Two Parameter Workload Characterization for Improved Dataflow Analysis Accuracy

Joost P.H.M. Hausmans[§] joost.hausmans@utwente.nl
Stefan J. Geuns[§] stefan.geuns@utwente.nl
Maarten H. Wiggers[*] [‡] maarten.wiggers@us.fujitsu.com
Marco J.G. Bekooij[§] [¶] marco.bekooij@nxp.com

[§] University of Twente, Enschede, The Netherlands
[‡] Fujitsu Laboratories of America, Sunnyvale, CA, USA
[¶] NXP Semiconductors, Eindhoven, The Netherlands

***Abstract*—Real-time stream processing applications, such as radios, can often be modeled intuitively with dataflow models. Given the Worst-Case Execution Times (WCETs) of the tasks, which characterizes workload with one parameter, dataflow analysis techniques have been used to compute the minimum throughput and maximum latency of these applications. However, a large difference between the WCETs of the tasks and their average execution times can result in a large difference between the computed worst-case throughput and the actual obtained throughput.**

To reduce the difference between the worst-case throughput, determined by analysis, and the actual obtained throughput, we introduce in this paper a two parameter (σ, ρ) workload characterization of the tasks to improve the accuracy of dataflow analysis. The (σ, ρ) workload characterization captures information on the maximum cumulative execution time of consecutive executions of a task and can therefore be seen as a generalization of the WCET characterization.

We show how the (σ, ρ) workload characterization can be used in combination with several types of dataflow graphs and how it can be used to improve the temporal analysis results of real-time stream processing applications. We illustrate this for a DVB-T radio application, a car-radio application and a data-dependent MP3 playback application.

I. Introduction

Real-time stream processing applications, such as radios, often have strict throughput requirements and relaxed latency requirements. These applications can be intuitively modeled with dataflow models [1], [2] and several analysis methods exist to verify whether the real-time requirements will be met [3], [4]. The modeling power is not limited to applications with static behavior as they also allow the modeling of the behavior of data dependent applications [5].

One important metric of worst-case temporal analysis methods is their accuracy. The accuracy of traditional analysis based on Synchronous Dataflow (SDF) graphs is limited because it is based on a workload characterization with one parameter, the Worst-Case Execution Time (WCET) of the task [2]. Such a simple workload characterization can result in a large difference between the computed minimum throughput and the actual throughput if there is large difference between the WCETs and the average execution time of the tasks. The same holds for tasks modeled as a Cyclo-Static Dataflow (CSDF) [6], [7] actor which uses only one parameter per phase. Therefore, until now a constant workload is considered for SDF graphs and a periodic workload is considered for CSDF graphs.

The difference between the computed worst-case and the actual throughput can often be reduced by making use of knowledge about the maximum sum of execution times for a number of consecutive executions. The fact that the worst-case cumulative execution time of n subsequent executions of a task is often smaller than n times the WCET of this task can be exploited. Knowledge of the worst-case cumulative execution time can be captured in so-called workload functions [8]. The use of aperiodic workload functions has not been considered in combination with dataflow analysis methods.

The contribution of this paper is a technique to represent workload functions characterized with two parameters, σ and ρ, in a dataflow model. This dataflow model can be analyzed with standard dataflow analysis techniques. The analysis of this dataflow model will result in a tighter minimum throughput bound and can be used in combination with run-time scheduling.

The presented approach can be used in combination with functional deterministic dataflow models for which a minimum throughput and maximum latency can be computed at design time. Examples of these models are the Homogeneous Synchronous Dataflow (HSDF), SDF [2], [9], CSDF and Variable-Rate Dataflow (VRDF) [5] models.

The case-study shows how the (σ, ρ) workload characterization can be used. We will show that tighter analysis results can be obtained for a DVB-T radio application when using the (σ, ρ) workload characterization instead of the traditional WCET workload characterization. Next to that we illustrate the effect of a latency constraint on the temporal results that can be obtained using a (σ, ρ) workload characterization with a car-radio application. We also combine the (σ, ρ) characterization with a data-dependent MP3 playback application which is modeled using a VRDF graph.

The outline of this paper is as follows. We discuss related work in Section II. In Section III we describe the basic idea behind the (σ, ρ) based analysis method and in Section IV we present the improved dataflow model and prove its correctness. Section V combines (σ, ρ) information with the modeling of cyclic temporal behavior. An evaluation of the (σ, ρ) workload characterization in combination with dataflow analysis techniques is presented in Section VI and we conclude this paper in Section VII.

[*]This work was done while the author was at the University of Twente, Enschede, The Netherlands.

This work is supported in part by the Dutch Technology Foundation STW, project NEST 10346 and the Sensor Technology Applied in Reconfigurable systems for sustainable Security (STARS) project.

II. Related Work

While [10] introduced a modeling relation between dataflow components and tasks to analyze the effects of run-time scheduling, state-of-the-art in modeling the task workload function is to have a one-to-one correspondence between dataflow firings and task executions. Each firing has a duration equal to an upper bound on the corresponding task execution time [10]–[13]. Our method does not have this restrictive one-to-one relation between firings and executions, instead we introduce a method that uses the modeling relation between dataflow components and tasks to capture the tasks workload functions. In [14] similar dataflow components as we use in this paper are used but the model from [14] also only allows to use one single (worst-case) execution time per task.

A method that uses knowledge about the execution time of consecutive executions and which aims at improving the accuracy of the temporal analysis results is presented in [8]. It uses workload functions to express changes in execution times of different modes of an application. However, these workload functions cannot be used directly in combination with dataflow models because they have an arbitrary infinite structure which cannot be described by a finite number of parameters. Dataflow models currently use only one parameter, the firing duration, and can therefore not include the effect of an arbitrary workload function. We present a workload characterization which can be combined with existing dataflow models and which can be described by two parameters. This workload characterization can be seen as an upper bound on a workload function. Methods which use an arbitrary workload function [8], [15] use iterative fixed-point computation which result in an high worst-case computation complexity.

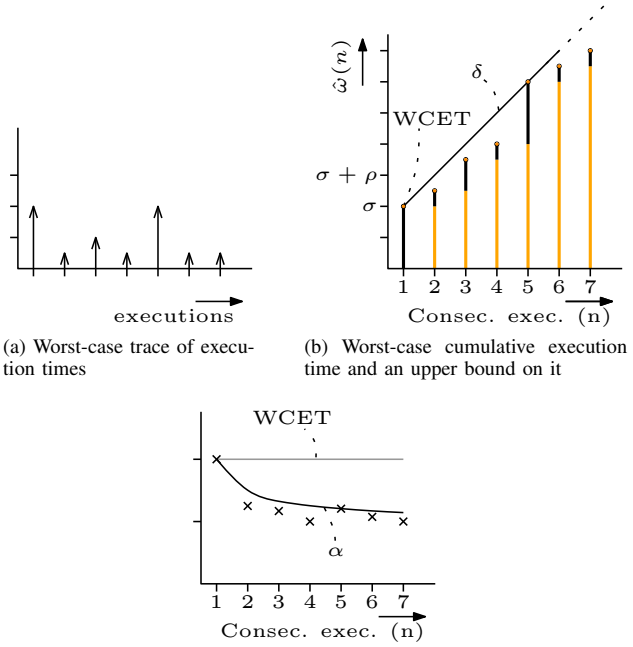
Workload functions can also be used to express cyclic temporal behavior. However, the methods of [8] and [15] have no mechanism to express this cyclic behavior. This loss of information can lead to an overestimation. Consider a task with cyclic temporal behavior that always starts with three executions which take maximally 1 time unit, followed by an execution that takes maximally 4 time units. The methods of [8] and [15] cannot distinguish between this sequence and, for example, its reverse; i.e., starting with the longest execution. Our method will behave similarly, when we model this task with actors without phases; e.g., HSDF or SDF. However, as explained further in Section V, we can model this cyclo-static sequence of execution times explicitly by modeling this task with for example a CSDF actor, resulting in a more accurate model.

Methods based on probabilistic execution times have been developed [16]–[18]. These methods are unsuitable for the derivation of the worst-case throughput and typically have a high computational complexity.

The method presented in this paper is independent of the number of tokens consumed and produced per execution. This means that our method can be combined with tasks that have data-dependent behavior [5] which is not considered for [8] and [15].

III. Basic Idea

This section contains the basic idea behind the (σ, ρ) workload characterization and how it can be applied. We first informally introduce the (σ, ρ) workload characterization. We then show the basic idea behind the conservative temporal analysis methods based on dataflow models and we conclude this section with an example that illustrates the benefits of using a (σ, ρ) workload characterization in combination with a dataflow model.



(a) Worst-case trace of execution times

(b) Worst-case cumulative execution time and an upper bound on it

(c) average worst-case execution times compared to the WCET

Figure 1. Worst case execution time information of consecutive executions

A. Sigma, Rho Workload Characterization

The (σ, ρ) workload characterization forms an upper bound on the worst-case cumulative execution time of a task. It bounds this worst-case cumulative execution time with a function which can be described by two parameters, σ and ρ. These two parameters can be used as the execution time description of a task for which we can provide worst-case temporal analysis results.

Figure 1 illustrates the idea of the (σ, ρ) workload characterization. Figure 1a shows the worst-case trace of execution times of seven executions of a task. Figure 1b contains the worst-case cumulative execution time $(\hat{\omega}(n))$ for up to seven consecutive executions, where $\hat{\omega}$ is what we call the workload function of the task. The first bar in the figure equals the WCET of the task. The solid line denoted by δ shows the upper bound specified by the σ and ρ parameters. Figure 1c shows the average worst-case execution time corresponding with the worst-case cumulative execution time of Figure 1b. The solid line denoted by α is the average worst-case execution time corresponding to the upper bound δ of Figure 1b.

Figure 1c also contains the average worst-case execution time when only using the WCET of the task (the horizontal gray line denoted by WCET). It illustrates the throughput improvement that can be achieved when using the (σ, ρ) workload characterization for this example. The computed minimum throughput of this example doubles according to the temporal analysis if the (σ, ρ) workload characterization is used instead of the WCET.

B. Conservative Temporal Analysis

To ensure that a dataflow model is temporally conservative to a task graph, sufficient conditions on the relation between

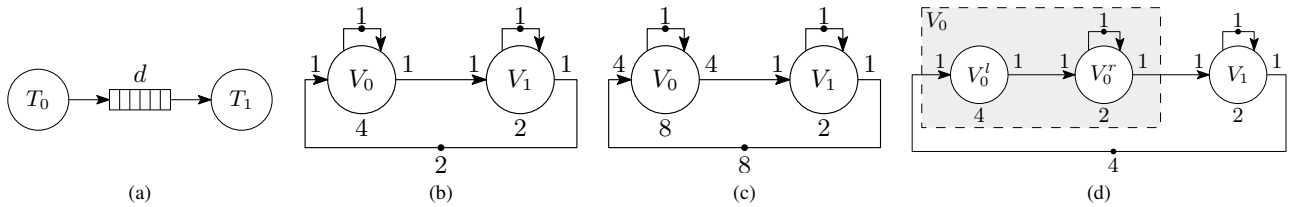


Figure 2. (a) Contains a task graph with two tasks and a buffer. This task graph is modeled as a dataflow model with two dataflow components in (b). For task T_0, information on the cumulative execution time of consecutive executions is exploited in two different ways (c) and (d)

such a dataflow model and a corresponding task graph are presented in [10]. This means that the temporal analysis results (minimum throughput and maximum latency) calculated using the dataflow model are also valid for the corresponding task graph.

The remainder of this subsection summarizes these results from [10]. The sufficient conditions are based on a one-to-one correspondence between the task graph and the dataflow model. Containers in the buffers of the task graph correspond with tokens in the dataflow model. Finite buffers in the task graph are modeled with two queues in the dataflow model. One forward queue, modeling the data production, and one backward queue modeling the back-pressure of the finite buffer. The number of initial tokens on the forward and backward queues represent the number of initially available full and empty containers in the buffer respectively. The sum of the initial tokens on these queues corresponds with the buffer size.

Tasks in the task graph correspond with dataflow components in the dataflow model. A dataflow component is a set of actors. The execution of a task corresponds with a firing of the dataflow component. A task is externally enabled when sufficient containers are available on all adjacent buffers. The corresponding dataflow component is externally enabled if the tokens that correspond with the required containers are present on the queues. We call the time at which a execution or firing is externally enabled the enabling time.

With the one-to-one correspondence between the task graph and the dataflow model, it is sufficient to show that for each task the corresponding dataflow component is temporally conservative. This can be done by showing that when the enabling time of execution i of a task is smaller or equal to the enabling time of firing i of the corresponding dataflow component, also the time at which execution i finishes is less or equal to the time at which firing i finishes. With this it can be shown that containers never arrive later in the buffers than the corresponding tokens arrive at their queues and thus the temporal analysis results of the dataflow model are conservative.

The rectangle V_0 in Figure 2d shows how we model the worst-case temporal behavior of a task with a (σ, ρ) workload characterization as a dataflow component. The dataflow component consists of two dataflow actors, one modeling the rate and together modeling the latency. The dataflow component V_0 of Figure 2d consists of the rate actor V_0^r (with self-edge to prevent overlapping firings) and an actor V_0^l (without self-edge). The idea is that the rate actor uses ρ as its firing duration such that the minimum throughput is computed using a value closer to the average execution time than the WCET. The actor V_0^l ensures that the dataflow component is still temporally conservative to the corresponding task. In the next subsection we use an example to illustrate what the benefits of modeling the (σ, ρ) workload characterization with two actors are.

C. Dataflow Modeling Example

Consider the task graph of Figure 2a which contains two tasks, T_0 and T_1 which communicate via a finite buffer of d locations. We assume a constant execution time for task T_1 of 2 time units. This task is modeled with a dataflow component V_1 in the other three figures of Figure 2. For task T_0 we assume that we have information on the maximum execution time of consecutive executions. The WCET of this task is equal to 4 and we know that four consecutive executions of the task take maximally 8 time units which means that the average execution time equals 2. The worst-case execution time trace for four consecutive executions can be used to find corresponding values for σ and ρ which are 6 and 2 time units respectively (see Section IV-D). We model task T_0 as a dataflow component V_0 in three different ways. The buffer is modeled with two queues between the dataflow components. The number of required initial tokens on the queue from V_1 to V_0 corresponds with the required capacity d of the buffer between the two tasks.

The simplest case is illustrated in Figure 2b. The two dataflow components are modeled with a single dataflow actor, each having a firing duration equal to the WCET of their corresponding task. For this case, two initial tokens are required to ensure the highest minimum throughput which is for this case equal to $\frac{1}{4}$ tokens per time unit. Two tokens are sufficient because then the self edge of V_0 forms the critical cycle which determines that the Maximum Cycle Mean (MCM) [19] equals 4. The throughput of the dataflow model is the inverse of the MCM.

We would now like to exploit the information on the execution time of consecutive executions to increase the minimum throughput that can be guaranteed by the temporal analysis method. A solution for this is illustrated in Figure 2c. It combines four firings of actor V_0 into one firing which now consumes and produces four times the number of tokens consumed/produced in the original dataflow model. One firing of V_0 now corresponds with four executions of T_0. We know that the cumulative execution time of four consecutive executions of T_0 is not equal to four times the WCET, but is actually maximally 8 time units. We thus can use a firing duration of 8 time units for the aggregated firing. This increases the minimum throughput to $\frac{4}{8}$ tokens per time unit. To achieve this throughput, eight initial tokens are needed. This corresponds to a buffer size, d, of eight locations which thus needs to be four times as high to guarantee the increased minimum throughput. Note that for this solution the more expressive SDF model is required instead of the original HSDF model. The solution proposed in this paper can be used in combination with the original type of dataflow model.

Figure 2d shows how including execution time information of consecutive executions in a dataflow model can refine

the dataflow actor V_0 of Figure 2c. In the remainder of this paper we will focus on this solution. Actor component V_0 of Figure 2d now consists of two actors instead of only one. These two actors together correspond to task T_0. We have one actor without a self-edge (V_0^l) and one actor with a self-edge (V_0^r). The firing duration of actor (V_0^r) equals ρ, which in this example corresponds to the average worst-case execution time of the producing task (2 time units). The firing duration of actor V_0^l is equal to $\sigma - \rho$, which is equal to 4 in this example. Modeling the information on consecutive executions like this, still leads to a guaranteed minimum throughput of $\frac{1}{2}$ tokens per time unit. However according to the temporal analysis for this dataflow model, four instead of eight initial tokens are sufficient to achieve this throughput.

In the next Section we show that this last modeling alternative can be proven to be temporal conservative to the task graph.

IV. Dataflow Analysis Using Sigma, Rho Workload Characterization

This section first formalizes the (σ, ρ) workload characterization. We then discuss in detail how to include the information captured in a (σ, ρ) workload characterization of a task in a dataflow component and prove the temporal conservativeness of this dataflow component. We then show how to use the (σ, ρ) workload characterization in combination with the temporal analysis of the settings of a run-time scheduler. We conclude this section with a method to transform execution time information of a finite number of consecutive executions to a (σ, ρ) workload characterization which contains execution time information for every number of consecutive execution.

A. Sigma, Rho Workload Characterization Formalization

The (σ, ρ) workload characterization contains execution time information of consecutive executions of a task. The ρ parameter is required to be an upper bound on the average execution time of an infinite number of executions of a task. The σ parameter must account for the maximum possible deviation from this average execution time. The (σ, ρ) workload characterization forms an upper bound on the maximum sum of execution times of consecutive executions of a task.

The upper bound defined by the (σ, ρ) workload characterization needs to hold for every number of consecutive executions and the two parameters should be chosen such that Equation (1) holds.

$$\forall_{k,i}\ k \leq i : \sum_{j=k}^{i} x(j) \ \leq\ \hat{\omega}(i-k+1) \ \leq\ \delta(i-k+1) \quad (1)$$

With $x(j)$ the execution time of execution j, $\hat{\omega}(n)$ the worst-case cumulative execution time and the function δ as follows:

$$\delta(n) = \sigma + (n-1) \cdot \rho \quad (2)$$

This means that $\delta(n)$ should be an upper bound on the sum of execution times of any sequence of n consecutive task executions. In particular, $\delta(1)$ should be at least as large as the WCET of the task. Note that if we choose $\sigma = \rho = WCET$, the original constant workload characterization is obtained.

Example 1:
Consider a task which has an alternating execution time of 8 *and* 4 *time units. The average execution time of an infinite number of executions equals* 6 *time units. When we select* σ *equal to* 8 *and* ρ *equal to* 6, *Equation* (1) *holds.*

Figure 3. The one-to-one relation between a task T in a task graph and a dataflow component D in a dataflow model

Using the definition of the (σ, ρ) workload characterization we can prove the conservativeness of the one-to-one relation between a task graph with (σ, ρ) workload characterizations for the tasks and a dataflow model which uses the σ and ρ parameters.

B. Conservative Dataflow Analysis with Sigma, Rho Workload Characterization

Figure 3 shows the one-to-one relation between a task and a dataflow component. Let $e(i)$ from this figure be the external enabling time of execution i of the task, i.e. the time at which the required number of containers is present on the adjacent buffers. The dataflow component is externally enabled at time $\hat{e}(i)$ which is the time at which the required tokens, corresponding with the containers in the task graph, are present on the adjacent queues. Let $f(i)$ be the time at which execution i of the task finishes, and we use $\hat{f}(i)$ as the finish time of the dataflow component. We assume that tasks execute self-timed, i.e. they start as soon as they are enabled. Next to this, we define task executions such that different executions of the same task do not overlap in time. Furthermore, we will use $x(i)$ for the execution time of execution i of the task which is defined as the interval $f(i) - f(i-1)$ when the two executions i and $i-1$ of the task execute without interruption, i.e. $e(i) \leq f(i-1)$.

As shown in [10], when proving that the dataflow component is temporally conservative to the corresponding task it is sufficient to prove that when the enabling time of firing i of the dataflow model is later or equal to the enabling time of execution i of the task, also the finish time of firing i is later or equal to the finish time of execution i of the task, i.e.

$$\forall_{i \geq 0}\ (\forall_j\ e(j) \leq \hat{e}(j)) \Rightarrow f(i) \leq \hat{f}(i) \quad (3)$$

Given that (3) holds for every component in the dataflow model, it has been proven in [20] that the results concerning the temporal behavior of the dataflow model are conservative (pessimistic) to the task graph.

We now first introduce the notion of a consecutive execution with which we can derive an upper bound on the finish time of execution i of a task T. Afterwards, the dataflow component is presented for which it is proven that (3) holds. The derived upper bound on the finish time of execution i of task T is used in this proof.

Definition 1:
Execution i *of task* T *is part of a consecutive execution that starts with execution* k *of* T *if for all executions* $k < j \leq i$ *of* T *holds that* $e(j) \leq f(j-1)$.

Every execution i is part of a consecutive execution because it either belongs to a consecutive execution that starts with execution k or it is the first firing of a consecutive execution,

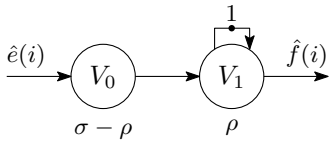


Figure 4. Conservative dataflow component for (σ, ρ) characterization

and thus $i = k$. Therefore, the finish time of execution i can be defined as:

$$f(i) = e(k) + \sum_{j=k}^{i} x(j) \tag{4}$$

The information on the execution times as provided by the (σ, ρ) workload characterization of task T can now be used to find an upper bound $f^u(i)$ on the finish time of execution i of T. We substitute (1) and (2) in (4) and find the upper bound:

$$\begin{aligned} f(i) \leq f^u(i) &= e(k) + \delta(i - k + 1) \\ &= e(k) + \sigma + (i - k) \cdot \rho \end{aligned} \tag{5}$$

Applying the bound $f^u(i)$ as given by (5) is difficult in general because the start of the consecutive execution cannot always be determined. We therefore rewrite $f^u(i)$ to an upper bound that does not depend on the knowledge of consecutive executions. This upper bound only uses the external enabling time of the current execution and the finish time of the previous execution and is presented in Theorem 1.

Theorem 1:
Given a (σ, ρ) characterization of task T, the upper bound on the finish time of execution i of task T is given by:

$$f(i) \leq f^u(i) = \max\left(e(i) + \sigma - \rho, f^u(i-1)\right) + \rho \tag{6}$$

Proof: For any execution i of task T we can distinguish two cases. Either $e(i) > f(i-1)$ which means that i is the first execution in a consecutive execution, or $e(i) \leq f(i-1)$. If $e(i) > f(i-1)$, then $k = i$ and $f^u(i) = e(i) + \sigma + (i - i) \cdot \rho$. If $e(i) \leq f(i-1)$, execution i starts as soon as the previous execution finishes and the time between the two finishes is equal to $f^u(i) - f^u(i-1) = \rho$ which follows from substitution of (5). Combining these two cases leads to the max expression: $f^u(i) = \max\left(e(i) + \sigma, f^u(i-1) + \rho\right)$ which can be rewritten to Equation (6). ∎

Figure 4 shows the dataflow component with which we model the (σ, ρ) workload characterization. It consists of one actor without a self-edge and a firing duration of $\sigma - \rho$ (actor V_0) and one actor with self-edge with a firing duration of ρ (actor V_1). Note that this dataflow component can consume and produce an arbitrary amount of tokens. It can thus for example be used as a component in an SDF graph or in a VRDF graph.

For this dataflow component we can derive the following max-expression by making use of the max-plus semantics of dataflow actors:

$$\hat{f}(i) = \max\left(\hat{e}(i) + \sigma - \rho, \hat{f}(i-1)\right) + \rho \tag{7}$$

We now show that the dataflow component as shown in Figure 4 is conservative to its corresponding task T.

Lemma 1. *Given the (σ, ρ) workload characterization of a task T, Equation* (3) *holds for the relation shown in Figure 3 if the dataflow component is chosen as shown in Figure 4.*

Proof: With (6) and (7) and assuming $f^u(-1) \leq \hat{f}(-1)$ we can prove with induction that $(\forall_j\ e(j) \leq \hat{e}(j)) \Rightarrow f^u(i) \leq \hat{f}(i)$ holds for every execution i of task T. This suffices because $f(i) \leq f^u(i)$ holds.
Base: $(\forall_j\ e(j) \leq \hat{e}(j)) \Rightarrow f^u(0) \leq \hat{f}(0)$ because $e(0) \leq \hat{e}(0)$ and $f^u(-1) \leq \hat{f}(-1)$ thus $\max(e(0) + \sigma - \rho, f^u(-1)) + \rho \leq \max(\hat{e}(0) + \sigma - \rho, \hat{f}(-1)) + \rho$.
Step: Given that $(\forall_j\ e(j) \leq \hat{e}(j)) \Rightarrow f^u(i) \leq \hat{f}(i)$ holds we need to show that if $\forall_j\ e(j) \leq \hat{e}(j)$ holds, also $f^u(i+1) \leq \hat{f}(i+1)$ holds. Because $(\forall_j\ e(j) \leq \hat{e}(j))$ we know that $f^u(i) \leq \hat{f}(i)$ and $e(i+1) \leq \hat{e}(i+1)$ and thus also $\max(e(i+1) + \sigma - \rho, f^u(i)) + \rho \leq \max(\hat{e}(i+1) + \sigma - \rho, \hat{f}(i)) + \rho$. ∎

C. Run-Time Scheduling

In [10] it is shown that run-time budget schedulers allow for an upper bound on the finish times of task executions. Budget schedulers guarantee a task a minimum amount of executing time B in a maximum time interval P. The method from [10] does not assume one single execution time for tasks and therefore allows us to combine it with the (σ, ρ) workload characterization.

We start this section with the upper bound on the finish times of task executions as derived in [10]. We then combine this upper bound with the (σ, ρ) workload characterization. The conservative dataflow component is derived with a similar method as in the previous section.

As shown in [10] an upper bound on the finish times is $f^w(i)$ with:

$$f(i) \leq f^w(i) = e(k) + \sum_{j=k}^{i} x(j) + (P - B) \left\lceil \frac{\sum_{j=k}^{i} x(j)}{B} \right\rceil \tag{8}$$

We can combine this upper bound with the knowledge of the (σ, ρ) workload characterization of task T and can rewrite it to a new upper bound $f^x(i) \geq f(i)$:

$$f^x(i) = e(k) + \sigma + (i-k) \cdot \rho + (P - B) \left\lceil \frac{\sigma + (i-k) \cdot \rho}{B} \right\rceil \tag{9}$$

Again, we provide an upper bound on the finish times which does not depend on the knowledge of consecutive executions:

Theorem 2:
For every scheduler that guarantees a task T a minimum amount of time B in every interval of time P and given a (σ, ρ) workload characterization for T, an upper bound on the finish time of execution i is given by:

$$\begin{aligned} &f(i) \leq f^y(i) \ \text{ with} \\ &f^y(i) = \max\left(e(i) + (P - B) + \tfrac{P \cdot (\sigma - \rho)}{B}, f^y(i-1)\right) + \frac{P \cdot \rho}{B} \end{aligned} \tag{10}$$

Proof: As in [10], we conservatively substitute $\lceil x \rceil$ by $x + 1$ in (9) and find a new upper bound $f^y(i)$:

$$f^x(i) \leq f^y(i) = e(k) + P - B + \frac{P \cdot (\sigma + (i-k) \cdot \rho)}{B} \tag{11}$$

For any execution i we distinguish two cases $e(k) > f^y(i-1)$ in which $i = k$ and $f^y(i) = e(i) + P - B + \frac{P \cdot \sigma}{B}$ and we have

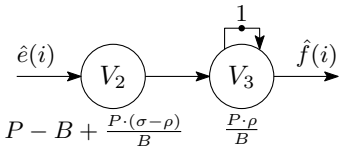


Figure 5. A conservative dataflow component for the (σ, ρ) characterization including the effect of budget scheduler settings

the case $e(k) \leq f^y(i-1)$ for which $f^y(i) - f^y(i-1) = \frac{P\cdot\rho}{B}$ holds. When we combine these two cases we get: $f^y(i) = \max\left(e(i) + P - B + \frac{P\cdot\sigma}{B}, f^y(i-1) + \frac{P\cdot\rho}{B}\right)$. Because we know that $f(i) \leq f^x(i)$ and $f^x(i) \leq f^y(i)$ we also know that (10) holds. ■

The dataflow component which combines budget scheduler settings with a (σ, ρ) workload characterization is shown in Figure 5. The max-expression corresponding to this dataflow component is as follows:

$$\hat{f}(i) = \max\left(\hat{e}(i) + (P - B) + \frac{P\cdot(\sigma-\rho)}{B}, \hat{f}(i-1)\right) + \frac{P\cdot\rho}{B} \tag{12}$$

We can now show that the dataflow component as shown in Figure 5 is conservative to its corresponding task T.

Lemma 2. *Given the (σ, ρ) workload characterization of a task T and budget scheduler settings P and B, Equation* (3) *holds for the relation shown in Figure 3 if the dataflow component is chosen as shown in Figure 5.*

Proof: Analogous to the proof for Lemma 1 with respectively Equations (10) and (12) instead of (6) and (7) ■

D. Transformation of a Finite Workload Characterization to a Sigma, Rho Workload Characterization

In some cases, only the workload characterization of a finite number of N consecutive executions is known. In this section we define a method for transforming such information on a finite number of consecutive executions to a (σ, ρ) workload characterization, which holds for every number of consecutive executions. Section IV-D2 improves this transformation for the case the WCET of the task is known. The idea behind the transformation to a (σ, ρ) workload characterization is the fact that every N consecutive executions can be individually bounded by the known finite workload characterization.

1) Transformation to Sigma, Rho Workload Characterization: An upper bound on the cumulative execution time for up to N consecutive executions can be defined if the workload characterization of a finite number of consecutive executions is known. This upper bound is formalized as follows:

$$\forall_{i,n}\ 1 \leq n \leq N: \sum_{j=i}^{i+n-1} x(j) \leq \eta(n) \quad \text{with} \quad \eta(n) = \varphi + (n-1)\cdot\gamma \tag{13}$$

Where γ is an upperbound on the average worst-case execution time of every N consecutive executions and φ accounts for the maximum possible deviation from this average.

Because this finite workload characterization holds for every $n \leq N$ consecutive executions, we can transform this characterization to a (σ, ρ) workload characterization that also holds for $n > N$. Equivalent to Equation (1) we have that the following equation needs to hold for the (σ, ρ) workload characterization:

$$\forall_{i,n}\ 1 \leq n: \sum_{j=i}^{i+n-1} x(j) \leq \sigma + (n-1)\cdot\rho \tag{14}$$

To derive a (σ, ρ) workload characterization that holds for every number of consecutive executions we split the consecutive executions into groups. Every number of consecutive executions, say n, can be split into $k = \left\lfloor \frac{n-1}{N} \right\rfloor$ groups of exactly N executions plus a group containing the remainder of the n executions: $n - k\cdot N$ executions. Note that $1 \leq n - k\cdot N \leq N$. The sum of the execution times of n executions can thus be written as:

$$\sum_{j=i}^{i+n-1} x(j) = \underbrace{\sum_{j=i}^{i+N-1} x(j) + \ldots + \sum_{j=i+(k-1)\cdot N}^{i+k\cdot N-1} x(j)}_{k \text{ groups of } N \text{ executions}} + \sum_{j=i+k\cdot N}^{i+n-1} x(j) \tag{15}$$

Because the finite workload characterization given in Equation (13) holds for every $n \leq N$ consecutive executions, we can use it to bound the execution times of the groups of consecutive executions of Equation (15):

$$\sum_{j=i}^{i+n-1} x(j) \leq \sum_{j=i+k\cdot N}^{i+n-1} x(j) + k\cdot\eta(N) \tag{16}$$
$$\leq \eta(n - k\cdot N) + k\cdot\eta(N) \tag{17}$$
$$\leq \varphi + (n - k\cdot N - 1)\cdot\gamma + k\cdot(\varphi + (N-1)\cdot\gamma) \tag{18}$$
$$\leq \varphi + (n-1)\cdot\gamma + k\cdot(\varphi - \gamma) \tag{19}$$

We can now approximate k with $\frac{n-1}{N}$

$$\leq \varphi + (n-1)\cdot\gamma + (n-1)\cdot\left(\frac{\varphi-\gamma}{N}\right) \tag{20}$$
$$\leq \varphi + (n-1)\cdot\left(\frac{\varphi + (N-1)\cdot\gamma}{N}\right) \tag{21}$$

From this we conclude that if we choose $\sigma = \varphi$ and $\rho = \frac{\varphi+(N-1)\cdot\gamma}{N}$, Equation (14) holds and we thus have derived a valid (σ, ρ) workload characterization.

Example 2:
Assume we know from the worst-case execution time trace of a task that the worst-case cumulative execution time of up to $N = 4$ consecutive executions can be bounded by $8 + (n-1)\cdot 4$ with $n \leq N$ the number of consecutive executions. An example of such a worst-case trace is shown in Figure 6. The finite workload characterization can be transformed in a (σ, ρ) workload characterization with $\sigma = 8$ and $\rho = 5$ which is illustrated by the dashed line. Because the first 4 consecutive executions are determined using the worst-case trace, we know that the next 4 consecutive executions cannot take more time than these first 4 consecutive executions. This is illustrated in Figure 6 with the dotted lines. As can be seen in Figure 6 the bound corresponding to the computed (σ, ρ) workload characterization forms an upper bound to the first 4 consecutive executions as well as the repeated executions.

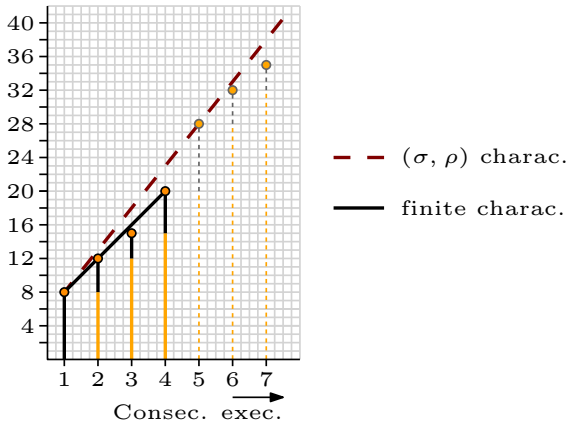


Figure 6. Example transformation of a finite workload characterization to a (σ, ρ) characterization that holds for every number of consecutive executions

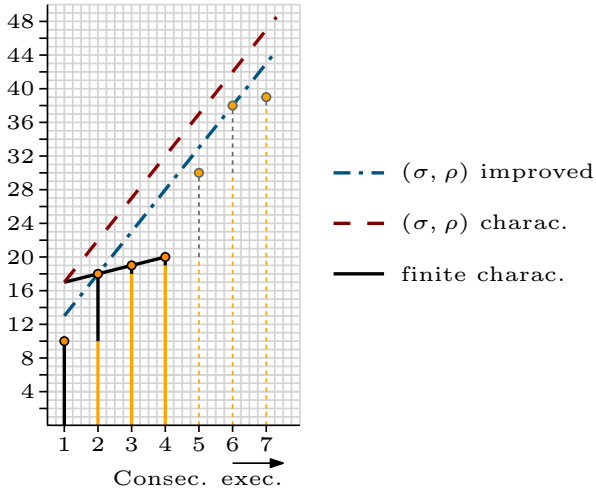


Figure 7. Transformation of a finite workload characterization to an (improved) (σ, ρ) characterization

2) Improved Transformation by Using the WCET: The transformation defined in the previous section can be improved if the WCET of the task is known. The value for σ derived in the previous section might be too conservative if φ is larger than the WCET. We show in this section that we can decrease, and thus improve, the value of σ such that it is higher than the WCET and such that is still conservative to the finite workload characterization for $n \geq 2$. We choose ρ equal to the value derived in the previous section, $\rho = \frac{\varphi+(N-1)\cdot\gamma}{N}$, and only derive a new value for σ. Note that we assume a value of N larger or equal to two.

Again a value for σ needs to be found such that Equation (14) holds. We apply case distinction on n to find this value. We consider two cases: $n = k \cdot N + 1$ and $n = k \cdot N + a$ with $2 \leq a \leq N$. We derive upper bounds for both cases and use the maximum of these two upper bounds as the (σ, ρ) workload characterization.

If $n = k \cdot N + 1$ we can split the sequence of executions in k groups of N executions plus one extra execution. For these values of n we have $k = \frac{n-1}{N}$. Because we know that the execution time of every execution is less or equal to the WCET we get:

$$\sum_{j=i}^{i+n-1} x(j) \leq WCET + k \cdot \eta(N) \tag{22}$$

$$\leq WCET + \frac{n-1}{N} \cdot \eta(N) \tag{23}$$

$$\leq WCET + (n-1) \cdot \rho \tag{24}$$

In the other case we have $n = k \cdot N + a$ with $2 \leq a \leq N$. In this case we have $k = \frac{n-a}{N}$ and we use the fact that $(a-2) \geq 0$. Furthermore, we know that $\rho \geq \gamma$ and thus $\gamma - \rho \leq 0$. With this information we derive the following upper bound:

$$\sum_{j=i}^{i+n-1} x(j) \leq \eta(a) + k \cdot \eta(N) \tag{25}$$

$$\leq \varphi + (a-1) \cdot \gamma + k \cdot (\varphi + (N-1) \cdot \gamma) \tag{26}$$

With $\rho = \frac{\varphi + (N-1)\cdot\gamma}{N}$ and $k = \frac{n-a}{N}$ we have

$$\leq \varphi + (a-1) \cdot \gamma + (n-a) \cdot \rho \tag{27}$$

$$\leq \varphi + \gamma - \rho + (n-1) \cdot \rho + (a-2) \cdot (\gamma - \rho) \tag{28}$$

$$\leq \varphi + \gamma - \rho + (n-1) \cdot \rho \tag{29}$$

We combine these two cases by choosing $\sigma = \max(WCET, \varphi + \gamma - \rho)$. Together with $\rho = \frac{\varphi+(N-1)\cdot\gamma}{N}$ we know that both cases are bounded and that Equation (14) holds for every n. This is an improvement of the (σ, ρ) workload characterization derived in the previous section because $\varphi \geq WCET$ and $\rho \geq \gamma$.

Example 3:
In Figure 7, the worst-case trace of 4 consecutive executions of a task is shown which can be bounded by $17 + (n-1) \cdot 1$. We furthermore know that the WCET of this task is equal to 10. The dashed line shows the original (σ, ρ) workload characterization with $\sigma = 17$ and $\rho = 5$. As can be seen in the figure, this (σ, ρ) workload characterization is too conservative since it can be shifted down and still be an upper bound to all the consecutive executions. The dash-dotted line illustrates the improved (σ, ρ) characterization with $\sigma = \max(10, 17 + 1 - 5) = 13$ and $\rho = 5$. It still is an upper bound for all the consecutive executions in the worst-case trace but improves the original (σ, ρ) workload characterization.

V. Cyclic Temporal Behavior Combined with Sigma, Rho Workload Characterization

CSDF graphs can be used to model the temporal behavior of tasks with explicit cyclic temporal behavior. In this section we combine the modeling of this cyclic behavior with information captured in a (σ, ρ) workload characterization. This (σ, ρ) workload characterization does contain different (σ, ρ) information for each phase of the cyclo-static period.

Cyclic behavior of a task can be bounded by distinguishing each phase of the cyclo-static period:

$$\forall_{k,i}\ k \leq i : \sum_{j=k}^{i} x(j) \quad \leq \sum_{j=k}^{i} \hat{x}_{j\%M} \tag{30}$$

With $\hat{x}_m$ the worst-case execution time of phase m of the cyclo-static period, $\%$ the modulo operation and M the number of phases.

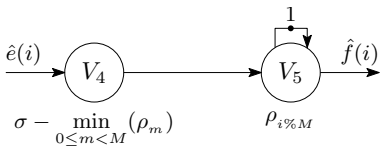


Figure 8. Conservative dataflow component for (σ, ρ) characterization in a CSDF graph

Instead of one worst-case execution time of a phase we have (σ, ρ) information for each phase. We have ρ_m the average duration of phase m and $\sigma - \rho_m$ the maximum deviation from this average over all executions of all phases. Instead of (30) we have:

$$\forall_{k,i}\ k \le i : \sum_{j=k}^{i} x(j) \quad \le \sigma + \sum_{j=k+1}^{i} \rho_{j\%M} \tag{31}$$

Note that Equation 31 is a generalization of the (σ, ρ) characterization shown in Equations (1) and (2) for non cyclo-static dataflow graphs.

This (σ, ρ) characterization can be used to define an upper-bound, $f^z(i)$, on the finish of execution i by substituting (31) in (4):

$$f(i) \le f^z(i) = e(k) + \sigma + \sum_{j=k+1}^{i} \rho_{j\%M} \tag{32}$$

Similar to the method used in Section IV-B we can derive an upper bound on the finish times of a task which does not depend on the knowledge of consecutive executions:

Theorem 3:
Given a (σ, ρ) characterization of a task T that has cyclic temporal behavior, the upper bound on the finish time of execution i of task T is given by:

$$f(i) \le \max\left(e(i) + \sigma - \min_{0 \le m < M}(\rho_{m\%M}), f^z(i-1)\right) + \rho_{i\%M} \tag{33}$$

Proof: We first rewrite $f^z(i)$. For any execution i we distinguish two cases. The first case: $e(k) > f^z(i-1)$ in which $i = k$ and with (33): $f^z(i) = e(i) + \sigma$ and we have the case $e(k) \le f^z(i-1)$ for which $f^z(i) - f^z(i-1) = \rho_{i\%M}$ holds. When we combine these two cases we get:

$$\begin{aligned} f^z(i) &\le \max\left(e(i) + \sigma, f^z(i-1) + \rho_{i\%M}\right) \\ f^z(i) &\le \max\left(e(i) + \sigma - \rho_{i\%M}, f^z(i-1)\right) + \rho_{i\%M} \\ f^z(i) &\le \max\left(e(i) + \sigma - \min_{0 \le m < M}(\rho_{m\%M}), f^z(i-1)\right) + \rho_{i\%M} \end{aligned} \tag{34}$$

With (32) and (34) we also know that (33) holds. ■

Note that the left term of the max-expression is rewritten such that it is constant. This is because CSDF actors are required to have a self-edge. The left actor of the dataflow component of Figure 8 is therefore a normal SDF actor with a constant firing duration.

With this result we can show, analogous to the proof in Section IV-B, that the dataflow component shown in Figure 8 models the temporal behavior of a task with cyclic temporal behavior in combination with a (σ, ρ) workload characterization conservatively. Note that actor V_5 in this figure is a CSDF actor.

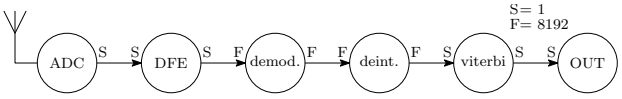


Figure 9. Task graph of a DVB-T application

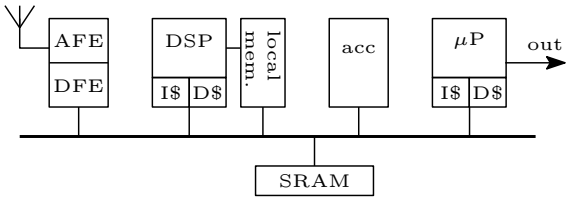


Figure 10. Software defined radio architecture with a shared bus, a shared SRAM, and processors with instruction and data caches.

VI. Case Study

In this section we show how the (σ, ρ) workload characterization can be used to improve the accuracy of the temporal analysis of a DVB-T receiver application. Next to that we give an example of a car-radio application that shows the effect of a latency constraint on the temporal analysis results that can be obtained using a (σ, ρ) workload characterization. We also use the (σ, ρ) workload characterization to improve the temporal results of a MP3 playback application which has a data-dependent behavior.

A. DVB-T Receiver Application

Figure 9 shows the task graph of a DVB-T receiver application. The tasks in the task graph communicate via finite FIFO buffers. The demodulation task, *demod.*, and the deinterleaving task, *deint.*, process and communicate complete symbols which consist of 8192 samples. The other tasks process and communicate per sample.

Figure 10 illustrates the Multiprocessor System-on-Chip (MPSoC) architecture for software defined radio applications on which the DVB-T receiver application is executed. It contains an analogue front-end (AFE), a digital front-end bandpass filter (DFE), a Digital Signal Processor (DSP) with instruction and data caches for demodulation, an accelerator for deinterleaving and error correction, a shared bus, a shared SRAM, and a microprocessor (μP) for control and interfacing with peripherals. The caches, the shared bus, and the shared SRAM memory port are hardware elements that can cause a significant difference between the WCET and the average execution time of a task executing for example on the DSP.

For this case study we consider the demodulation task because this task will be executed on the DSP while the other tasks are executed on dedicated hardware (front-ends and the accelerator) and have a fixed execution time.

The demodulation task has variability in its execution time due to sharing of the used memory port, control flow in the task and caches in the DSP. Despite this variability the DVB-T receiver has a strict throughput constraint. The demodulation task should, on average, process one frame per 952μs. Traditionally the WCET, which includes the maximum variability, should be used to give guarantees on the throughput. Typically sufficient buffer sizes need to be determined to meet the throughput

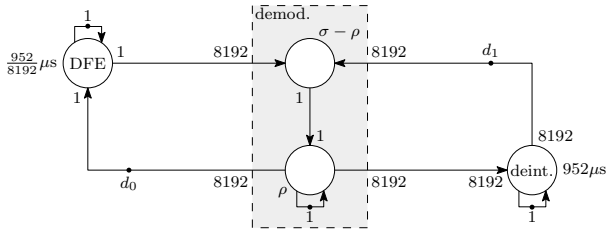


Figure 11. Dataflow model of the *DFE*, *demod.* and *deint.* tasks. The *demod.* task is modeled using its (σ, ρ) workload characterization

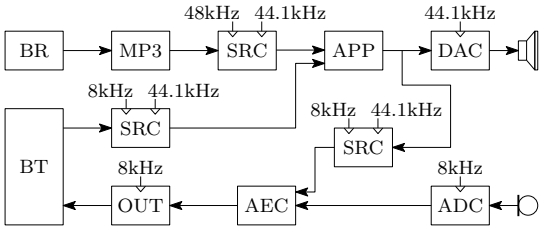


Figure 12. Block diagram of an Audio Echo Cancellation application

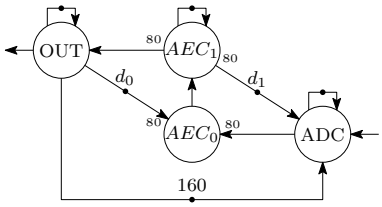


Figure 13. SDF model of a part of an Audio Echo Cancellation application

constraint of 1 frame per 952μs. However, if the WCET of the demodulation task is larger than 952μs, analysis methods that use only this information would indicate that the throughput constraint cannot be met, independent of the buffer sizes.

The throughput constraint can be met if we for example know that the average execution time of every eight consecutive executions is less or equal to 952μs and that of every eight executions, maximally four consecutive executions have a higher execution time than 952μs. The execution time of these longer executions is maximally 15% higher than 952μs.

With this information we can define a (σ, ρ) workload characterization. According to the information, a ρ equal to 952μs forms an upper bound on the cumulative execution because the maximum cumulative execution time of every 8 consecutive executions equals $8 \cdot 952\mu$s. Maximally four per eight executions take longer than 952μs and we know that the maximum possible deviation from the average execution time is caused by these four executions taking 15% longer than 952μs. The maximum deviation is thus 60%. After eight consecutive executions, the cumulative execution time is again smaller or equal to the average execution time which means that σ equal to $1.6 \cdot 952\mu$s and ρ equal to 952μs forms an upper bound on the cumulative execution time of the demodulation task.

With these values for σ and ρ we can compute the buffer sizes which do allow the demodulation task to process one frame per 952μs on average. Figure 11 shows the dataflow model of the (σ, ρ) workload characterization corresponding to the demodulation task. It also shows the adjacent tasks, *DFE* and *deint.*, which have a constant execution time of $\frac{952}{8192}\mu$s and 952μs respectively.

The required number of initial tokens such that the required throughput is met can be calculated with buffer sizing methods such as [3]. Calculation of this required number of initial tokens for this dataflow model shows that $2.6 \cdot 8192$ tokens are required for d_0 and $3 \cdot 8192$ tokens are required for d_1.

B. Audio Echo Cancellation Application

In this section we model the temporal behavior of a car-radio application. The application is taken from [7] in which a SDF model of the application is presented. We take a part of the model to illustrate the effect of a latency constraint on the temporal analysis results that can be obtained using a (σ, ρ) workload characterization.

Figure 12 shows the block diagram of the application. A Bluetooth device (*BT*) is used to make a phone call. Simultaneously, an MP3 file is playing at a lower volume. Next to that, the application contains an Audio Echo Cancellation (AEC) algorithm to prevent the howling effect (feedback loop between microphone and speaker) and to prevent the sound from the speaker to be transmitted via the Bluetooth device. The latency between the microphone and the Bluetooth device should be small to prevent annoying delays in the speech.

We focus on the AEC task in this case-study. We use a (σ, ρ) workload characterization to model its varying temporal behavior which occurs due to sharing of resources. Figure 13 shows the SDF graph of the pipeline from microphone via the *ADC*, *AEC* and *OUT* tasks to the Bluetooth device. The *ADC* task executes strictly periodic at a frequency of 8kHz. The firing durations of the *AEC* and *OUT* tasks are equal to $\frac{1}{8}$ms. The *AEC* task is modeled with a (σ, ρ) workload characterization where AEC_0 and AEC_1 have firing durations equal to $\sigma - \rho$ and ρ respectively. In [7] the worst-case response time of the *AEC* task is determined to be 9.091ms. The *AEC* task also has an algorithmic delay of 6ms caused by its 48 taps filter. In this case-study we assume that the worst-case response time of the *AEC* task is measured and that it can be sporadically higher than 9.091ms. We assume that this behavior can be characterized using values for ρ and σ equal to 9.091 and $1.2 \cdot 9.091 = 10.909$.

Using this information one can compute the required buffer sizes d_0 and d_1 to meet the throughput constraint of 8kHz. However, the dataflow model of Figure 13 also contains a latency constraint which is modeled with the bottom edge. For a correct user experience, we enforce a maximum latency of 26ms between input and output. The algorithmic delay of the *AEC* task is subtracted from this latency and we enforce the remaining 20ms by creating a cycle via the bottom edge. The average transfer rate of the model is equal to 8kHz so a maximum latency of 20ms can be enforced by including this edge in the model with in total $20 * 8 = 160$ initial tokens on the edge.

When this latency constraint is taken into account the conclusion is that no buffer sizes can be found that adhere to all the constraints in the dataflow model. This is because the latency constraint is too tight to allow for the compensation of executions with a larger execution time than the average. When we relax the maximum latency constraint to for example 28ms, the throughput constraint can be met with this (σ, ρ) workload characterization.

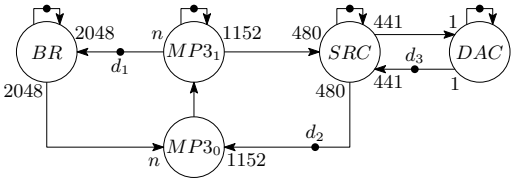


Figure 14. Data-dependent VRDF model of a MP3 playback application

C. MP3 Playback Application

This Section shows how the (σ, ρ) workload characterization can be used in combination with applications containing data-dependent behavior. To illustrate this, we use the MP3 playback application of [5]. This application can be modeled with the VRDF model shown in Figure 14. We model the MP3 task using a (σ, ρ) dataflow component because we assume it has a varying execution time. Actor $MP3_0$ has a firing duration equal to $\sigma - \rho$ and $MP3_1$ has a firing duration of ρ.

The amount of tokens consumed per execution of the MP3 task is dependent on the input and can vary during execution. Each firing, n tokens are consumed and produced. The only information we know about n is that n is less or equal to 960 for an MP3 file with a bit-rate of 320 kbit per second. Furthermore, we have a throughput constraint because the application requires that the DAC executes strictly periodic at a frequency of 44.1kHz.

In [5] the maximum response times of the different tasks are computed given this throughput constraint. The conclusion is that the throughput of the application can only be met if the response time of the MP3 task is less or equal to 24ms. However the model as shown in Figure 14 also allows for worst-case response times larger than 24ms as long as ρ is less or equal to 24ms.

Because the model shown in Figure 14 is a VRDF graph, the method presented in [5] can be used to find sufficient buffer capacities that given a (σ, ρ) workload characterization meet the throughput constraint. If we choose ρ equal to 24ms, σ equal to $1.5 \cdot 24 = 36$ms and all the other firing duration equal to the maximum allowed firing duration, we obtain the following sufficient buffer capacities from the analysis: $d_1 = 6491$, $d_2 = 3836$ and $d_3 = 882$. With a one-actor model using this higher response time for the MP3 task, the throughput constraint cannot be met.

VII. Conclusion

This paper presents a dataflow modeling technique which exploits knowledge about the maximum cumulative execution time of consecutive executions. To exploit this knowledge in a dataflow model, the (σ, ρ) workload characterization has been introduced. It is shown that this (σ, ρ) workload characterization can be temporal conservatively modeled with a dataflow component consisting of two actors. One actor represents the maximum rate and the two actors together model the maximum latency.

We furthermore showed that the (σ, ρ) workload characterization can be combined with the temporal analysis of scheduler settings of a budget scheduler. Next to that, a technique is presented which can derive a (σ, ρ) workload characterization from temporal information on a finite number of consecutive executions.

In the case study we have illustrated that the accuracy of the temporal analysis of a DVB-T application can be improved with the (σ, ρ) workload characterization. It is shown that a temporal analysis method can use an upper bound on the average execution time over a finite number of executions for calculating the throughput. Existing buffer sizing techniques have been used to compute the buffer sizes that are required to cope with the variability in the execution time of the tasks.

References

[1] E. Lee and T. Parks, "Dataflow Process Networks," *Proceedings of the IEEE*, vol. 83, no. 5, pp. 773–801, 1995.

[2] S. Sriram and S. Bhattacharyya, *Embedded Multiprocessors: Scheduling and Synchronization*, ser. Signal Processing and Communications Series. Marcel Dekker, Inc., 2000.

[3] O. Moreira *et al.*, "Buffer Sizing for Rate-Optimal Single-Rate Data-Flow Scheduling Revisited," *IEEE Transactions on Computers*, vol. 59, no. 2, pp. 188–201, 2010.

[4] M. Wiggers *et al.*, "Simultaneous Budget and Buffer Size Computation for Throughput-Constrained Task Graphs," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, 2010, pp. 1669 –1672.

[5] M. Wiggers, M. Bekooij, and G. Smit, "Computation of Buffer Capacities for Throughput Constrained and Data Dependent Inter-Task Communication," in *Design, Automation and Test in Europe (DATE)*. IEEE, 2008, pp. 640–645.

[6] G. Bilsen *et al.*, "Cyclo-Static Dataflow," *IEEE Transactions on Signal Processing*, vol. 44, no. 2, pp. 397–408, 1996.

[7] M. Wiggers *et al.*, "Efficient Computation of Buffer Capacities for Cyclo-Static Real-Time Systems with Back-Pressure," in *Proc. of the IEEE Real Time and Embedded Technology and Applications Symposium*. IEEE Computer Society, 2007, pp. 281–292.

[8] A. Maxiaguine, S. Künzli, and L. Thiele, "Workload Characterization Model for Tasks with Variable Execution Demand," in *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, vol. 2. IEEE Computer Society, 2004, pp. 1040–1045.

[9] E. Lee and D. Messerschmitt, "Synchronous Data Flow," *Proc. of the IEEE*, vol. 75, no. 9, pp. 1235–1245, 1987.

[10] M. Wiggers, M. Bekooij, and G. Smit, "Monotonicity and Run-Time Scheduling," in *Proc. of the ACM Int'l Conf. on Embedded Software*. ACM, 2009, pp. 177–186.

[11] O. Moreira, F. Valente, and M. Bekooij, "Scheduling Multiple Independent Hard-Real-Time Jobs on a Heterogeneous Multiprocessor," in *Proc. of the 7th ACM & IEEE Int'l Conf. on Embedded Software*, 2007, pp. 57–66.

[12] M. Wiggers *et al.*, "Efficient Computation of Buffer Capacities for Multi-Rate Real-Time Systems with Back-Pressure," in *Proc. of the Int'l Conf. on Hardware/Software Codesign and System Synthesis*. ACM, 2006, pp. 10–15.

[13] A. Ghamarian, M. Geilen, T. Basten, and S. Stuijk, "Parametric Throughput Analysis of Synchronous Data Flow Graphs," in *Design, Automation and Test in Europe, 2008. DATE'08*, 2008, pp. 116–121.

[14] M. Wiggers, M. Bekooij, and G. Smit, "Modelling Run-Time Arbitration by Latency-Rate Servers in Dataflow Graphs," in *Proc. of the Int'l Workshop on Software & Compilers for Embedded Systems*. ACM, 2007, pp. 11–22.

[15] S. Quinton, M. Hanke, and R. Ernst, "Formal Analysis of Sporadic Overload in Real-Time Systems," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2012, pp. 515–520.

[16] G. Bernat, A. Colin, and S. Petters, "WCET Analysis of Probabilistic Hard Real-Time Systems," in *Real-Time Systems Symposium, 2002. RTSS 2002. 23rd IEEE*, 2002, pp. 279–288.

[17] X. Hu, T. Zhou, and E. Sha, "Estimating Probabilistic Timing Performance for Real-Time Embedded Systems," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 9, no. 6, pp. 833–844, 2001.

[18] T. Tia *et al.*, "Probabilistic Performance Guarantee for Real-Time Tasks With Varying Computation Times," in *Proc. of Real-Time Technology and Applications Symposium*. IEEE, 1995, pp. 164–173.

[19] R. Reiter, "Scheduling Parallel Computations," *Journal of the ACM (JACM)*, vol. 15, no. 4, pp. 590–599, 1968.

[20] M. Geilen, S. Tripakis, and M. Wiggers, "The Earlier the Better: A Theory of Timed Actor Interfaces," in *Int'l Conf. on Hybrid Systems: Computation and Control (HSCC'11)*, April 2011.