Complex Task Activation Schemes in System Level Performance Analysis

Wolfgang Haid Lothar Thiele

Computer Engineering and Networks Laboratory
Swiss Federal Institute of Technology (ETH) Zurich, Switzerland
{haid, thiele}@tik.ee.ethz.ch

ABSTRACT

The design and analysis of today's complex real-time systems requires advanced methods. Due to ever growing functionality, hardware complexity and component interaction, applying traditional methods like HW/SW co-simulation is getting increasingly difficult. On the other hand, analytic approaches have proven their usefulness and efficiency for system analysis when end-to-end performance figures like delay, throughput and memory consumption are requested. One of the main drawbacks of these methods is the limited set of systems that can be analyzed with high accuracy: Only simple models for task interaction and task semantics can be used. In this paper, we extend existing methods for analyzing heterogeneous multiprocessor systems such that (a) non-preemptive scheduling policies, (b) complex activation schemes for tasks and (c) conditional behavior of task executions can be modeled and analyzed. We demonstrate the usefulness of the proposed approach in a case study.

Categories and Subject Descriptors

C.3 [**Computer Systems Organization**]: Special-Purpose and Application-Based Systems—*Real-Time and Embedded Systems*; C.4 [**Computer Systems Organization**]: Performance of Systems—*Modeling Techniques*

General Terms

Design, Performance, Theory

1. INTRODUCTION

State-of-the-art embedded systems are often implemented on heterogeneous multiprocessor system-on-chip (MpSoC) architectures to provide the performance and flexibility required by today's advanced applications. The complexity of MpSoC systems in terms of hardware, software and HW/SW interaction has turned system level performance analysis into a major challenge.

In classic approaches like HW/SW cosimulation higher complexity translates into increased set-up effort, long run-times, and potentially incomplete coverage of corner cases. These problems can be avoided by using analytic methods. Based on powerful abstractions of application and architecture behavior, these models and methods can be used to obtain hard bounds on properties like end-to-end delays, memory usage and throughput.

One of the drawbacks of analytic approaches for the analysis of distributed embedded systems is their limited modeling capability, leading to estimation results that are often overly pessimistic. For example, it is common to model tasks with single-input single-output components only. As a result, tasks with multiple inputs or outputs which can exhibit complex task behavior (such as conditional function execution, blocking and non-blocking activation patterns) cannot be modeled in sufficient detail and the overall system analysis may suffer from reduced accuracy.

In this paper, we consider essential extensions to existing modular analysis frameworks for the performance analysis of distributed embedded systems, such as non-preemptive resource sharing and a generic task activation scheme that contains as special cases the well known OR/AND-activation.

Non-preemptive scheduling. While in the domain of general-purpose computing non-preemptive scheduling plays a minor role, it can be often found in embedded systems. Examples of such systems are network routers, systems with low-latency paths for user or control feedback, and systems using non-preemptive bus communication.

Generalized activation scheme. While OR/AND-activation of tasks and non-preemptive scheduling can occur in any system, they are often used implicitly when programming single tasks of an application. Frequently, even more general activation schemes are used in the task implementation. Depending on the availability of data, a task executes different code segments (conditional behavior) and a mixture of blocking and non-blocking queue accesses is used. Algorithm 1 shows an example of such a task. Depending on the availability of events on one or several inputs, one code segment is executed mutually exclusive of any other code segment.

In this work, we use the analytic framework of modular performance analysis—real-time calculus (MPA-RTC) [1] for modeling and performance analysis. The contributions presented in this paper can be summarized as follows:

Algorithm 1 Example of a complex task activation scheme: The task has five inputs, namely *reconfig* (carries reconfiguration data) and *dataA* – *dataD*. The *test* function returns *true* if data are available on the specified stream, otherwise *false*. The task is assumed to be repeatedly called by the global scheduler.

```
1: if test(reconfig) then
2:     execute code that reconfigures the task;
3: else if test(dataA) or test(dataB) then
4:     process first event arrived at dataA or dataB;
5: else if test(dataC) and test(dataD) then
6:     process first event in dataC and in dataD;
7: end if
```

- We present an analytic method to describe the execution of a task that is triggered by events on multiple input event streams using OR-activation, AND-activation, or a combination thereof.
- We present an analytic method to describe the execution of a set of tasks on a resource with non-preemptive scheduling.
- By combining the above concepts in a hierarchical manner, general activation schemes of tasks can be modeled such as conditional behavior, complex activation patterns and blocking/non-blocking queue access.
- We present a case study in which we obtain an accurate characterization of a system based on the derived expressions.

By this extension, the scope of currently available analytic performance analysis methods for embedded systems is greatly extended, leading to (a) a larger class of systems that can faithfully be modeled and (b) a higher accuracy of analysis in terms of bounds that are closer to the actually observable worst case.

1.1 Related Work

Realistic systems are often implemented using tasks that exhibit complex activation schemes. To obtain accurate results, it is crucial that these schemes are taken into account. In system simulation or trace-based methods, this is usually not a problem because the activation conditions are hard-coded in the implementation. In other methods, however, complex activations are often not considered in the first place for ease of analysis. Nevertheless, since modeling of activations schemes is essential for analyzing many applications, several methods have been proposed to deal with them. In MAST (Modeling and Analysis Suite for Real Time Applications), a framework for holistic scheduling analysis, complex activation patterns can be modeled using *event handlers*, such as *barriers* (comparable to AND-activation) and *concentrators* (comparable to OR-activation) [3]. In SymTA/S (Symbolic Timing Analysis for Systems) [4], complex activation patterns can be modeled using tasks with OR/AND-activation semantics [5]. In MPA-RTC, state machines have been proposed to handle OR/AND-activation of tasks [8]. Compared to the method proposed in this contribution, however, that method is computationally more expensive. In all cases, there is no possibility to model more complex conditional task behavior and mixtures of blocking/non-blocking queue access schemes which can be found in typical embedded system applications.

The analysis of a component using a non-preemptive fixed priority (FP) scheduling policy is related to the analysis of non-preemptive priority multiplexers in network queuing theory. In [2], delay and backlog bounds for a non-preemptive FP multiplexer with constant output rate are derived. The obtained bounds are a special case of the results presented in this contribution. Non-preemptive FP queuing of two streams has been considered in [6] and of multiple streams in [7]. In contrast to our analysis, the mentioned methods neither provide lower bounds on the amount of events at the output of a task nor upper bounds on the amount of the remaining service at the processing resource where the task is executed. Therefore, they are not amenable to be part of a modular performance analysis method that allows to faithfully model hierarchical and distributed resource sharing scenarios.

2. BASIC CONCEPTS AND NOTATION

We use MPA-RTC as the basis for our analysis, see [1]. Therefore, we shortly review the corresponding basic concepts and introduce our notation in this section.

2.1 Arrival and Service Curves

The timing properties of event streams will be described by arrival curves that are defined as follows:

Definition 2.1 (Arrival Curve). *Let $R[s,t)$ denote the number of events that arrive on an event stream in the time interval $[s,t)$. Then, the corresponding upper and lower arrival curves are denoted as α^u and α^l, respectively, and satisfy:*

$$\alpha^l(t-s) \le R[s,t) \le \alpha^u(t-s), \quad \forall s < t, \tag{1}$$

where $\alpha^u(0) = \alpha^l(0) = 0$.

In this model, the timing information of standard event stream models like *periodic*, *periodic with jitter*, *periodic with bursts* or *sporadic* can be represented by an appropriate choice of α^u and α^l [1]. Moreover, it is also possible to determine the values of α^u and α^l corresponding to any given finite length event trace from calculations, (formal) specification or simulation.

In a similar way, the properties of resources are described using the concept of service curves.

Definition 2.2 (Service Curve). *Let $C[s,t)$ denote the number of events that a resource can process in the time interval $[s,t)$. Then, the corresponding upper and lower service curves are denoted as β^u and β^l, respectively, and satisfy:*

$$\beta^l(t-s) \le C[s,t) \le \beta^u(t-s), \quad \forall s < t, \tag{2}$$

where $\beta^u(0) = \beta^l(0) = 0$.

Event streams and resource capabilities are described in terms of "event units" and not in terms of "resource units", such as the number of cycles or number of bytes. The resource-based service curve $\overline{\beta}(\Delta)$ denotes the available resource units available in any time interval of length Δ. In a similar way, the resource-based arrival curve $\overline{\alpha}(\Delta)$ denotes the requests in terms of resource units that arrive in any time interval of length Δ. The relation between the event-based and resource-based curves is given by workload transformations defined as follows:

Definition 2.3 (Workload Transformation). *The functions $\mathcal{W}^u_\beta(r)$ and $\mathcal{W}^l_\beta(r)$ denote the maximum and minimum number of successive events that can be processed with r resource units. The functions $\mathcal{W}^u_\alpha(e)$ and $\mathcal{W}^l_\alpha(e)$ denote the maximum and minimum number of resource units that are necessary to process e successive events.*

Using this definition, we can derive the following relations that connect resource-based and event-based arrival and service curves:

$$\beta^u = \mathcal{W}^u_\beta\left(\overline{\beta}^u\right) \quad \beta^l = \mathcal{W}^l_\beta\left(\overline{\beta}^l\right)$$
$$\overline{\alpha}^u = \mathcal{W}^u_\alpha\left(\alpha^u\right) \quad \overline{\alpha}^l = \mathcal{W}^l_\alpha\left(\alpha^l\right)$$

In the case of fixed worst-case and best-case resource units for each event (e.g. cycles, execution times), we can derive simple expressions for the workload transformations. If C^{max} and C^{min} denote the maximum and minimum amount of resource units for the processing of one event, we find:

$$\mathcal{W}^u_\beta(r) = r/C^{min} \qquad \mathcal{W}^l_\beta(r) = r/C^{max}$$
$$\mathcal{W}^u_\alpha(e) = e \cdot C^{max} \qquad \mathcal{W}^l_\alpha(e) = e \cdot C^{min}$$

In order to describe the operations on arrival and service curves we define the following operators:

Definition 2.4 (Min/Max-Plus Operators). *For two functions f and g, the min-plus convolution $(f \otimes g)$, the min-plus deconvolution $(f \oslash g)$, the max-plus convolution $(f \overline{\otimes} g)$, and the max-plus deconvolution $(f \overline{\oslash} g)$ are defined as follows [6]:*

$$(f \otimes g)(\Delta) = \inf_{0 \le \lambda \le \Delta} \{f(\Delta - \lambda) + g(\lambda)\}$$
$$(f \oslash g)(\Delta) = \sup_{\lambda \ge 0} \{f(\Delta + \lambda) - g(\lambda)\}$$
$$(f \overline{\otimes} g)(\Delta) = \sup_{0 \le \lambda \le \Delta} \{f(\Delta - \lambda) + g(\lambda)\}$$
$$(f \overline{\oslash} g)(\Delta) = \inf_{\lambda \ge 0} \{f(\Delta + \lambda) - g(\lambda)\}$$

2.2 Greedy Processing Component

In our analysis, we use an abstract component referred to as greedy processing component (GPC) to model the execution of tasks on a single resource. The GPC is a component that is triggered whenever an event is available on the input event stream (described by the arrival curve α) and produces a single output event stream (described by the arrival curve α'). At every event arrival, a task is instantiated to process the incoming event. Events are processed in a greedy fashion in first-in first-out order, while being restricted by the availability of processing resources described by the service curve β. The GPC can be modeled using the following relations where β' denotes the remaining service available to process other event streams [1]:

$$\alpha'^u = \min\{(\alpha^u \otimes \beta^u) \oslash \beta^l, \beta^u\} \tag{3}$$
$$\alpha'^l = \min\{(\alpha^l \oslash \beta^u) \otimes \beta^l, \beta^l\} \tag{4}$$
$$\beta'^u = (\beta^u - \alpha^l) \,\overline{\oslash}\, 0 \tag{5}$$
$$\beta'^l = (\beta^l - \alpha^u) \,\overline{\otimes}\, 0 \tag{6}$$

Upper bounds on the number of events in the input queue (backlog) $b(t)$ and the event delay $d(t)$ can be determined as follows:

$$b(t) \le B(\alpha^u,\ \beta^l) \qquad d(t) \le D(\alpha^u,\ \beta^l) \tag{7}$$

with

$$\begin{aligned} B(\alpha^u, \beta^l) &= \sup_{\lambda \ge 0} \left\{\alpha^u(\lambda) - \beta^l(\lambda)\right\} \\ D(\alpha^u, \beta^l) &= \sup_{\Delta \ge 0} \left\{\inf\{\tau \ge 0 : \alpha^u(\Delta) \le \beta^l(\Delta + \tau)\}\right\} \end{aligned} \tag{8}$$

To be able to apply these relations, the curves must be expressed in the same units, that is, either in the number of events or in the amount of resource units. This can be achieved by the workload transformations introduced above.

3. COMPLEX ACTIVATION SCHEMES

In the following, we will develop the analysis for tasks with complex activation schemes step by step. After analyzing the simpler case of OR- and AND-activation, we will model and analyze non-preemptive FP scheduling. Based on these results that are important on their own (see section 1), we will use a hierarchical and component-based approach to model complex task structures with conditional behavior and blocking/non-blocking queue access.

3.1 Components with Multiple Inputs

In realistic systems, the flow of data between components is not limited to one-to-one connections. Rather, the activation of one component often depends on events arriving on input streams from multiple components.

First results to model such components have been obtained for the periodic with jitter event model by Jersak et al. [5]. Jersak et al. need to represent the patterns emerging from the boolean merging of multiple streams by a standard event stream model which leads, in general, to too pessimistic bounds. In contrast, in the approach proposed in this paper, arbitrary event models can be handled.

In the following we adopt a compositional approach, see Fig. 1, which separates the activation scheme f, i.e. OR- or AND-activation, from the greedy processing of events.

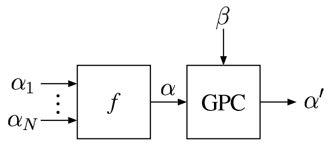

Figure 1: Principle of analyzing components with multiple inputs.

3.2 Abstract OR Component

The two-input abstract OR component produces an event on its output whenever an event is available on either of the two input streams.

Theorem 3.1 (Abstract OR Component). *Assume an abstract OR component with two input event streams with arrival curves $\left[\alpha^u_1, \alpha^l_1\right]$ and $\left[\alpha^u_2, \alpha^l_2\right]$. Then, the outgoing arrival curves $\left[\alpha^u_{OR}, \alpha^l_{OR}\right]$ are given by:*

$$\alpha^u_{OR} = \alpha^u_1 + \alpha^u_2 \qquad \alpha^l_{OR} = \alpha^l_1 + \alpha^l_2 \tag{9}$$

Moreover, the event streams observe no delays and backlogs.

PROOF. Only a sketch is provided here. We note that OR-semantics implies that we just have to add the number of events on all input event streams to obtain the number of events on the output event stream of the abstract OR component. Using (1), the desired result is obtained. □

Let us next determine the maximum delay and backlog for the OR-activated greedy processing element GPC, see Fig. 1 with $f = OR$. As there is no delay and backlog associated to the abstract OR component, we can simply determine a bound on the delay d_i between the input stream i and the output α' and on the backlog b as follows:

$$d_i(t) \leq D\left(\sum_i \alpha_i^u,\ \beta^l\right) \quad b(t) \leq B\left(\sum_i \alpha_i^u,\ \beta^l\right)$$

3.3 Abstract AND Component

The two-input abstract AND component produces an event on its output whenever at least one event is available on both input event streams. For each output event, one event is removed from both input streams.

THEOREM 3.2 (ABSTRACT AND COMPONENT). *Assume an abstract AND component with two input event streams with arrival curves $[\alpha_1^u, \alpha_1^l]$ and $[\alpha_2^u, \alpha_2^l]$. Let B_1^0 and B_2^0 denote the initial backlogs of the two streams, with $B_1^0 \cdot B_2^0 = 0$. Then, the outgoing arrival curves $[\alpha_{AND}^u, \alpha_{AND}^l]$ are given by:*

$$\begin{aligned} \alpha_{AND}^u = \max\Big\{&\min\{\alpha_1^u \oslash \alpha_2^l + B_1^0 - B_2^0,\ \alpha_2^u\},\\ &\min\{\alpha_2^u \oslash \alpha_1^l + B_2^0 - B_1^0,\ \alpha_1^u\}\Big\}\\ \alpha_{AND}^l = \min\Big\{&\max\{\alpha_1^l \,\overline{\oslash}\, \alpha_2^u + B_1^0 - B_2^0,\ \alpha_2^l\},\\ &\max\{\alpha_2^l \,\overline{\oslash}\, \alpha_1^u + B_2^0 - B_1^0,\ \alpha_1^l\}\Big\} \end{aligned} \tag{10}$$

Using (8), the delays d_i and backlogs b_i for the two event streams satisfy:

$$\begin{aligned} d_1(t) &\leq D(\alpha_1^u + B_1^0,\ \alpha_2^l + B_2^0)\\ d_2(t) &\leq D(\alpha_2^u + B_2^0,\ \alpha_1^l + B_1^0)\\ b_1(t) &\leq \max\Big\{B(a_1^u + B_1^0,\ \alpha_2^l + B_2^0),\ 0\Big\}\\ b_2(t) &\leq \max\Big\{B(a_2^u + B_2^0,\ \alpha_1^l + B_1^0),\ 0\Big\} \end{aligned} \tag{11}$$

PROOF. We only prove the upper bound α_{AND}^u. α_{AND}^l can be proven analogously. The cumulative flow $R_{\text{AND}}(\tau)$ after the AND component is given by:

$$R_{\text{AND}}(\tau) = \min\left\{R_1(\tau) + B_1^0,\ R_2(\tau) + B_2^0\right\} \tag{12}$$

Thus, for all $s \leq t$,

$$\begin{aligned} R_{\text{AND}}(t) - R_{\text{AND}}(s) = &\min\left\{R_1(t) + B_1^0,\ R_2(t) + B_2^0\right\}\\ &- \min\left\{R_1(s) + B_1^0,\ R_2(s) + B_2^0\right\} \end{aligned} \tag{13}$$

Observe that for $a,\ b,\ c,\ d\ \in\ \mathbb{R}$ the following equation applies:

$$\begin{aligned} \min\{a,\ b\} - \min\{c,\ d\} = \max\{&\min\{a - c,\ b - c\},\\ &\min\{a - d,\ b - d\}\} \end{aligned} \tag{14}$$

We can now compute an upper bound for (13) using (14) and the following two inequalities: $R_1(t) - R_2(s) \leq \left(\alpha_1^u\ \oslash\ \alpha_2^l\right)(t - s)$ and $(\alpha_1 \oslash \alpha_2)(\Delta) \leq \alpha_1(\Delta)$.

$$\begin{aligned} &R_{\text{AND}}(t) - R_{\text{AND}}(s) =\\ &= \max\big\{\min\big\{R_1(t) + B_1^0 - \big(R_1(s) + B_1^0\big),\\ &\qquad\qquad R_2(t) + B_2^0 - \big(R_1(s) + B_1^0\big)\big\},\\ &\qquad \min\big\{R_1(t) + B_1^0 - \big(R_2(s) + B_2^0\big),\\ &\qquad\qquad R_2(t) + B_2^0 - \big(R_2(s) + B_2^0\big)\big\}\big\}\\ &\leq \max\Big\{\min\Big\{\alpha_1^u,\ \Big(\alpha_2^u\ \oslash\ \alpha_1^l\Big)(t - s) + B_2^0 - B_1^0\Big\},\\ &\qquad \min\Big\{\Big(\alpha_1^u\ \oslash\ \alpha_2^l\Big)(t - s) + B_1^0 - B_2^0,\ \alpha_2^u\Big\}\Big\} \end{aligned}$$

The lower arrival curve α_2^l can be interpreted as a lower service curve concerning the processing of events on R_1. Therefore, we can simply apply the delay and backlog bounds (7) for the GPC, taking into consideration the initial backlogs. □

We can now determine a bound on the maximum delay and backlog for an AND-activated GPC, see Fig. 1 with $f = AND$. The simplest possibility is to just add the quantities for the AND and the GPC by using (7), (10) and (11):

$$\begin{aligned} d_1(t) &\leq D(\alpha_{AND}^u,\ \beta^l) + D(\alpha_1^u + B_1^0,\ \alpha_2^l + B_2^0)\\ b_1(t) &\leq B(\alpha_{AND}^u,\ \beta^l) + B(a_1^u + B_1^0,\ \alpha_2^l + B_2^0) \end{aligned}$$

The relations for d_2 and b_2 can be obtained similarly.

3.4 Non-Preemptive FP Scheduling

In order to model complex task activation schemes and perform the corresponding performance analysis, we need to handle non-preemptive fixed priority (FP) scheduling. The following theorem describes the transfer function of a GPC with non-preemptive FP scheduling.

THEOREM 3.3 (NON-PREEMPTIVE FP SCHEDULING). *Assume a set of N single-input tasks ordered according to their priorities (N is highest and 1 is lowest). Then, non-preemptive FP scheduling can be modeled using (3) – (6) by replacing the event-based arrival and service curves α, α' and β' by their resource-based counterparts $\overline{\alpha}$, $\overline{\alpha}'$ and $\overline{\beta}'$ respectively. In addition, the event-based service curve β needs to be replaced by $\overline{\gamma}$ which is related to the initial resource-based service curve $\overline{\beta}$ as follows:*

$$\begin{aligned} \overline{\gamma}_i^u(\Delta) = \min\{&\ \overline{\beta}^u(\Delta),\\ &\inf_{\lambda \geq 0}\{\overline{\beta}^u(\Delta + \lambda) - \sum_{j=i+1}^{N} \overline{\alpha}_j^l(\Delta + \lambda)\} + C_i^{max}\} \end{aligned} \tag{15}$$

$$\overline{\gamma}_i^l(\Delta) = \max\{\ 0, \tag{16}$$

$$\sup_{0 \leq \lambda \leq \Delta}\{\overline{\beta}^l(\Delta - \lambda) - \sum_{j=i+1}^{N} \overline{\alpha}_j^u(\Delta - \lambda)\} - \max_{1 \leq j < i}\{C_j^{max}\}\}$$

PROOF. We sketch the prove for the lower service curve (16) only. The proof for the upper service curve (15) follows the same line of arguments. First, let us consider the conventional preemptive FP scheme, see (3) – (6). In this case, the remaining resource service $\overline{\beta}_i'^l$ after processing task i is available to the processing component for task $i - 1$, i.e. $\overline{\beta}_{i-1}^l = \overline{\beta}_i'^l$. Thus, the resulting input service curve for task i is:

$$\overline{\beta}_i^l(\Delta) = \sup_{0\le\lambda\le\Delta}\left\{\overline{\beta}^l(\Delta-\lambda) - \sum_{j=i+1}^{N}\overline{\alpha}_j^u(\Delta-\lambda)\right\}$$

In case of non-preemptive scheduling, task i can be prevented from accessing the resource by time at most $\max_{1\le j<i}\{C_j^{max}\}$ because of a single not yet finished event from lower priority tasks. This is equivalent to having an additional single high priority task with $\alpha(0) = 0$ and $\alpha(\Delta) = \max_{1\le j<i}\{C_j^{max}\}$ for all $\Delta > 0$. This leads to expression (16). $\Box$

3.5 Mixed Activation Schemes

By combining the previous results, i.e. the OR/AND-activation and non-preemptive FP scheduling, we can faithfully model a large class of activation schemes of processing components. The basis for this results is the observation that common structures such as in Algorithm 1 can be modeled by a combination of the previous schemes. This underlines the advantages of a compositional analysis methodology. Instead of providing a generic transformation from a class of algorithms to the associated analysis structure, we will describe an artificial example in Algorithm 2 that demonstrates almost all techniques that can be applied. We suppose that executing code c_i requires at least C_i^{min} and at most C_i^{max} resource units and the function test(*input*) returns *true* if there is an event in the queue associated to *input*.

Algorithm 2 Example of a complex task activation scheme.

```
 1: if test(inputA) then
 2:     remove one event from inputA;
 3:     execute code $c_1$;
 4:     send one event to outputA;
 5: else if test(inputB) or test(inputC) then
 6:     remove first event arrived at inputB or inputC;
 7:     execute code $c_2$;
 8:     send one event to outputA;
 9: else if test(inputD) and test(inputE) then
10:     remove one event from inputD and inputE;
11:     execute code $c_3$;
12:     send one event to outputB;
13: end if
```

Fig. 2 represents the building blocks modeling the above activation scheme. Note that the depicted analysis component with input arrival curves $\alpha_A, \ldots, \alpha_E$ and input service $\overline{\beta}$ can be used in a global system analysis which involves other tasks (with other activation schemes) and other (hierarchical) resource sharing methods, see e.g. [1]. In Fig. 2, there are the analysis components AND and OR (see Fig. 1 and the associated equations), the GPC, a component representing equations (15, 16), and a component to determine the remaining service, see also (5, 6):

$$\overline{\beta'^l}(\Delta) = \sup_{0\le\lambda\le\Delta}\left\{\overline{\beta^l}(\Delta-\lambda) - \sum_{(j)}\overline{\alpha_j^u}(\Delta-\lambda)\right\}$$
$$\overline{\beta'^u}(\Delta) = \inf_{\lambda\ge 0}\left\{\overline{\beta^u}(\Delta+\lambda) - \sum_{(j)}\overline{\alpha_j^l}(\Delta+\lambda)\right\} \quad (17)$$

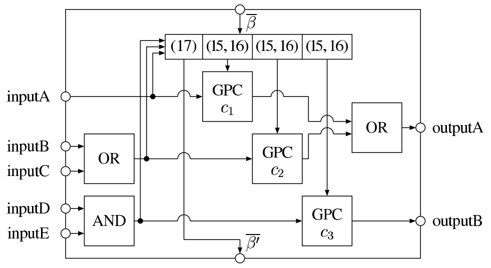

Figure 2: Analysis model for Algorithm 2.

4. MPEG-2 DECODER CASE STUDY

As a case study, an MPEG-2 video/audio decoder is modeled and analyzed in this section. Fig. 3 shows the mapping of the MPEG-2 decoder onto an MpSoC hardware platform.

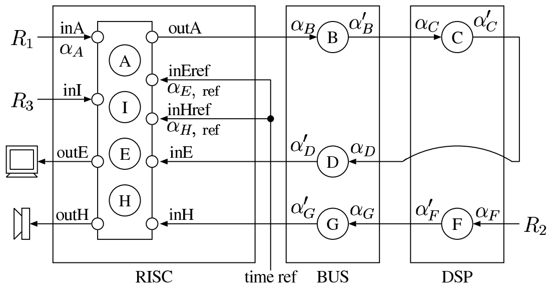

Figure 3: Analysis model of the MPEG-2 decoder mapped onto an MpSoC platform.

As can be seen in Fig. 3, the MPEG-2 decoder is partitioned into several tasks where the tasks A, E, H, I are subtasks of a single decoding task, see Algorithm 3. A short description of the (sub)tasks and their best and worst case execution times are listed in Table 1. The hardware platform is periodically clocked and consists of a RISC processor (750 *MHz*) and a DSP (250 *MHz*) that are interconnected by a single bus (125 *MHz*). The scheduler on the DSP and the bus arbiter use a non-preemptive FP scheduling policy. In Table 1, the according task priorities are shown.

Algorithm 3 Decoding task on RISC.

```
 1: if test(inH) and test(inHref) then
 2:     remove one event from inH and inHref;
 3:     execute code H;
 4:     send one event to outH;
 5: else if test(inA) then
 6:     remove one event from inA;
 7:     execute code A;
 8:     send one event to outA;
 9: else if test(inE) and test(inEref) then
10:     remove one event from inE and inEref;
11:     execute code E;
12:     send one event to outE;
13: else if test(inI) then
14:     remove one event from inI;
15:     execute code I;
16: end if
```

The input arrival curves for the video and the audio stream are shown in Fig. 4. For the video stream, we assume

stream	task	function	resource demand in 10^3 cycles	priority
video	A	VLD, IQ, IS	[5, 10]	—
	B	data transfer	[1, 1]	2
	C	IDCT, MC	[3, 4]	1
	D	data transfer	[1, 1]	1
	E	assemble video-frames	[0.1, 0.1]	—
audio	F	DEC, IMDCT, SYN	[500, 1000]	2
	G	data transfer	[100, 100]	3
	H	assemble audio-frames	[10, 10]	—
—	I	playback control	[2, 2]	—

Table 1: Tasks and their resource demand. The resource demand is specified in number of cycles per macroblock for the video stream and in number of cycles per frame for the audio stream.

a PAL-resolution (720 × 576 pixels) and a frame refresh rate of 25 frames per second, yielding a macroblock rate of 40.500 *macroblocks/s*. (A macroblock consists of 16 × 16 pixels.) Concerning the audio stream, the frame rate is 38.3 *frames/s* since the duration of a frame is known to be 26.1 *ms*. The two rates define the slope of the arrival curves in the long term. During short time intervals, however, the input rates can vary due to load variations on a second bus that is used to deliver the streams to the RISC and the DSP (for simplicity, not explicitly modeled here).

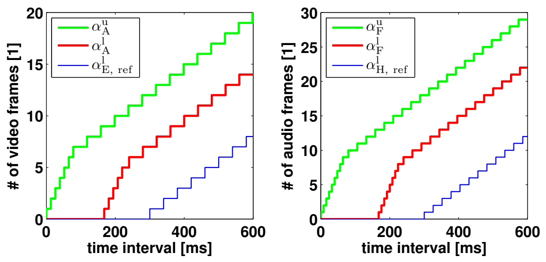

Figure 4: Input arrival curves of the video stream (left) and audio stream (right).

Fig. 4 also shows the time references for the AND-activated subtasks E and H, represented as lower arrival curves. For correct functioning of the system, the lower output arrival curves of the tasks D and G must be greater than these references for any time interval. This ensures that after an initial pause of 300 *ms* the video/audio stream can be played without interruptions.

The methodology and the required relations to analyze this system have been shown in the previous sections, so we only report the results here. The plot in Fig. 5 shows the obtained output arrival curves α'_D of task D. As can be seen, the required bound is just met. The result for the audio stream (task G) is also within the required bound, but less critical (not shown). We have thereby proven that the system is compliant with the specified goal and will correctly play back any video/audio stream that conforms to the specified arrival curve.
Additionally, the right plot in Fig. 5 shows the remaining service of the bus that would be available to other tasks on the same resource, represented by the service curves $\overline{\beta}'_{\mathrm{BUS}}$.

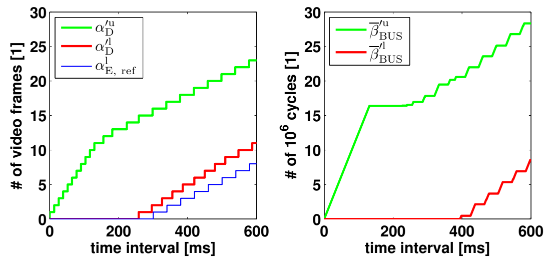

Figure 5: Output arrival curves of task D (left) and remaining service of the bus, represented as lower service curves (right).

5. CONCLUSION

We considered system level performance analysis of systems with tasks that exhibit complex activation patterns and conditional execution of sub-tasks. In particular, we focused on the analytic framework of MPA-RTC and proposed an approach based on OR/AND-activation and non-preemptive fixed priority scheduling to analyze such systems. We presented the according components and proved the required relations, thereby extending the modeling capabilities of MPA-RTC. The proposed techniques do not only allow modeling a broader class of systems but can be used to improve the accuracy of system level analysis using MPA-RTC, in general. Finally, we applied the proposed techniques to analyze an MPEG-2 decoder mapped onto an MpSoC platform.

6. REFERENCES

[1] S. Chakraborty, S. Künzli, and L. Thiele. A General Framework for Analyzing System Properties in Platform-Based Embedded System Design. In *Proc. 6th Design, Automation and Test in Europe (DATE)*, pages 190–195, Munich, Germany, Mar. 2003.

[2] R. L. Cruz. A Calculus for Network Delay, Part I: Network Elements in Isolation. *IEEE Trans. Inform. Theory*, 37(1):114–131, Jan. 1991.

[3] M. González Harbour, J. J. Gutiérrez García, J. C. Palencia Gutiérrez, and J. M. Drake Moyano. MAST: Modeling and Analysis Suite for Real Time Applications. In *Proc. 13th Euromicro Conference on Real-Time Systems*, pages 125–134, Delft, The Netherlands, June 2001.

[4] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst. System Level Performance Analysis — The SymTA/S Approach. *IEE Proceedings Computers and Digital Techniques*, 152(2):148–166, Mar. 2005.

[5] M. Jersak, K. Richter, and R. Ernst. Performance Analysis for Complex Embedded Systems. *Int'l Journal of Embedded Systems*, 1(1–2):33–49, 2005.

[6] J.-Y. Le Boudec and P. Thiran. *Network Calculus — A Theory of Deterministic Queuing Systems for the Internet*, volume 2050 of *Lecture Notes in Computer Science*. Springer Verlag, 2001.

[7] J. Schmitt. On Average and Worst Case Behaviour in Non-Preemptive Priority Queuing. In *Proc. 2003 Int'l Symp. on Performance Evaluation of Computer and Telecommunication Systems*, pages 197–204, 2003.

[8] E. Wandeler and L. Thiele. Abstracting Functionality for Modular Performance Analysis of Hard Real-Time Systems. In *Proc. 2005 Conference on Asia South Pacific Design Automation (ASP-DAC'05)*, pages 697–702, Shanghai, China, Jan. 2005.