$$

holds for all $\Delta t_1, \Delta t_2 \in \mathbb{R}_0^+$. Furthermore let $\omega^+(\Delta t)$ be monotonically non-decreasing.

$$\omega^+(\Delta t_1) \leq \omega^+(\Delta t_2) \tag{3}$$

holds for all $\Delta t_1, \Delta t_2 \in \mathbb{R}_0^+$ where $\Delta t_1 < \Delta t_2$.

The clock function bound can be derived from the frequency of the clock in the component and its maximum deviation. Assume a clock with a frequency of 1 kHz and a maximum deviation of 5 ppm then for any interval of length $\Delta t$ at most

$$\omega^+(\Delta t) = \left\lceil (1 + 5 \cdot 10^{-6}) \cdot 1000 \cdot \Delta t \right\rceil$$

cycles would have ended, where the length of the interval $\Delta t$ is given in seconds.

### 2.2 Event Model

The amount of cycles that have ended in an interval of given length is needed, because it is assumed that events only occur at cycle boundaries. Therefore the event model describes the relationship between an amount of consecutive cycles and the amount of events that occurred during those cycles. The event model is an abstract model which is defined similar to the arrival curves of the real-time calculus [1].

**Definition 3** (Event Function). Let $\eta(c, \Delta c)$ denote the number of events that occurred in the cycle interval $[c, c + \Delta c)$.

$$\eta : \mathbb{Z} \times \mathbb{N}_0 \to \mathbb{N}_0, (c, \Delta c) \mapsto \eta(c, \Delta c)$$

For the response-time analysis a bound for the event function is also needed, defined analogously to the clock function bound.

**Definition 4** (Event Function Bound). Let $\eta^+(\Delta c)$ denote a subadditive upper bound for the number of events that occurred in any cycle interval of length $\Delta c$.

$$\eta(c, \Delta c) \leq \eta^+(\Delta c) \tag{4}$$

holds for all $c \in \mathbb{Z}, \Delta c \in \mathbb{N}_0$ and

$$\eta^+(\Delta c_1 + \Delta c_2) \leq \eta^+(\Delta c_1) + \eta^+(\Delta c_2) \tag{5}$$

holds for all $\Delta c_1, \Delta c_2 \in \mathbb{N}_0$. Furthermore let $\eta^+(\Delta c)$ be monotonically non-decreasing.

$$\eta^+(\Delta c_1) \leq \eta^+(\Delta c_2) \tag{6}$$

holds for all $\Delta c_1, \Delta c_2 \in \mathbb{N}_0$ where $\Delta c_1 < \Delta c_2$.

The event function bound can be derived from the actual application of the clock. Assume a timer which produces an event e.g. an interrupt every 100 cycles then for $\Delta c$ consecutive cycles at most

$$\eta^+(\Delta c) = \left\lceil \frac{\Delta c}{100} \right\rceil$$

events could have been produced.

### 2.3 Task Model

Each task is mapped to exactly one resource and each resource uses a fixed priority preemptive scheduling scheme. Furthermore each task is event triggered where the events are generated by a source e.g. a sensor or another task. Whenever a task is triggered by an event, a job of the task is generated which is then executed by the resource the task is mapped to.

**Definition 5** (Task). Let $\tau$ denote a task consisting of a clock function upper bound $\omega^+$, an event function upper bound $\eta^+$, a worst-case execution time $c^+$ and a priority $\phi$.

$$\tau := (\omega^+, \eta^+, c^+, \phi)$$

$\omega^+$ specifies the upper bound of cycles of the clock which is used in the source that triggers the task. The amount of events that can occur is bounded by $\eta^+$. The worst-case execution time $c^+$ specifies the maximum amount of time that a job of the task needs for it to be completely executed by the resource it is mapped to without any interference of the other tasks. $\phi$ is the priority of the task.

**Definition 6** (Task set). Let $\Gamma$ denote the set of tasks, which are mapped to the same resource. Let the tasks be sorted in decreasing order by their priority, meaning that $\tau_1$ is the task with the highest priority, $\tau_2$ with the second highest priority and so on.

$$\Gamma := \{\tau_1, \ldots, \tau_n\}$$

## 3 Response-Time Analysis

The response time $r_{\tau_i,k}$ of the $k$-th job of task $\tau_i$

$$r_{\tau_i,k} := \vartheta_{\tau_i,k} - \Delta t_{\tau_i,k}$$

is the length of the interval from its request time $\Delta t_{\tau_i,k}$ up to its completion time $\vartheta_{\tau_i,k}$. For the worst-case response time of a job

$$r^+_{\tau_i,k} := \vartheta^+_{\tau_i,k} - \Delta t^-_{\tau_i,k}$$

the earliest possible request time $\Delta t^-_{\tau_i,k}$ and the latest possible completion time $\vartheta^+_{\tau_i,k}$ is assumed. The worst-case response time of task $\tau_i$ is the maximum of the worst-case response times of its jobs. To compute it only the jobs in the busy window have to be considered as shown in [5].

$$r^+_{\tau_i} := \max_{k \in [1,m]} \left\{ r^+_{\tau_i,k} \right\} \quad m := \min_{k \in \mathbb{N}} \left\{ k | \vartheta^+_{\tau_i,k} \leq \Delta t^-_{\tau_i,k+1} \right\}$$

### 3.1 Request time

First we will show the bound for the number of events that can occur in an interval of length $\Delta t$ with following lemma:

**Lemma 1.** *Given the clock function bound $\omega^+$ of the clock that generated the event sequence which is bounded by the event function bound $\eta^+$ then the maximum number of events that can occur in an interval of length $\Delta t$ is bounded by $\eta^+(\omega^+(\Delta t))$.*

$$\forall c \, \forall t \, \forall \Delta t \quad \eta(c, \omega(t, \Delta t)) \leq \eta^+(\omega^+(\Delta t))$$

*Proof.*

$$\eta(c,\omega(t,\Delta t)) \le \eta^+(\omega(t,\Delta t)) \le \eta^+(\omega^+(\Delta t))$$

$\eta(c,\omega(t,\Delta t)) \le \eta^+(\omega(t,\Delta t))$ follows directly from (4) and $\eta^+(\omega(t,\Delta t)) \le \eta^+(\omega^+(\Delta t))$ follows from (1) and (6). $\square$

With the bound given by lemma 1 we now show the earliest possible request time $\Delta t^-_{\tau_i,k}$ with following lemma:

**Lemma 2.** *Given the clock function upper bound $\omega^+$ and the event function upper bound $\eta^+$, then the smallest possible interval in which $k$ events can occur is bounded by:*

$$\Delta t^-_{\tau_i}(k) := \inf_{\Delta t \ge 0} \left\{ \Delta t \,\middle|\, \eta^+_{\tau_i}(\omega^+_{\tau_i}(\Delta t)) \ge k \right\} \tag{7}$$

*Proof.* Assume a smaller interval than $\Delta t^-(\tau_i, k)$ exists in which $k$ events occur.

$$\exists c_0 \exists t_0 \exists \Delta t_0 < \Delta t^-_{\tau_i}(k) : k \le \lim_{\Delta t \to \Delta t_0^+} \eta(c_0, \omega(t_0, \Delta t))$$

Due to $\Delta t^-_{\tau_i}(k)$ being the infimum of all intervals in which at least $k$ events occur, it follows that for any interval less than $\Delta t^-_{\tau_i}(k)$ less than $k$ events occur.

$$\forall \Delta t_0 < \Delta t^-_{\tau_i}(k) \qquad \lim_{\Delta t \to \Delta t_0^+} \eta^+_{\tau_i}(\omega^+_{\tau_i}(\Delta t)) < k$$

It follows

$$\begin{aligned} k &\le \lim_{\Delta t \to \Delta t_0^+} \eta(c_0, \omega(t_0, \Delta t)) \\ &\le \lim_{\Delta t \to \Delta t_0^+} \eta^+_{\tau_i}(\omega^+_{\tau_i}(\Delta t)) < k \end{aligned}$$

which is a contradiction. $\square$

## 3.2 Completion time

For the latest possible completion time of the $k$-th job of task $\tau_i$ we first define:

$$f_{\tau_i,k}(\Delta t) := k \cdot c^+_{\tau_i} + \sum_{j=1}^{i-1} \rho^+_{\tau_j}(\Delta t)$$

where $\rho^+_{\tau_i}$ denotes the upper bound of the request function, the maximum amount of processing time which can be requested in an interval of length $\Delta t$. This is the maximum number of events that can occur in an interval of length $\Delta t$ times the worst-case execution time $c^+_{\tau_j}$.

$$\rho^+_{\tau_j}(\Delta t) := \eta^+_{\tau_j}(\omega^+_{\tau_j}(\Delta t)) \cdot c^+_{\tau_j}$$

Since the response-time analysis belongs to the family of busy window algorithms, the completion time of the $k$-th job of task $\tau_i$ is the smallest fix-point of $f_{\tau_i,k}(\Delta t)$

$$\vartheta^+_{\tau_i,k} := \min_{\Delta t \ge 0} \{ \Delta t \,|\, \Delta t = f_{\tau_i,k}(\Delta t) \} \tag{8}$$

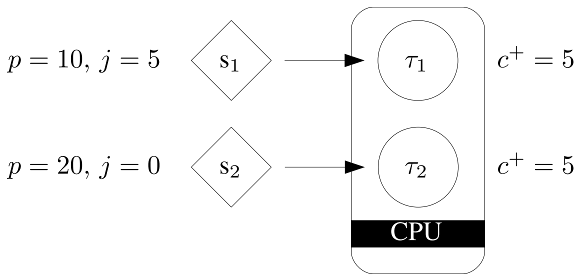

**Figure 1. Example system**

Fix-point (8) can be calculated by following iterative procedure

$$\Delta t^{\#n+1}_{\tau_i,k} = \begin{cases} f_{\tau_i,k}(0) & \text{if } n = 0 \\ f_{\tau_i,k}(\Delta t^{\#n}_{\tau_i,k}) & \text{if } n > 0 \end{cases} \tag{9}$$

Note that iteration (9) can be started with any value less than or equal to fix-point (8). Furthermore note that (8) is only valid for jobs within the busy window of $\tau_i$ as described by Lehoczky [5]. A general proof that (8) exists and that (9) converges to (8) can be found in [7].

## 3.3 Example

In the following example we will show the effect of an asynchronous system on the response time. First we assume the system to be synchronous and then we change the speed of one clock to make it asynchronous. In both cases we will perform a response-time analysis. The system we will use is shown in Fig. 1. It consists of one processor executing two tasks. $\tau_1$ has a higher priority than $\tau_2$ and both tasks have a worst-case execution time of 5 ms. Each task receives events by its own source. Source $s_1$ generates an event every 10 cycles and has a jitter of 5 cycles. Source $s_2$ generates events every 20 cycles and has no jitter. Furthermore source $s_2$ has a clock with 1 kHz generating one cycle every millisecond.

For the synchronous case we assume that source $s_1$ also has a clock with 1 kHz. Every component operates synchronous on the global time base. The task set we are using is defined as follows:

$$\Gamma := \left\{ \left( \left\lceil \frac{\Delta t}{1} \right\rceil, \left\lceil \frac{\Delta c + 5}{10} \right\rceil, 5, 1 \right), \left( \left\lceil \frac{\Delta t}{1} \right\rceil, \left\lceil \frac{\Delta c}{20} \right\rceil, 5, 2 \right) \right\}$$

Now we compute the worst-case response time of $\tau_2$. Table 1 shows the iteration for the first job of task $\tau_2$ as given by (9). As can be seen, the fix-point is found at 15 ms. The request time of the second job is 20 ms. Because $15 \le 20$ we are done and the worst-case response time of $\tau_2$ is 15 ms. A gantt-chart for this worst-case scenario is shown in Fig. 2.

For the asynchronous case we assume that the clock of source $s_1$ has changed to 1.25 kHz, generating one cycle every 0.8 milliseconds. Everything else of the system

**Table 1. Fix-point iteration for the first job of task $\tau_2$ (synchronous case)**

| n | $\Delta t^{\#n}_{\tau_2,1}$ | $\omega^+_{\tau_1}(\Delta t^{\#n}_{\tau_2,1})$ | $\eta^+_{\tau_1}(\omega^+_{\tau_1}(\Delta t^{\#n}_{\tau_2,1}))$ | $\Delta t^{\#n+1}_{\tau_2,1}$ |
|---|---|---|---|---|
| 1 | 10 | 10 | 2 | 15 |
| 2 | 15 | 15 | 2 | 15 |

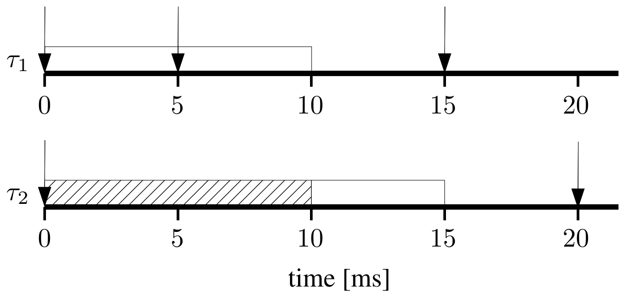

**Figure 2. Gantt-chart of the worst-case scenario (synchronous case)**

stays unchanged. The task set we are using is defined as follows:

$$\Gamma := \left\{ \left( \left\lceil \frac{\Delta t}{0.8} \right\rceil, \left\lceil \frac{\Delta c + 5}{10} \right\rceil, 5, 1 \right), \left( \left\lceil \frac{\Delta t}{1} \right\rceil, \left\lceil \frac{\Delta c}{20} \right\rceil, 5, 2 \right) \right\}$$

Again we compute the worst-case response time of $\tau_2$. Table 2 shows the fix-point iteration for the first job of task $\tau_2$ as given by (9). As can be seen, this time the fix-point is found at 20 ms. With the same argument as in the synchronous case, the worst-case response time of task $\tau_2$ is 20 ms. A gantt-chart for this worst-case scenario in the asynchronous case is shown in Fig. 3.

The faster clock of source $s_1$ causes more events to occur in the same amount of time compared to the synchronous case. Thus $\tau_2$ experiences more interference by task $\tau_1$ leading to an increased worst-case response time. Note that the increased number of events also increased the worst-case response time of task $\tau_1$. It increased from previously 5 ms in the synchronous case to 6 ms in the asynchronous case.

**Table 2. Fix-point iteration for the first job of task $\tau_2$ (asynchronous case)**

| n | $\Delta t^{\#n}_{\tau_2,1}$ | $\omega^+_{\tau_1}(\Delta t^{\#n}_{\tau_2,1})$ | $\eta^+_{\tau_1}(\omega^+_{\tau_1}(\Delta t^{\#n}_{\tau_2,1}))$ | $\Delta t^{\#n+1}_{\tau_2,1}$ |
|---|---|---|---|---|
| 1 | 10 | 13 | 2 | 15 |
| 2 | 15 | 19 | 3 | 20 |
| 3 | 20 | 25 | 3 | 20 |

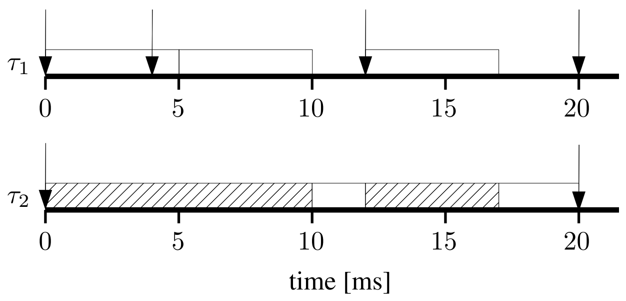

**Figure 3. Gantt-chart of the worst-case scenario (asynchronous case)**

## 4 Conclusion

In this paper we have introduced a new abstract clock model and adapted the response-time analysis for fixed-priority preemptive systems to include this clock model. With an example we have shown how a faster clock affects the response times of tasks in a system. The example also shows the importance of modeling the stimulation of a task correctly. Assuming a synchronous system which actually is asynchronous can lead to too optimistic assertions with the response-time analysis.

In future work we will investigate the possibility of transferring the introduced model to the case where the used processors have different processing capabilities. Furthermore we will investigate how the introduced clock model can be used with the real-time calculus.

## References

[1] S. Chakraborty, S. Künzli, and L. Thiele. A General Framework for Analysing System Properties in Platform-Based Embedded System Designs. In *Design, Automation and Test in Europe Conference and Exhibition*, 2003.

[2] M. Joseph and P. Pandya. Finding Response Times in a Real-Time System. *The Computer Journal*, 29(5):390–395, 1986.

[3] H. Kopetz and G. Grünsteidl. TTP - A Time-Triggered Protocol for Fault-Tolerant Real-Time Systems. In *Digest of Papers: FTCS-23, The Twenty-Third Annual International Symposium on Fault-Tolerant Computing*, pages 524–533, Jun 1993.

[4] H. Kopetz and W. Ochsenreiter. Clock Synchronization in Distributed Real-Time Systems. *IEEE Transactions on Computers*, 36(8):933–940, Aug 1987.

[5] J. P. Lehoczky. Fixed Priority Scheduling of Periodic Task Sets with Arbitrary Deadlines. In *Proceedings of the 11th Real-Time Systems Symposium*, pages 201–209, Dec 1990.

[6] R. Münzenberger, M. Dörfel, R. Hofmann, and F. Slomka. A general time model for the specification and design of embedded real-time systems. *Microelectronics Journal*, 34(11):989–1000, 2003.

[7] S. Stein, J. Diemer, M. Ivers, S. Schliecker, and R. Ernst. On the Convergence of the SymTA/S analysis. Technical report, Technical University Braunschweig, 2008.

[8] K. Tindell and J. Clark. Holistic Schedulability Analysis for Distributed Hard Real-time Systems. *Microprocessing and Microprogramming*, 40:117–134, April 1994.