Holistic Real-Time Analysis with an Expressive Event Model

Steffen Kollmann, Victor Pollex and Frank Slomka
{firstname}.{lastname}@uni-ulm.de
Ulm University

Abstract. Due to complex embedded systems it is no longer possible to find the optimal design solution instantly and a design space exploration is necessary. In each step the real-time constraints of the system have to be verified. This can be performed by a schedulability analysis. To find the optimal solution the real-time analysis must deliver tight results of the tasks' worst-case response times. This can only be done by using appropriate models. The calculation of tight results leads to an increased runtime of the analysis. But due to short design phases fast real-time feasibility tests and analysis algorithms are necessary for a high acceptance of the formal techniques by industrial software engineers. In this paper both the possibility to calculate tight worst-case response times of the tasks in a distributed system and the reduction of the algorithm's runtime is presented. The correctness of the approach is proven analytically and experimental.

1. Introduction

Embedded systems are the most widespread computer systems in the world. The complexity of them increases from generation to generation with the effect that the optimal design solution is no longer obvious. One possibility to analyze different solutions is to perform a design space exploration. In each step of the search different constraints like space, energy, reliability, safety and time of the system can be evaluated and compared to other solutions. One special issue during such an exploration is to verify whether the time constraints of the systems can be met.

Schedulability analysis techniques play an important role to verify the real-time constraints of embedded systems. The idea of such analysis methodologies is to determine an upper bound of the worst-case response times in the system. It is very important that these upper bounds are as tight as possible, because overestimated response times lead to oversized processors and therefore to worse design solutions concerning energy and space. To determine worst-case response times an event model to describe the stimulation in a distributed system is necessary. A common approach is to assume an event-triggered system where each task generates an event at the end of its execution to notify other tasks. By means of the event model and the worst-case and best-case execution time of the tasks the worst-case response times can be determined by a global fixpoint iteration over the whole system. To accomplish tight results of the response times an expressive event model is necessary which is able to describe the stimulation in a system exactly. Based on the event stream model presented in [6] we will show how tight response times can be determined and compare these results with well known methodologies like Tindell and Clark [15] and SymTA/S [12].

Since the calculation of tight response times leads to an increased runtime of the real-time analysis we will also show that the runtime can be improved despite using an expressive event stream model. The idea of the improvement is to define an upper bound and a lower bound for the search interval which calculates the worst-case response time of a task.

The paper is organized as follows: Section 2 gives an overview about related work. The model is presented in section 3. The necessary real-time analysis is described in section 4. In section 5 we introduce our runtime improved real-time analysis concerning the worst-case response time. The work closes with experiments and a conclusion.

2. Related Work

Many different approaches exist to determine the feasibility of a real-time system. Liu and Layland [9] have developed a sufficient feasibility test for strict periodic task sets scheduled by the rate monotonic policy. Since then a lot of conditions for sufficient feasibility tests have been found. A good overview is given in [17]. Recently, Bini and Baruah [2] have presented an upper bound condition for the worst-case response time of a task. By means of this bound condition a new sufficient feasibility analysis for periodic tasks has been introduced.

Lehoczky [8] has introduced a worst-case response time analysis with arbitrary deadlines. Sjödin and Hansson [13] have proposed some lower bound conditions for the starting job of this test to improve its performance. An approach to reduce the number of jobs during a worst-case response time analysis by upper bound condition has been introduced by Pollex et al. [11]. In [11] the sufficient feasibility analysis from Bini and Baruah [2] has been used to derive an upper bound condition for the worst-case response time. The analysis is based on the periodic model with jitter.

Other approaches have also been considered to improve the runtime of the worst-case response time. In [4], [5] and [10] some techniques are proposed in order to reduce the number of iterations during the analysis. But these approaches do not consider a job number reduction.

For distributed real-time systems many different models which can specify the stimulation of a system have been developed. The most popular is the analysis by Tindell and Clark presented in [15]. Based on the periodic model with jitter (p, j) Tindell et al. introduces in [15] the analysis for distributed real-time systems.

Later, Richter presents in [12] an extension of this model by introducing a minimal distance between events (p, j, d). This parameter d has the effect that if the jitter is greater than the period the events cannot occur simultaneously. This results finally in tighter worst-case response times.

The lack of both approaches is that the used event models are not able to describe arbitrary stimulation in the system exactly due to the few parameters of the models. Another big disadvantage is that the jitter parameter can only grow during the analysis, meaning that events can only occur closer. A relaxation of the jitter is not provided. But this is important for a tight real-time analysis.

The real-time calculus [16] has no discrete model in order to describe the occurrence of events in a system. This technique consists of approximated curves describing the arrival of events and the capacity of the processors. Since this methodology bases on the network calculus [3], it is not possible to apply the runtime improvement presented in section 5 to this methodology.

3. System Model

3.1. Task Model

Γ is the set of tasks on one resource $\Gamma = \{\tau_1, ..., \tau_n\}$. A task $\tau = (c^+, c^-, d, \phi, \Theta^+)$ consists of c^+ its worst-case execution time, c^- its best-case execution time, d its deadline, ϕ its priority for the scheduling (the lower the number the higher the priority) and Θ^+ defines the maximum stimulation (maximum number of events in an interval). Let $\tau_{i,j}$ be the j-th job/execution of task τ_i. Each job generates an event at the end of its execution to trigger other tasks. On each resource we assume a fixed-priority schedule.

3.2. Event Model

Event streams have been first defined in [6]. The basic idea is to define an event function $\eta(\Delta t, \Theta)$ which describes the maximum amount of events occurring in an interval of length Δt. The event function needs a properly described model which makes it easy to extract the information.

3.2.1. Maximum Event Streams: The idea of the maximum event streams is to write for any given number of events the minimum interval in which this number of events can occur. Therefore we get an interval for one event, two events and so on.

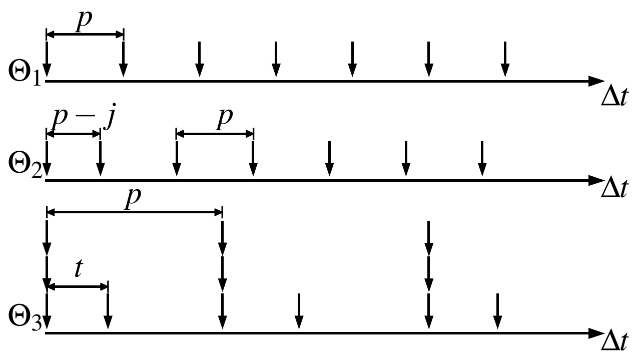


Fig. 1: Example of three different event streams

The length of the interval for one event is infinitesimal small and therefore considered to be zero. The result is a sequence of interval lengths showing a non-decreasing behavior. The reason for this behavior is, that the minimum interval length for n events cannot be smaller than the minimum interval length for $n-1$ events since the first interval also includes $n-1$ events.

Definition 1: A maximum event stream is a set of event stream elements $\theta : \Theta^+ = \{\theta_1, \theta_2, ..., \theta_n\}$ and each event stream element $\theta = (p, a)$ consists of an offset-interval a and a period p. The event function of the maximum event stream has the property of sub-additivity: $\eta(\Delta t_1 + \Delta t_2, \Theta^+) \leq \eta(\Delta t_1, \Theta^+) + \eta(\Delta t_2, \Theta^+)$.

This means that the maximum number of events of an interval cannot exceed the cumulated maximum number of events of its subintervals. Each event stream element θ describes a set of interval lengths $\{a_\theta + k \cdot p_\theta | k \in \mathbb{N}\}$ of the sequence. The event function is defined as follows:

Definition 2: The event function calculates the number of events for a given event stream Θ and a given length of the interval Δt:

$$\eta(\Delta t, \Theta) = \sum_{\substack{\theta \in \Theta \\ a_\theta \leq \Delta t}} \left\lceil \frac{\Delta t - a_\theta}{p_\theta} \right\rceil \tag{1}$$

Events that occur exactly at the end of the interval $\Delta t = [t_1, t_2)$ have not to be considered, because the executions of the considered tasks have just finished before the new events arrive. The inverse function to the event function is the interval function.

Definition 3: The interval function calculates the smallest length of the interval in which the given number of events n occur in the given event stream Θ:

$$\Delta t(n, \Theta) = inf\{\Delta t | \eta(\Delta t, \Theta) \geq n\} \tag{2}$$

With an infinite (∞) period it is possible to model irregular behaviour of periodic events. A detailed definition of the concept and the mathematical foundation of the event streams can be found in [1].

Example 1: In figure 1 some examples for event streams can be found. The first one Θ_1^+ = {(p,0)} has a strictly periodic stimulus with a period p. The second example Θ_2^+ = {(∞,0), (p,p-j)} shows a periodic stimulus in which the single events can jitter within a interval of size j. In the third example $\Theta_3^+ = \{(p,0),(p,0),(p,0),(p,t)\}$ three events occur at the same time and the fourth occurs after a time t. This pattern is repeated with a period of p. Event streams can describe all these examples in an easy and intuitive way.

3.2.2. Normalized Event Streams: The introduced model allows an event stream to be described in several ways. For an efficient implementation of the approach we define a normalized event stream as introduced in [7]. By means of this normalization the interval function can be efficiently computed. For a detailed description how to transform an event stream to a normalized event stream and how the interval function can be described efficiently see [7]. In the following we use the definition:

Definition 4: A normalized event stream Θ has the form:

$$\{(\infty, a_1), \ldots, (\infty, a_m), (p, a_{m+1}), \ldots, (p, a_n)\} : (0 \leq m \leq n \wedge a_i \leq a_j \Leftrightarrow i \leq j) \tag{3}$$

We define further that $N_\infty = m$ is the number of aperiodic elements of an event stream and $N_p = n - m$ is the number of periodic elements of an event stream. With definition 4 we can formulate the utilization of a task τ as follows:

Definition 5: The utilization of a task τ and its normalized event stream Θ_τ^+ is defined as:

$$U_\tau = \frac{N_p \cdot c_\tau^+}{p} \tag{4}$$

4. Real-Time Analysis

In order to get tighter bounds for the response times we combine the event stream model with the real-time analysis from Tindell and Clark presented in [15]. As described in [15] in each global iteration step of the real-time analysis the worst-case response time and the outgoing stimulation for each task in the system are computed until a fix-point is found. The convergence of the global fix-point iteration is proven in [14].

First we have to show how the worst-case response time analysis can be performed and then how the outgoing event streams can be determined. For this we define the term busy-window:

Definition 6: The busy-window of a task τ is the smallest interval greater zero in which the execution demand of the jobs of the task τ itself or jobs of higher priority tasks $\forall \tau_i \in \Gamma : \phi_{\tau_i} < \phi_\tau$ is completely executed.

4.1. Worst-Case Response Time

To calculate the worst-case response time of a task τ we have adapted the approach from [8].

Definition 7: If for all tasks $\forall\tau \in \Gamma$ the worst-case response time is smaller or equal to its deadline $r^+(\tau) \leq d_\tau$, the task set Γ is feasible and the real-time analysis is successful. The worst-case response time of a task considering event streams can be calculated by:

$$\begin{aligned} r^+(\tau) &= \max_{k\in[1,m]}\{r^+(k,\tau) - \Delta t(k,\Theta_\tau^+)\} \; m = \min_{n\in\mathbb{N}}\{n|r^+(n,\tau) \leq \Delta t(n+1,\Theta_\tau^+)\} \\ r^+(k,\tau) &= \min\{\Delta t|\Delta t = k\cdot c_\tau^+ + \sum_{\tau'\in\Gamma_{HP}} \eta(\Delta t,\Theta_{\tau'}^+)\cdot c_{\tau'}^+\} \end{aligned}$$

The equations are similar to the definition of the worst-case response time introduced in [8]. Only the calculation of higher priority tasks (Γ_{HP}) has been changed, because we use the event stream model.

Assuming that Δt is the smallest length of an interval in which all the execution demand of the considered tasks have been processed $k\cdot c_\tau^+$. During the fix-point iteration we have to verify whether more interrupts in this interval from the considered tasks can occur or not. So the amount of execution demand produced by higher priority tasks can be calculated by the event function $\eta(\Delta t,\Theta_{\tau'}^+)$ multiplied by the worst-case execution time c^+. By means of a fix-point iteration the worst-case response time can be calculated for every job k in the busy-window as shown in [8]. The convergence of the fix-point iteration is generally proven in [14].

4.2. Outgoing Event Density

Now, we will show how it is possible to determine the event density a task can produce. For this we have to identify when the worst-case occurs. In order to prove our presumptions we define the following terms:

Definition 8: The completion time $r^\pm(n,\tau)$ of the n-th job is the interval from the request of the first job up to the point in time where the n-th job has finished its execution. The response time of a job is the completion time $r^\pm(n,\tau)$ minus the request time $\Delta t(n,\Theta_\tau^+)$.
Now we define how it is possible to determine the maximum event density which can be produced by a task.

Lemma 1: A number of outgoing events occur in maximum density when the first event occurs as late as possible and all further events occur as early as possible. So the maximum event density is bounded by:

$$\Delta t_{min}(n,\tau) = \begin{cases} 0 & n=1 \\ max(\Delta t(n,\Theta_\tau^+), r^+(\tau) + \Delta t_{min}(n-1,\tau)) + c_\tau^- - r^+(\tau) & n>1 \end{cases} \tag{5}$$

Proof: The proof is a proof by contradiction and we have to show that it exists no interval Δt lower than the interval in the assumption.

Case 1 ($n=1$): According to the maximum event stream definition one event occurs in the interval 0.

Case 2 ($n>1$): Assuming an interval Δt which is smaller than $max(\Delta t(n,\Theta_\tau^+), r^+(\tau) + \Delta t_{min}(n-1,\tau)) + c_\tau^- - r^+(\tau)$ and $i,j \in \mathbb{N} : j-i+1 = n$. Then it follows:

$$\begin{aligned} \Delta t &< \Delta t(n,\Theta_\tau^+) + c_\tau^- - r^+(\tau) \\ \Delta t(j,\Theta_\tau^+) + (r^\pm(j,\tau) - \Delta t(j,\Theta_\tau^+)) - (\Delta t(i,\Theta_\tau^+) + (r^\pm(i,\tau) - \Delta t(i,\Theta_\tau^+)) &< \Delta t(n,\Theta_\tau^+) + c_\tau^- - r^+(\tau) \end{aligned}$$

This is a contradiction, since $\Delta t(j,\Theta_\tau^+) - \Delta t(i,\Theta_\tau^+) < \Delta t(n,\Theta_\tau^+)$ violates the sub-additivity from definition 1, $r^\pm(j,\tau) - \Delta t(j,\Theta_\tau^+) < c_\tau^-$ the definition of the best-case execution time and $r^\pm(i,\tau) - \Delta t(i,\Theta_\tau^+) > r^+(\tau)$ the definition of the worst-case response time. So we have to consider

$$\begin{aligned} \Delta t &< (r^+(\tau) + \Delta t_{min}(n-1,\tau)) + c_\tau^- - r^+(\tau) \\ \Delta t(j,\Theta_\tau^+) + (r^\pm(j,\tau) - \Delta t(j,\Theta_\tau^+)) & \\ -(\Delta t(i,\Theta_\tau^+) + (r^\pm(i,\tau) - \Delta t(i,\Theta_\tau^+)) &< (r^+(\tau) + \Delta t_{min}(n-1,\tau)) + c_\tau^- - r^+(\tau) \end{aligned}$$

From the discussion above we know $\Delta t(n,\Theta_\tau^+)$ is a bound for $\Delta t(j,\Theta_\tau^+) - \Delta t(i,\Theta_\tau^+)$ and it follows:

$$\Delta t(n,\Theta_\tau^+) + (r^\pm(j,\tau) - \Delta t(j,\Theta_\tau^+)) - (r^\pm(i,\tau) - \Delta t(i,\Theta_\tau^+)) < (r^+(\tau) + \Delta t_{min}(n-1,\tau)) + c_\tau^- - r^+(\tau)$$

But this is also a contradiction, since the n-th job cannot be executed until the previous job has been finished $\Delta t(n,\Theta_\tau^+) < (r^+(\tau) + \Delta t_{min}(n-1,\tau))$, $r^\pm(j,\tau) - \Delta t(j,\Theta_\tau^+) < c_\tau^-$ violates the definition of the best-case execution time and $r^\pm(i,\tau) - \Delta t(i,\Theta_\tau^+) > r^+(\tau)$ the definition of the worst-case response time. So equation (5) bounds the length of the interval in which n events can occur. ■

By means of equation (5) it is possible to calculate the outgoing event streams in the system. For an efficient implementation of the interval function the normalized event streams are used. Lemma 1 in conjunction with the methodologies introduced in [7] extracts the concrete outgoing event stream. In an analogous way we could define minimal event streams and the corresponding best-case response time analysis. But for brevity these definitions and lemma are omitted.

5. Runtime Improved Real-Time Analysis

Next, we will show how the runtime of the analysis can be improved. Thereby we focus on reducing the number of jobs that have to be considered during a worst-case response time analysis in a busy-window as shown in figure 2.

Sjoedin and Hansson have presented in [13] several methods on how to reduce the run-time complexity of the response-time analysis. The main idea of their presented methods is to define lower bound conditions where the analysis can start without changing the result of the analysis. This allows to skip every evaluation up to these lower bound conditions. This has been improved in [11] and is adapted in section 5.1 to the event stream model.

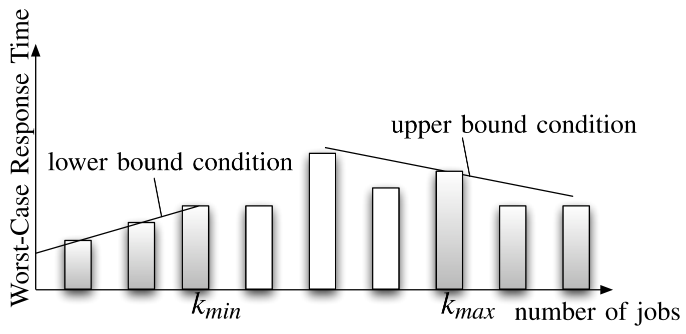


Fig. 2: Job reduction by upper and lower bounds.

We also introduce in section 5.2 a condition which allows us to stop evaluating jobs as early as possible. It is based on the upper bound condition presented in [2] for a sufficient analysis by Bini and Baruah. This bound has been extended in [11] to incorporate task jitter and used to improve the performance of the exact analysis.

5.1. Lower Bound Condition

An improvement of the lower bound condition k_{min} is given in [11]. The idea of this approach is the following: If between two successive events the distance is smaller or equal to the worst-case execution time of the task, the response time of the first event is smaller or equal to the response time of the second event. So we can determine the lower bound condition by the following lemma:

Lemma 2: The starting job of the worst-case response time can be described by:

$$k_{min} = \min_{k \in \mathbb{N}} \{k | \Delta t(k+1,\Theta_\tau^+) - \Delta t(k,\Theta_\tau^+) > c^+\} \tag{6}$$

Proof: First we show, if the distance between any two events (k and $k+1$) is smaller than the worst-case execution time then the response time of the $k+1$st job is greater than the response time of the $k-$th job. We start with the condition from the lemma 2:

$$\begin{array}{rrcl} & \Delta t(k+1,\Theta^+) - \Delta t(k,\Theta^+) & \leq & c^+ \\ \Rightarrow & \Delta t(k+1,\Theta^+) - \Delta t(k,\Theta^+) & \leq & r^+(k+1,\tau) - r^+(k,\tau) \\ \Leftrightarrow & r^+(k,\tau) - \Delta t(k,\Theta^+) & \leq & r^+(k+1,\tau) - \Delta t(k+1,\Theta^+) \end{array}$$

It follows that the response time of the $k-$th job is smaller than the response time of the $k+1$st job. So we can conclude that the following implication holds:

$$\Delta t(k+1,\Theta^+) - \Delta t(k,\Theta^+) \le c^+ \Rightarrow r^+(k,\tau) - \Delta t(k,\Theta^+) \le r^+(k+1,\tau) - \Delta t(k+1,\Theta^+)$$

Since the condition is an implication we are not able to determine what happens when the condition does not hold. So we can only conclude that the first jobs for which the proposition $\Delta t(k+1,\Theta^+) - \Delta t(k,\Theta^+) \le c^+$ holds can be omitted for the worst-case response time analysis. ■

We can now integrate this new bound into the worst-case response time analysis as follows:

Definition 9: The worst-case response time can start at the job k_{min}

$$r^+(\tau) = \max_{k \in [k_{min}, m]} \{r^+(k,\tau) - \Delta t(k,\Theta^+_\tau)\} \; m = \min_{n \in \mathbb{N}} \{n | r^+(n,\tau) \le \Delta t(n+1,\Theta^+_\tau)\} \tag{7}$$

5.2. Upper Bound Condition

After improving the lower bound condition of the starting job of the tasks, we will now derive a new condition which will allow us to skip the evaluation of all remaining jobs from the point where the new condition holds. The new approach is based on the upper bound condition for the response time given in [2]. This upper bound condition is adapted to the event stream model by extending the upper bound condition of the workload. Bini and Baruah presented in [2] a sufficient feasibility test, in which they characterize the completion time as follows:

$$I_i^k = X_{i-1}(k \cdot c^+_\tau) \tag{8}$$

where $X_i(h) = min_t\{t : H_i(t) \ge h\}$, with $H_i(t) = t - W_i(t)$ being the worst-case idle time and $W_i(t)$ being the worst-case workload of the i highest priority tasks over an interval of length t [2].

To adapt this technique to *normalized* event streams we will show how a sufficient test with event streams can be performed.

Lemma 3: An upper bound for the completion time of a task with normalized event streams is given by:

$$r^{up}(k,\tau) = \frac{kc^+_\tau - \sum\limits_{\tau_j \in \Gamma_{HP}} \max\limits_{N_\infty < l \le |\Theta^+_{\tau_j}|} (U_{\tau_j}(\Delta t(l,\Theta^+_{\tau_j}) + c^+_{\tau_j}) - lc^+_{\tau_j})}{1 - \sum\limits_{\tau_j \in \Gamma_{HP}} U_{\tau_j}} \tag{9}$$

and therefore an upper bound for the response time is: $r^{up}(k,\tau) - \Delta t(k,\Theta^+_\tau)$.

Proof: An upper bound condition of the workload is a linear function given by $w(t) = Ut + x$. To calculate the upper bound condition for a given task it is necessary to compute the constant x, which is performed by solving the equation $w(\Delta t(k,\Theta^+_\tau) + c^+_\tau) = k \cdot c^+_\tau$ giving us the following value: $x = -U_\tau \cdot (\Delta t(k,\Theta^+_\tau) + c^+_\tau) + k \cdot c^+_\tau$

At this point we have to show which intervals Δt and which k we have to consider in order to determine the constant. Since we can have more than only one periodic element in a normalized event stream, we have to consider each of them. This is founded by the fact, that we need the greatest value for x, so that the resulting straight line is always over the originally curve. So it follows: $x = \max\limits_{N_\infty < l \le |\Theta^+_{\tau_j}|} (-U_{\tau_j}(\Delta t(l,\Theta^+_{\tau_j}) + c^+_{\tau_j}) + lc^+_{\tau_j})$

With this constant we are able to transform the upper bound of the workload like Bini and Baruah have shown in [2]. So we get for the workload:

$$W^{\text{ub}}_\tau(\Delta t) = \sum_{\tau_j \in \Gamma} U_{\tau_j} \Delta t - \sum_{\tau_j \in \Gamma} \max_{N_\infty < l \le |\Theta^+_{\tau_j}|} (U_{\tau_j}(\Delta t(l,\Theta^+_{\tau_j}) + c^+_{\tau_j}) - lc^+_{\tau_j})$$

From this the idle time follows as defined in [2]:

$$H^{\text{lb}}_\tau(\Delta t) = \Delta t(1 - \sum_{\tau_j \in \Gamma} U_{\tau_j}) + \sum_{\tau_j \in \Gamma} \max_{N_\infty < l \le |\Theta^+_{\tau_j}|} (U_{\tau_j}(\Delta t(l,\Theta^+_{\tau_j}) + c^+_{\tau_j}) - lc^+_{\tau_j})$$

We then get the pseudo-inverse from the idle time as defined in [2]:

$$X_\tau^{\mathrm{ub}}(h) = \frac{h - \sum\limits_{\tau_j \in \Gamma} \max\limits_{N_\infty < l \leq |\Theta_{\tau_j}^+|} (U_{\tau_j}(\Delta t(l, \Theta_{\tau_j}^+) + c_{\tau_j}^+) - l c_{\tau_j}^+)}{1 - \sum\limits_{\tau_j \in \Gamma} U_{\tau_j}}$$

This leads finally to the upper bound condition:

$$r^{up}(k, \tau) = \frac{k \cdot c_\tau^+ - \sum\limits_{\tau_j \in \Gamma_{HP}} \max\limits_{N_\infty < l \leq |\Theta_{\tau_j}^+|} (U_{\tau_j}(\Delta t(l, \Theta_{\tau_j}^+) + c_{\tau_j}^+) - l c_{\tau_j}^+)}{1 - \sum\limits_{\tau_j \in \Gamma_{HP}} U_{\tau_j}}$$

■

By means of this upper bound we are able to define a sufficient real-time test with event streams:

Lemma 4: If the following condition holds for all elements in an event stream the task τ will meet its deadline:

$$1 \leq k \leq |\Theta_\tau^+| : r_{up}^+(k, \tau) - \Delta t(k, \Theta_\tau^+) \leq d_\tau \tag{10}$$

Proof: We have to show how many jobs have to be considered in the busy-window. Since we assume that we have to perform the test for each tuple in the event stream only once, we only have to show that the approximation of the jobs produced by a periodic tuple is monotonically non-increasing. So the following condition must hold:

$$r^{up}(k + N_p, \tau) - \Delta t(k + N_p, \Theta_\tau^+) \leq r^{up}(k, \tau) - \Delta t(k, \Theta_\tau^+) \tag{11}$$

Since we consider the same tuple, we have the same constant and therefore we can omit the max operation. Further note that $\Delta t(k + N_p, \Theta_\tau^+) - \Delta t(k, \Theta_\tau^+) = p$. So we can transform the assumption as follows:

$$\begin{aligned}
\frac{(k+N_p) \cdot c_\tau^+ - \sum\limits_{\tau_j \in \Gamma_{HP}} (U_{\tau_j}(\Delta t + c_{\tau_j}^+) - l c_{\tau_j}^+)}{1 - \sum\limits_{\tau_j \in \Gamma_{HP}} U_{\tau_j}} - \Delta t(k + N_p, \Theta_\tau^+) &\leq \frac{k \cdot c_\tau^+ - \sum\limits_{\tau_j \in \Gamma_{HP}} (U_{\tau_j}(\Delta t + c_{\tau_j}^+) - l c_{\tau_j}^+)}{1 - \sum\limits_{\tau_j \in \Gamma_{HP}} U_{\tau_j}} - \Delta t(k, \Theta_\tau^+) \\
\frac{(k+N_p) \cdot c_\tau^+}{1 - \sum\limits_{\tau_j \in \Gamma_{HP}} U_{\tau_j}} &\leq \frac{k \cdot c_\tau^+}{1 - \sum\limits_{\tau_j \in \Gamma_{HP}} U_{\tau_j}} + p \\
\frac{N_p c_\tau^+}{1 - \sum\limits_{\tau_j \in \Gamma_{HP}} U_{\tau_j}} &\leq p \\
\frac{N_p c_\tau^+}{p} + \sum_{\tau_j \in \Gamma_{HP}} U_{\tau_j} &\leq 1
\end{aligned}$$

It follows that we have an utilization less or equal than one and therefore our assumption is true. This finally means that only the first job produced by a periodic element has to be considered and therefore each element has to be considered only once. ■

Now we can finally formulate the condition which will reduce the number of jobs which have to be considered by the worst-case response time analysis.

Corollary 1: The remaining jobs in the busy-period can be omitted when the following condition holds:

$$k_{max} = \min_{k > N_\infty} \{k | \forall_{k \leq j < k + N_p} : r^{up}(j, \tau) - \Delta t(j, \Theta_\tau^+) \leq \max_{1 \leq x \leq j} (r^+(x, \tau) - \Delta t(x, \Theta_\tau^+))\} \tag{12}$$

Proof: The approximation applies only for periodic elements. So the worst-case response times for aperiodic elements must be computed exactly ($k \leq N_\infty$). Since we have shown in lemma 4 that the approximation for the jobs generated by one element is monotonically non-increasing, we have to check whether the condition holds for every periodic element. If the condition holds for every periodic element the worst-case response time has been found an the remaining jobs can be omitted. ■

We can now integrate this new bound into the worst-case response time analysis as follows:

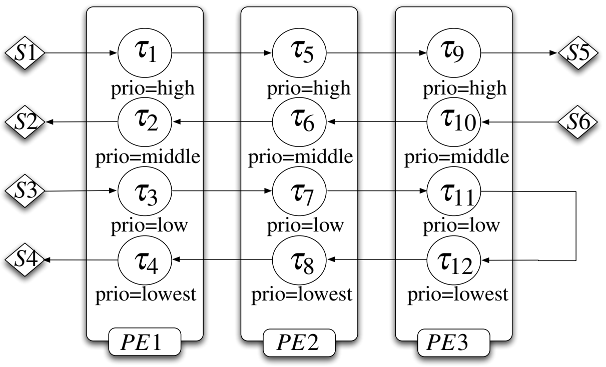
Definition 10: The number of jobs considered during a worst-case response time analysis is bounded by the condition:

$$r^+(\tau) = \max_{k\in[k_{min},m]} \{r^+(k,\tau) - \Delta t(k,\Theta_\tau^+)\}\ m = min(k_{max}, \min_{n\in\mathbb{N}}\{n|r^+(n,\tau) \leq \Delta t(n+1,\Theta_\tau^+)\}) \tag{13}$$

6. Experiments

Next, we will present the results of the experiments we have conducted. The system depicted in figure 3 consists of twelve tasks which are distributed equally between three processing elements. Their priorities are also described in figure 3. The best-case and worst-case execution time of each task is listed in the table in figure 3.

The system has been analyzed with the periodic model with jitter [15] (PJ), the periodic model with jitter and minimal distance [12] (PJD) and the event stream model (ES) as introduced in this paper. We compared the resulting average worst-case response times over all tasks and the runtime of the different models. In the experiments the average system utilization has been varied from 50% to 99.5% (step size 0.5) by modifying the input stimulation S1, S3 and S6. In order to vary the input stimulation only the period and the jitter have been modified and mapped to the different event models. The jitter was up to five times the period. For each utilization step the average of 100 variations has been taken.



τ	1	2	3	4	5	6	7	8	9	10	11	12
c^- [$t.u.$]	20	20	1500	300	5	8	1000	500	5	4	900	1000
c^+ [$t.u.$]	20	20	1500	300	5	16	1000	1000	15	4	1200	1100

Fig. 3: System used for the experiments

For each event model four variations were used. One with the lower bound as introduced in section 5.1, one with the upper bound as introduced in section 5.2, one with both bounds and one with no bounds. For each model the variation with no bounds is used as reference for the other variations regarding their runtime. For all variations the runtime was measured as well as the average calculated worst-case response time.

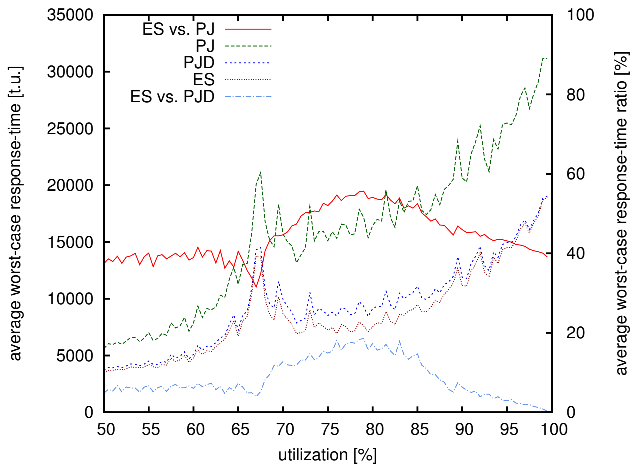
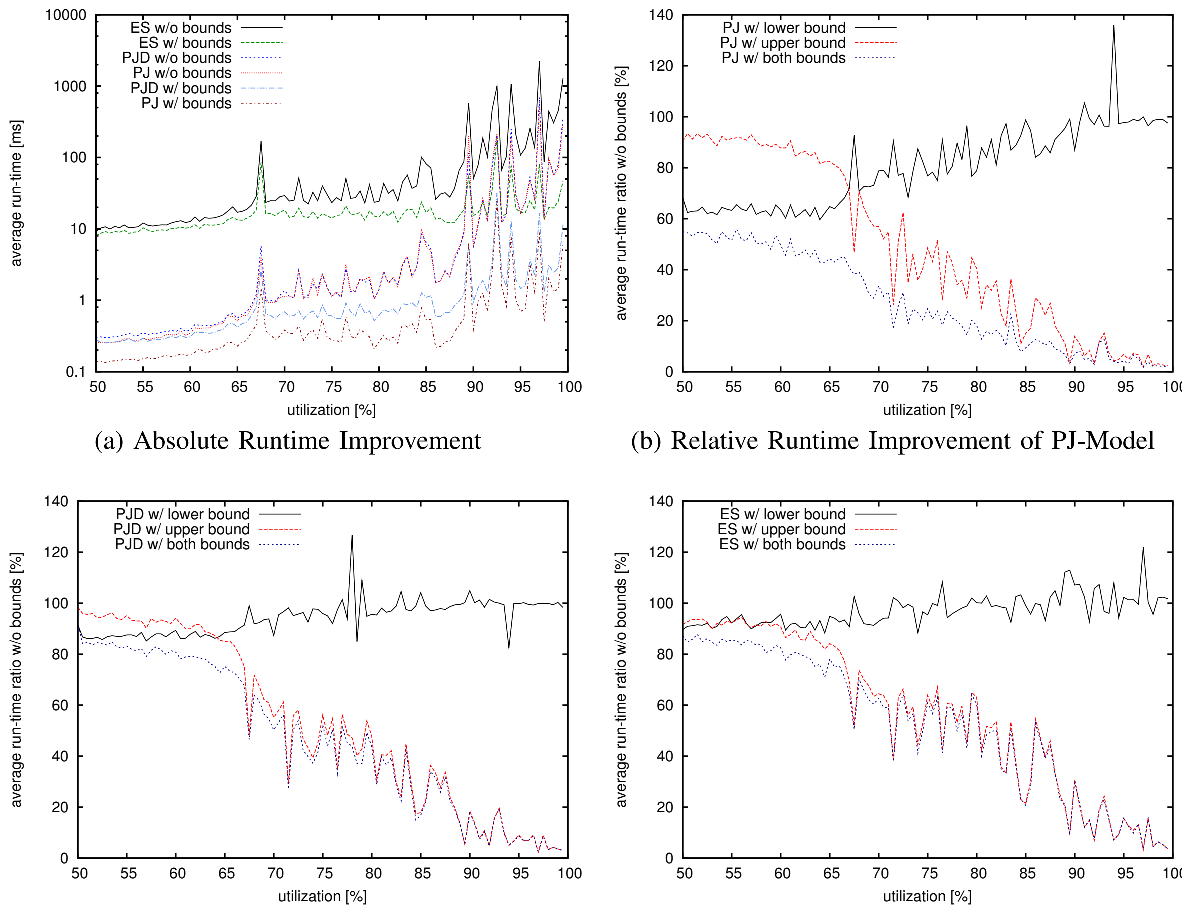


Fig. 4: Average wcrt (Absolute and Ratio)

In figure 4 the average worst-case response-time over all tasks for each event model is shown. Also the relative improvements of the event stream model compared to the other two models are shown. As expected the PJD-Model delivers better results than the PJ-model and the event stream model delivers better results than both the PJD-model and the PJ-model. Especially between a utilization of 67% to 89% we have a significant improvement of the worst-case response times of the event stream model. The ES-model computes up to 19% lower worst-case response-times than the PJD-model and up to 56% lower response times compared to the PJ-model.

The result is that the usage of the event stream model during a design space exploration can lead to improved response times and thus a slower processor can be used eventually reducing energy consumption and costs.



(a) Absolute Runtime Improvement

(b) Relative Runtime Improvement of PJ-Model

(c) Relative Runtime Improvement of PJD-Model

(d) Relative Runtime Improvement of ES-Model

Fig. 5: Comparison of the runtimes of the different models

An overview of the average runtime that was needed by the reference and the variation using both bounds of each event model is shown in figure 5a. It can be clearly seen, that the event stream model needs more time than the other models. This is due to the fact that event streams are able to describe the event density more accurately. In contrast both periodic models need considerable less time due to their simpler model. Nonetheless reducing the number of jobs that have to be considered in the analysis by the introduced bounds has a big impact on the runtime of all three models. Especially when the utilization of the system is very high the runtime is reduced significantly.

For a better understanding of the job reduction from section 5 we have considered the lower and upper bound separately. In figure 5b the PJ-model is considered. By using only the lower bound, the runtime is reduced to about 60 percent of the reference at medium utilization levels. With increasing utilization of the system the runtime converges to the runtime of the reference. Using only the upper bound on the other hand leads to a converse behavior. At medium utilization levels the runtime reduction is rather low, but with increasing utilization the upper bound leads to a significantly reduced runtime. A similar impact of the bounds can be seen for the PJD-model in figure 5c.

Using the lower bound with the event stream model depicted in figure 5d has nearly to no impact on the runtime. The upper bound, however, has again a big impact, leading to a reduction of the runtime to about 5 percent of the reference at very high utilization levels. Using both bounds leads to a slightly improved runtime compared to using only the upper bound.

7. Conclusion

In order to avoid oversized processors during a design space exploration for an embedded system architecture it is necessary to determine the worst-case response time of the tasks exactly when evaluating the real-time constraints. In this paper we have used the holistic real-time analysis introduced in [15] and adapted it for the expressive event stream model of Gresser [6]. By means of this model it is possible to calculate the worst-case response times of the tasks more precisely. In the experiments we have up to 56% better response times versus the PJ-Model from Tindell and Clark and up to 19% better response times versus the PJD-Model from the SymTA/S approach. Due to improved response times resulting from the event stream model slower processors can be used reducing energy consumption and costs.

During a design space exploration we need fast algorithms in order to calculate the response times in a distributed system. For this we have shown how lower and upper bounds for the number of jobs which have to be considered can be used to reduce the runtime of the analysis. For this the improvements of [2], [11] and [13] are combined and extended to the event stream model. The event stream model in conjunction with these bounds is an event model which fulfills the constraints of fast runtime and exact response times very well. There is certainly a trade-off between runtime and accuracy but we believe that only expressive event models in combination with fast algorithms can cope with requirements of industrial companies in the future. A limitation is that adaptive analysis techniques cannot be easily applied on the introduced method.

The next step is to integrate this methodology into a design space exploration. The improved worst-case response times lead presumably to better design solutions and the introduced runtime improvements will cope the computation time complexity.

References

[1] Karsten Albers and Frank Slomka. An event stream driven approximation for the analysis of real- time systems. In *ECRTS '04: Proceedings of the 16th Euromicro Conference on Real-Time Systems*, pages 187–195. IEEE, July 2004.

[2] Enrico Bini and Sanjoy K. Baruah. Efficient computation of response time bounds under fixed-priority scheduling. In *Proceedings of the 15th International Conference on Real-Time and Network Systems*, pages 95–104, March 2007.

[3] Jean-Yves Le Boudec and Patrick Thiran. *Network calculus: a theory of deterministic queuing systems for the internet.* Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[4] Reinder J. Bril, Wim F. J. Verhaegh, and Evert-Jan D. Pol. Initial values for online response time calculations. In *Proceedings of 15th Euromicro Conference on Real-Time Systems*, pages 13–22, 2003.

[5] Robert I. Davis, Attila Zabos, and Alan Burns. Efficient exact schedulability tests for fixed priority real-time systems. *IEEE Trans. Comput.*, 57(9):1261–1276, 2008.

[6] Klaus Gresser. An event model for deadline verification of hard real-time systems. In *Proceedings of the 5th Euromicro Workshop on Real-Time Systems*, pages 118–123, 1993.

[7] Steffen Kollmann, Karsten Albers, and Frank Slomka. Effects of simultaneous stimulation on the event stream densities of fixed-priority systems. In *Spects'08: Proceedings of the International Simulation Multi-Conference*, pages 353–360. IEEE, June 2008.

[8] John P Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *Proceedings of the 11th IEEE Real-Time Systems Symposium*, pages 201–209, December 1990.

[9] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, 1973.

[10] Wan-Chen Lu, Jen-Wei Hsieh, and Wei-Kuan Shih. A precise schedulability test algorithm for scheduling periodic tasks in real-time systems. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 1451–1455, New York, NY, USA, 2006. ACM.

[11] Victor Pollex, Steffen Kollmann, Karsten Albers, and Frank Slomka. Improved worst-case response-time calculations by upper-bound conditions. In *DATE '09: Proceedings of the conference on Design, Automation and Test in Europe*, 2009.

[12] Kai Richter. *Compositional Scheduling Analysis Using Standard Event Models - The SymTA/S Approach.* PhD thesis, University of Braunschweig, 2005.

[13] Mikael Sjödin and Hans Hansson. Improved response-time analysis calculations. In *IEEE Real-Time Systems Symposium*, pages 399–408, 1998.

[14] Steffen Stein, Jonas Diemer, Matthias Ivers, Simon Schliecker, and Rolf Ernst. On the convergence of the symta/s analysis. Technical report, TU Braunschweig, 2008.

[15] Ken Tindell and John Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessing and Microprogramming*, 40:117–134, April 1994.

[16] Ernesto Wandeler. *Modular Performance Analysis and Interface-Based Design for Embedded Real-Time Systems.* PhD thesis, ETH Zurich, September 2006.

[17] Jianjia Wu, Jyh-Charn Liu, and Wei Zhao. On schedulability bounds of static priority schedulers. In *RTAS '05: Proceedings of the 11th IEEE Real Time on Embedded Technology and Applications Symposium*, pages 529–540, Washington, DC, USA, 2005. IEEE Computer Society.