Cyclo-Static DataFlow Phases Scheduling Optimization for the Throughput Constrained Buffer Sizes Minimization Problem

Mohamed Benazouz, Alix Munier-Kordon

Cyclo-Static DataFlow Phases Scheduling Optimization for the Throughput Constrained Buffer Sizes Minimization Problem

Mohamed Benazouz and Alix Munier-Kordon
LIP6, Université Pierre et Marie Curie, Paris (France)
Email: {Mohamed.Benazouz, Alix.Munier}@lip6.fr

***Abstract*—Cyclo-Static DataFlow (CSDF) is a powerful model for the specification of DSP applications. However, as in any asynchronous model, the synchronization of the different communicating tasks (processes) is made through buffers that have to be sized such that timing constraints are met. In this paper, we want to determine buffer sizes such that the throughput constraint is satisfied. This problem has been proved to be of exponential complexity. Exact techniques to solve this problem are too time and/or space consuming because of the self-timed schedule needed to evaluate the maximum throughput. Therefore, a periodic schedule is used. Each CSDF actor is associated with a period that satisfies the throughput constraint and sufficient buffer sizes are derived in polynomial time. However, within a period, an actor phases can be scheduled in different manners which impacts the evaluation of sufficient buffer sizes.**

This paper presents a Min-Max Linear Program that derives an optimized periodic phases scheduling per CSDF actor in order to minimize buffer sizes. It is shown through an MP3 Playback and an H.263 Encoder that this Min-Max Linear Program allows to obtain close to optimal values while running in polynomial time. The impact of phases scheduling on periodic schedulability of applications with critical cycles is also highlighted on a Channel Equalizer.

***Index Terms*—Cyclo-Static DataFlow, Buffer Capacity, Min-Max Linear Program, Digital Signal Processing, High-Level Synthesis.**

I. Introduction and related works

Dataflow paradigm is a powerful Model of Computation (MoC) for timing analysis of system behavior. This process-based MoC represents computation as a set of concurrent processes performing treatments on data streams that are exchanged through buffered channels. Therefore, it is widely used for modelling digital signal processing (DSP) applications. These applications are often associated with real-time constraints. In this paper, we want to determine buffer sizes such that throughput constraints are satisfied.

Initially, Synchronous DataFlow Graphs [1] (SDFG) were adopted. Processes are represented by nodes called actors and buffers are modelled by arcs. Actors consume and produce a constant and fixed number of data. Then, to increase the scope, Cyclo-Static DataFlow graphs (CSDFG) were introduced [2]. In this model, the number of data consumed and produced by an actor is allowed to vary from a firing to the next in a cyclic pattern. This pattern is constant and known at compile time. Its length defines the number of actor phases. Phases can be seen as iterations of a for loop that is represented by the actor. In this for loop, a condition that depends on the iteration number determines the transfer rate towards a buffer. Now, this entire loop can also be modelled by an SDF actor. The resulted single phase of this SDF actor has then for transfer rate the cumulative transfer rates from the pattern. This modelling results in a significant overestimation of buffer capacities as all the necessary data are consumed/produced in one atomic fashion respectively at the start/end of the actor execution [3]. Besides, modelling some processes by an SDF actor may introduce deadlock situation when the actor is involved in a cycle even if the CSDF model is deadlock-free [2], [4]. Furthermore, [5] proves that CSDF modelling enlarge the scope of channels that can be considered. For instance, an extension is provided for shared buffers (under condition) with multiple consumers and producers.

An accurate CSDFG of an application can be obtained through an automatic extraction from the Cycle Accurate Model (CAM) delivered by the synthesis tool at the Register-Trasfer Level (RTL). An approximate modelling at higher level such as the Transactional-Level Modelling (TLM) can also be used under condition of bounding the Worst Case Execution Time (WCET) of actors. This would allow performance evaluation of a design at a higher level without having to generate the RTL.

Another issue related to buffers is preloading feedback buffers, *i.e.* to determine the number of data that should be initially in a feedback buffer. A feedback buffer creates a cycle (closed loop) from the forward and the feedback paths. An empty feedback buffer results in deadlock situation if buffers on the forward path are not sufficiently initialized. In addition, an under-preloaded feedback buffer results in insufficient throughput. Indeed, the first access to the feedback buffer determines the number of data in the cycle just created (closed) and thus the throughput. Non-blocking read can overcome this issue as data are read only when available in the buffer. However, dataflow semantic supposes blocking read. Furthermore, in some High-Level Synthesis tools that need to simulate C code against RTL for verification purpose, insufficiently prefilled feedback buffers cause C and/or RTL simulation to deadlock or stutter [6].

The capacity of a buffer is defined as the maximum number of data that the buffer can store at a time. We associate a cost per unit of storage and we limit our study to buffers for which all data stored are of the same cost. This cost may be related to

the width of the buffer (size of data) or the technology/location of the memory allocated to the buffer. Then, the cost of a buffer is given by its capacity times the cost of one data. Both cost and size are used indifferently in this paper to refer to a buffer cost.

Contribution: The principal contribution of this paper is a Min-Max Linear Program (LP) that derives an optimized periodic phases scheduling per CSDF actor in order to minimize buffer sizes. This is a major improvement to some techniques based on periodic schedules [7], [8] to minimize the overall buffer sizes that satisfies a fixed throughput. While these techniques considered a phases scheduling that takes into account only phases execution times, the Min-Max LP will enable to consider in addition phases transfer rates, resulting then in a better estimation of sufficient buffer sizes.

This is accomplished as follows. First, the throughput constrained buffer sizes minimization problem is formulated by an Integer Linear Program (ILP). The built ILP accepts any valid phases scheduling. Then, it is shown that the lower bound on the capacity of a buffer computed by this ILP is closely related to the phases schedulings of its consumer and producer. Furthermore, it is proved that the analytical approach presented in this paper enables to optimize separately these two phases schedulings. This leads to the conclusion that an actor phases scheduling optimization can be performed in isolation from the other actors. An actor may have several adjacent buffers and the cost per unit of storage may differ from an adjacent buffer to another. These constraints are taken into account and the actor phases scheduling optimization problem is modelled by a Min-Max LP. Once all actors phases scheduling optimization done, they are reported into the ILP to derive the start time of the first execution of each actor with the objective of minimizing sufficient buffer sizes. As it will be demonstrated in the experimental results section, the computed values for buffer sizes are much better than those computed in [7], [8] while the temporal impact due to the additional Min-Max LP calculation remains limited.

Related Works: The problem tackled in this paper has been addressed in [3]. A polynomial heuristic is presented that schedules actor phases periodically with a period that satisfies the throughput constraint. The approach used in [3] is close to network calculus [9] in the way that it uses (linear) upper/lower bounds on the token production / consumption times to derive sufficient buffer capacities. It is shown that the buffer capacity is proportional to the delay between the upper bound on the data production times and the lower bound on the data consumption times. Then, the producer and the consumer phases schedulings are optimized such that this delay is minimized. These linear bounds are defined per buffer in isolation, so are the derived phases schedulings and thus the delays. Indeed, such a technique hides the fact that optimizing a phases scheduling can be done per actor in isolation as it is proved in this paper. Therefore, the technique presented in [3] results, for an actor, in as much phases schedulings as the number of adjacent buffers to this actor. Authors do not specify the technique used to combine these schedulings to derive one phases scheduling per actor. Moreover, it is not clear how the impact of different costs for the adjacent buffers may be integrated, as the technique by construction cannot consider buffer costs. In the experimental results section, it is shown that our algorithm results in a more accurate estimation of sufficient buffer capacities compared to [3].

Exact techniques compute optimal buffer capacities that satisfy the throughput constraint. In such techniques the self-timed execution is used to evaluate the maximum throughput that can be accomplished by a solution. The self-timed execution, called also the as soon as possible (in short *asap*) schedule, consists in executing actors as soon as there are enough data in their input buffers. No polynomial algorithm exists to evaluate the maximum throughput for SDFGs and thus for CSDFGs. The consequence is that related optimization problems are probably not in NP. In [10], it is proved that buffer sizes minimization problem under throughput constraint is NP-complete for Single Rate DataFlow graphs (SRDFG), *i.e.* SDFGs with all transfer rates equal 1. Thus, the problem is NP-Hard for general SDFGs and CSDFGs.

Exact techniques can be of two classes depending on how the maximum throughput is evaluated. One way to evaluate this throughput for a CSDFG is through a translation to the equivalent Single Rate DataFlow graph (in short SRDFG) [2]. The Maximum Cycle Ratio analysis [11] can then be applied. However, the obtained SRDF graph is of exponential size on the size of the initial CSDF graph. Thus, techniques based on this approach are too space and time consuming to handle real-life DSP applications.

To the best of our knowledge, [12] proposes the only technique of the second class; the class of techniques that do not need a translation of the CSDFG to the equivalent SRDFG. Using a state-space exploration technique, authors determine the Pareto space of the bi-criterion problem throughput-buffer capacities for marked CSDFGs. Marked CSDFGs are graphs for which the initial number of data in known for all buffers. Thus, uninitialized feedback buffers are not considered. The maximum throughput calculation consists of finding a cycle in the state space to which corresponds a self-timed execution pattern that can be repeated infinitely. The maximum throughput of an actor equals the number of executions in this pattern divided by the duration of the pattern. The number of states to visit before finding a cycle can be exponentially large, especially if the number of actor phases is large. Thus, this maximum throughput analysis has an exponential worst-case complexity. Furthermore, the set of storage distributions to explore is of exponential size on the number of buffers to be sized and is closely related to the values of transfer rates. Therefore, the exact technique presented in [12] has an exponential complexity, which, as a consequence, limits the CSDFG instances that can be treated. Authors propose an approximation technique to reduce the set of storage distributions to be explored, and thus lowers run-times. Moreover, they raise a thought about the usefulness that may have fast heuristics, as the one proposed in this paper, in speeding-up the techniques proposed in [12]. In fact, both the exact and the approximation techniques can start exploring from the result of a heuristic, which may lead to a better solution than the heuristic in much less run-time than the exact technique. This idea can be pushed further in the growing context of high-level synthesis

and design exploration [13], [14]. The ability of high-level synthesis to deliver an RTL quickly enough enables to iterate (explore) over several variations of a design until finding the most suitable solution that satisfies a specified feature (metric). The throughput constitutes one of the most common metrics used during the exploration. A fast heuristic that sizes buffers under a throughput constraint can be of great help in this decision process as it provides a sufficiently fast evaluation tool to allow to go through several designs. Once the design is fixed, exact techniques can be used to obtain optimal values. In fact, potentially excessive run-times of exact techniques do not allow their use during the design exploration process. Thus, heuristics and exact techniques complement each other.

The paper is organized as follows: CSDF graphs are presented in Section II. Conditions for a valid periodic schedule are recalled in Section III. Section IV is dedicated to our actors isolation technique and the constraints linearization step; a necessary steps for our ILP formulation of the buffer sizes minimization problem presented in Section V. The principal contribution, that consists in a Min-Max LP formulation of the phases scheduling optimization problem, is presented in Section VI. Section VII is dedicated to experimental results. We conclude in Section VIII.

II. Cyclo-Static DataFlow Graphs

A Cyclo-Static DataFlow Graph $\mathcal{G} = (T, A)$ is a directed graph where the set of nodes T models actors and the set of arcs A corresponds to buffers. Every actor $t \in T$ is decomposed into $\varphi(t) \in \mathbb{N} - \{0\}$ distinct phases that constitute a periodic execution sequence. As described in the introduction, t can be viewed as a loop of $\varphi(t)$ iterations while every phase represents an iteration. For every value $k \in \{1, \cdots, \varphi(t)\}$, the kth phase of t is denoted by t_k and has a constant duration $\ell_t(k) \in \mathbb{Q}^+$. Notice that, by analogy to an execution of a loop, one execution of the actor $t \in T$ will refer to the ordered execution of all its phases (iterations) $t_1, \cdots, t_{\varphi(t)}$. We denote by $\langle t, n\rangle, n \in \mathbb{N} - \{0\}$, the nth execution of t. Similarly, for every phase $k \in \{1, \cdots, \varphi(t)\}$, $\langle t_k, n\rangle$ denotes the nth execution of the kth phase of t. For every couple $(k, n) \in \{1, \cdots, \varphi(t)\} \times \mathbb{N} - \{0\}$, $Pred\langle t_k, n\rangle$ is the preceding execution phase of $\langle t_k, n\rangle$. More formally,

$$Pred\langle t_k, n\rangle = \begin{cases} \langle t_{k-1}, n\rangle & if\ k > 1 \\ \langle t_{\varphi(t)}, n-1\rangle & if\ k = 1. \end{cases}$$

The execution $\langle t_{\varphi(t)}, 0\rangle$ is fictitious and is only introduced to simplify the definition of $Pred$.

Every arc $a = (t, t') \in A$ represents a buffer $b(a)$ of unbounded capacity from the actor t to the actor t'. $\forall k \in \{1, \cdots, \varphi(t)\}$, $w_a(k)$ data are produced in $b(a)$ at the end of an execution of t_k. To enable the execution of the phase $t'_{k'}, \forall k' \in \{1, \cdots, \varphi(t')\}$, $v_a(k')$ data are needed to be available in $b(a)$. They are consumed before $t'_{k'}$ starts its execution. We define the cumulative number of data produced on a by one execution of the actor t as $w_a \cdot \mathbb{1} = \sum_{k=1}^{\varphi(t)} w_a(k)$. Similarly, we define the cumulative number of data consumed from a by one execution of the actor t' as $v_a \cdot \mathbb{1} = \sum_{k=1}^{\varphi(t')} v_a(k)$. In a DataFlow Graph, data are represented by tokens. Their initial number for each buffer $b(a)$ is reported by $M_0(a) \in \mathbb{N}$. An arc $a = (t, t')$ models the synchronization introduced by a buffer $b(a)$ between the actors t and t'. However, this forward arc cannot by itself model the limited capacity of $b(a)$. To model the bounded capacity of $b(a)$ a backward arc $a' = (t', t)$ is added that models the free space. Hence, $M_0(a')$ reports the number of empty containers initially in $b(a)$. For every $k \in \{1, \cdots, \varphi(t)\}$, $v_{a'}(k) = w_a(k)$ and for every $k' \in \{1, \cdots, \varphi(t')\}$, $w_{k'}(a') = v_{k'}(a)$. The capacity of the buffer $b(a)$ equals the sum $M_0(a) + M_0(a')$ (*see.* Figure 1).

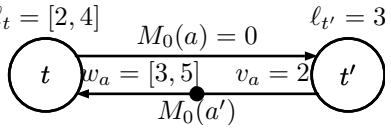

Fig. 1. A bounded buffer $b(a)$ of capacity $M_0(a) + M_0(a')$. $\varphi(t) = 2$ and $\varphi(t') = 1$.

Two phases or two successive executions of an actor are not supposed to overlap. To prevent such a behaviour, a self-arc (t, t) with one token is added for every actor $t \in T$ to guarantee that at most only one phase is running at a time. These arcs are not pictured, but their effect is taken into account when constructing a schedule (*see.* Subsection III-B).

Without loss of generality, it is assumed that the application is modelled using a connected CSDFG. It is also assumed that all the actors are connected by buffers with limited capacity, thus the graph is strongly connected. A path of $\mathcal{G}$ of length $p \in \mathbb{N} - \{0\}$ is defined by a list of actors $\overrightarrow{t^1 t^p} = (t^1, t^2, \cdots, t^p)$ such that for any $k \in \{1, \cdots, p-1\}$, $(t^k, t^{k+1}) \in A$. A cycle is a path such that $t^p = t^1$. The weight of a path $\overrightarrow{t^1 t^p}$ defines a relation between the cumulative transfer rates of the actors and is given by the ratio

$$W(\overrightarrow{t^1 t^p}) = \prod_{a \in \overrightarrow{t^1 t^p}} \frac{w_a \cdot \mathbb{1}}{v_a \cdot \mathbb{1}}.$$

A feasible (valid) schedule of a CSDFG is a function s that associates, for every triple (t, k, n) with $t \in T$, $k \in \{1, \cdots, \varphi(t)\}$ and $n \in \mathbb{N} - \{0\}$, a start time $s(t_k, n)$ for the nth execution of t_k such that the number of token in every arc $a \in A$ remains non negative, *i.e.* no data is read before it is available. The start times of an actor coincide with those of its first phase, *i.e.* $s(t, n) = s(t_1, n)$.

Consistency is a necessary (not sufficient) condition for the existence of a valid schedule within bounded memory [15]. Two methods exist to check consistency. In [2], Bilsen *et al.* extend the condition for the consistency of SDFGs [15] to the case of CSDFGs by considering the cumulative number of tokens produced/consumed by actors. Let us consider the pre-post $|A| \times |T|$ matrix Γ defined by

$$\Gamma_{at} = \begin{cases} w_a \cdot \mathbb{1} & \text{if } a = (t, t'),\ t' \in T \\ -v_a \cdot \mathbb{1} & \text{if } a = (t', t),\ t' \in T \\ 0 & \text{Otherwise.} \end{cases}$$

The CSDFG is consistent if the rank of Γ is $|T| - 1$. Then, the rank of solutions space of the balance equation $\Gamma \cdot r = 0$ is not null. The smallest non trivial ($\forall i \in \{1, \cdots, |T|\}, r(i) \neq 0$)

positive integer solution is called the repetition vector. This vector determines the number of executions per actor in a periodic execution pattern that brings back the CSDFG to a state (*i.e.* the number of tokens per arc) identical to its initial state. In terms of phases, for an actor t^i, $r(i) \times \varphi(t^i)$ phases are executed within this execution pattern.

Another way to check consistency is to prove that all paths that connect a couple of actors have the same weight, otherwise the graph is inconsistent [7]. By analogy to Weighted Event Graph [16], a subclass of PetriNets that is equivalent to SDFG, these graphs are called unitary. A unitary graph is a graph in which all cycles have a weight of one. This property can be checked in $O(|T|\times|A|)$ using the shortest path Bellman-Ford algorithm, which, in general, performs faster than solving the system of linear equations $\Gamma \cdot r = 0$ that is in $O(|T|^3)$ [17].

In the following, we restrict our study to consistent strongly connected CSDFGs as inconsistent graphs will either deadlock or need unbounded buffers.

We consider the existence of an input or output actor $t^\star \in T$ for which a minimum throughput (frequency) of value $\nu_{t^\star}$ is required. The throughput of the CSDFG for a schedule s is defined as

$$\nu(s) = \lim_{n\to\infty} \frac{n}{s(t^\star, n)}$$

and must then verify $\nu(s) \geq \nu_{t^\star}$. For any infinite valid schedule achieved within bounded memory, to ensure no tokens lack/overflow, throughputs of two adjacent actors t and t' with $a = (t, t') \in A$ must verify

$$\nu_t = \frac{v_a \cdot \mathbb{1}}{w_a \cdot \mathbb{1}} \nu_{t'}.$$

This relationship is subsequently used to compute all tasks throughputs. Starting from $t^\star$ for which the throughput $\nu_{t^\star}$ is set, the throughput of every actor t can be computed by a Depth-First Search algorithm [17]. Consistency ensures that whatever the path used to reach $t \in T - \{t^\star\}$, the computed throughput ν_t is always the same, thus the following

$$\forall t \in T - \{t^\star\},\ \nu_t = \nu_{t^\star} / W(\overrightarrow{tt^\star})$$

where $\overrightarrow{tt^\star}$ is a path of $\mathcal{G}$ from t to $t^\star$.

In the next section, some already established results for the feasibility of periodic schedules are presented to enhance understanding of the contributions of this paper.

III. Valid Schedules

Let us consider a bounded buffer $b(a)$ modelled by the couple of arcs $a = (t, t')$ and $a' = (t', t)$. Let us consider also the phases t_k and $t'_{k'}$. The fact that an execution $\langle t'_{k'}, n'\rangle$ has to wait until an execution $\langle t_k, n\rangle$ has finished is called a precedence constraint from $\langle t_k, n\rangle$ to $\langle t'_{k'}, n'\rangle$. The Dataflow semantic imposes a blocking read; an execution cannot start until enough data on input buffers are available. The forward arc a under this semantic induces then an infinite number of precedence constraints from t_k executions to $t'_{k'}$ executions. To prevent data from being overwritten we will also suppose a blocking write semantic; an execution cannot start until enough empty space on output buffers are available. This semantic is equivalent to a blocking read from the backward arc a' on which the number of tokens represents the number of empty containers. As mentioned, a schedule is valid if the number of tokens remains non negative all the time. Then, a valid schedule should respect all the precedence constraints for all the couple of phases. Otherwise, some tokens are consumed before they are available and the number of tokens becomes negative which invalidates the schedule.

A. Precedence constraint

Formally, an arc a induces a precedence constraint from the execution $\langle t_k, n\rangle$ to the execution $\langle t'_{k'}, n'\rangle$ if
1. $\langle t'_{k'}, n'\rangle$ can be executed at the completion of $\langle t_k, n\rangle$,
2. $Pred\langle t'_{k'}, n'\rangle$ can be executed before the end of $\langle t_k, n\rangle$ but not $\langle t'_{k'}, n'\rangle$.

Let us denote by $D_a^+\langle t_k, n\rangle$ (*resp.* $D_a^-\langle t'_{k'}, n'\rangle$) the cumulative number of tokens produced (*resp.* consumed) by t (*resp.* t') in the arc a from the first execution until $\langle t_k, n\rangle$ (*resp.* $\langle t'_{k'}, n'\rangle$) included. Lemma 1, proved in [7], defines a necessary and sufficient condition for the existence of a precedence constraint from $\langle t_k, n\rangle$ to $\langle t'_{k'}, n'\rangle$ in terms of $D_a^+\langle t_k, n\rangle$ and $D_a^-\langle t'_{k'}, n'\rangle$.

Let us denote for every arc $a = (t, t') \in A$,

$$gcd_a = \gcd(w_a \cdot \mathbb{1}, v_a(1), \cdots, v_a(\varphi(t'))),$$
$$gcd'_a = \gcd(v_a \cdot \mathbb{1}, w_a(1), \cdots, w_a(\varphi(t))),$$

where gcd is the greatest common divisor of a given list of non negative integers. For every integer $\alpha \in \mathbb{Z}$ and $\gamma \in \{gcd_a, gcd'_a\}$, we also set

$$\lfloor\alpha\rfloor^\gamma = \left\lfloor \frac{\alpha}{\gamma} \right\rfloor \cdot \gamma \text{ and } \lfloor\alpha\rfloor_\gamma = \begin{cases} \alpha - \gamma & \text{if } \alpha \text{ is a multiple of } \gamma \\ \lfloor\alpha\rfloor^\gamma & \text{otherwise.} \end{cases}$$

Lemma 1 ([7]). *Let $a = (t, t') \in A$ and the couple of executions $\langle t_k, n\rangle$ and $\langle t'_{k'}, n'\rangle$. Then there exists a precedence constraint from $\langle t_k, n\rangle$ to $\langle t'_{k'}, n'\rangle$, iff:*

$$\min\{H_{max}(k), H'_{max}(k, k')\} \geq D_a^+\langle t_k, n\rangle - D_a^-\langle t'_{k'}, n'\rangle \geq H_{min}(k, k')$$

with

$$H_{max}(k) = \left\lfloor -M_0(a) - D_a^+ Pred\langle t_k, 1\rangle \right\rfloor_{gcd_a} + D_a^+\langle t_k, 1\rangle,$$
$$H'_{max}(k, k') = \left\lfloor -M_0(a) + D_a^-\langle t'_{k'}, 1\rangle \right\rfloor_{gcd'_a} - D_a^-\langle t'_{k'}, 1\rangle + w_a(k)$$

and $H_{min}(k, k') = \max\{0, w_a(k) - v_a(k')\} - M_0(a)$.

B. Periodic schedules

Periodic schedules are a mean to overcome the exponential complexity of the self-timed execution analysis. In counterpart to an overestimation of sufficient buffer capacities, the analysis can be made in polynomial time. In a periodic schedule s, each actor and its phases must be fired periodically with a period that ensures the actor consumes/produces an average amount of data per time units that is necessary and sufficient to satisfy the throughput constraint $\nu(s)$. Thus, every actor $t \in T$ is fired periodically with a period $\mu_t = 1/\nu_t \geq \ell_t \cdot \mathbb{1}$ such that
1. $\forall n > 0$, $s(t, n) = s(t, 1) + (n-1)\mu_t = s(t_1, 1) + (n-1)\mu_t$.
2. $\forall k \in \{1, \cdots, \varphi(t)\}$, $s(t_k, n) = s(t, n) + \zeta_t(k)$.
3. $\zeta_t(1) = 0$; $\forall k \in \{2, \cdots, \varphi(t)\}$, $\zeta_t(k) \geq \zeta_t(k-1) + \ell_t(k-1)$ and $\zeta_t(\varphi(t)) + \ell_t(\varphi(t)) \leq \mu_t$.

Condition 1 expresses the periodicity of the executions of t. Condition 2 defines the phases scheduling ζ_t that is used to determine the start times of phases in relation to the start time of the current execution of t. Condition 3 on the phases scheduling ζ_t ensures that two successive phases do

not overlap and that the last phase finishes before the next execution of t, *i.e.* $\langle t, n+1\rangle$, starts.

A periodic schedule is then completely defined by providing for each actor $t \in T$, the start time of its first execution $s(t,1)$, its period μ_t and its phases scheduling ζ_t. In [7], a periodic schedule is constructed that fixes $\zeta_t(k)$ to $\sum_{l=1}^{k-1} \frac{\ell_t(l)}{\ell_t \cdot \mathbb{1}} \mu_t$. This phases scheduling satisfies the previous conditions as $\zeta_t(k) - \zeta_t(k-1) = \frac{\ell_t(k-1)}{\ell_t \cdot \mathbb{1}} \mu_t \geq \ell_t(k-1)$ if $\mu_t \geq \ell_t \cdot \mathbb{1}$. However, it introduces an additional overestimation of the sufficient buffer sizes as it does not take into account phases transfer rates. The main goal of this paper is to compute an optimized phases scheduling to derive a better estimation of sufficient buffer sizes. Initially, we will assume that the values of $\zeta_t(k)$ are arbitrary fixed so we can generalize results from [7] to any phases scheduling. Later, in Section VI, a Min-Max LP is proposed to fix these values.

A precedence constraint induces a delay between the considered executions in any valid schedule. Furthermore, it has been proved [7] in the case of a periodic schedule that the set of precedence constraints per couple of phases is respected if a delay exists between the start times of the first executions of t and t'.

Lemma 2 ([7]). *Let $a = (t,t') \in A$. The set of precedence constraints from t_k to $t'_{k'}$ is respected if:*

$$s(t',1) - s(t,1) \geq f(k,k') + \frac{\mu_t}{w_a \cdot \mathbb{1}} \min\left\{ \lfloor g(k,k') \rfloor_{gcd_a}, \lfloor g(k,k') \rfloor_{gcd'_a} \right\}$$

with $f(k,k') = \zeta_t(k) + \ell_t(k) - \zeta_{t'}(k')$

$g(k,k') = -M_0(a) - D_a^+ Pred\langle t_k, 1\rangle + D_a^- \langle t'_{k'}, 1\rangle$.

Proof: This is a generalization of the proof in [7] to the case of any valid phases scheduling.

Let us suppose that $a = (t,t')$ induces a precedence constraint from $\langle t_k, n\rangle$ to $\langle t'_{k'}, n'\rangle$ with $k \in \{1, \cdots, \varphi(t)\}$, $k' \in \{1, \cdots, \varphi(t')\}$ and $(n,n') \in (\mathbb{N} - \{0\})^2$. Then, it induces a delay between these two executions,

$$s(t_k, n) + \ell_t(k) \leq s(t'_{k'}, n').$$

Since we consider periodic schedules, this inequality can be written in terms of the start times of the first executions of t and t':

$$s(t',1) - s(t,1) \geq f(k,k') + (n-1)\mu_t - (n'-1)\mu_{t'}$$

with $f(k,k') = \zeta_t(k) + \ell_t(k) - \zeta_{t'}(k')$.

Now, we want to derive a delay that satisfies all the set of precedence constraints from t_k to $t'_{k'}$, *i.e.* for all the couples (n,n') such there exists a precedence constraint from $\langle t_k, n\rangle$ to $\langle t'_{k'}, n'\rangle$.

By definition of D_a^+ and D_a^-,

$$D_a^+\langle t_k, n\rangle - D_a^-\langle t'_{k'}, n'\rangle = D_a^+\langle t_k, 1\rangle + (n-1)w_a \cdot \mathbb{1} - D_a^-\langle t'_{k'}, 1\rangle - (n'-1)v_a \cdot \mathbb{1}.$$

We deduce then

$$n - 1 = \frac{1}{w_a \cdot \mathbb{1}} \left(D_a^+\langle t_k, n\rangle - D_a^-\langle t'_{k'}, n'\rangle + (n'-1)v_a \cdot \mathbb{1} - D_a^+\langle t_k, 1\rangle + D_a^-\langle t'_{k'}, 1\rangle \right).$$

If we replace $n-1$ in the inequality by this value, we get $s(t',1) - s(t,1) \geq f(k,k') - (n'-1)\left(\mu_{t'} - \frac{v_a \cdot \mathbb{1}}{w_a \cdot \mathbb{1}} \mu_t\right) + \frac{\mu_t}{w_a \cdot \mathbb{1}} \left(D_a^+\langle t_k, n\rangle - D_a^-\langle t'_{k'}, n'\rangle - D_a^+\langle t_k, 1\rangle + D_a^-\langle t'_{k'}, 1\rangle \right)$.

As mentioned before, for any infinite valid schedule achieved within bounded memory, throughputs of t and t' must verify

$$\nu_t = \frac{v_a \cdot \mathbb{1}}{w_a \cdot \mathbb{1}} \nu_{t'}.$$

Since, for any periodic schedule that satisfies the thoughput constraint $\mu_t = 1/\nu_t$ and $\mu_{t'} = 1/\nu_{t'}$, then

$$\mu_{t'} = \frac{v_a \cdot \mathbb{1}}{w_a \cdot \mathbb{1}} \mu_t.$$

Hence,

$$\mu_{t'} - \frac{v_a \cdot \mathbb{1}}{w_a \cdot \mathbb{1}} \mu_t = 0.$$

Thus,

$$s(t',1) - s(t,1) \geq f(k,k') + \frac{\mu_t}{w_a \cdot \mathbb{1}} \left(D_a^+\langle t_k, n\rangle - D_a^-\langle t'_{k'}, n'\rangle - D_a^+\langle t_k, 1\rangle + D_a^-\langle t'_{k'}, 1\rangle \right).$$

According to Lemma 1, $D_a^+\langle t_k, n\rangle - D_a^-\langle t'_{k'}, n'\rangle \leq \min\{H_{max}(k), H'_{max}(k,k')\}$ for any couple (n,n') such there exists a precedence constraint from $\langle t_k, n\rangle$ to $\langle t'_{k'}, n'\rangle$. Then, if we replace $D_a^+\langle t_k, n\rangle - D_a^-\langle t'_{k'}, n'\rangle$ by $\min\{H_{max}(k), H'_{max}(k,k')\}$ in the inequality, the obtained delay between $s(t',1)$ and $s(t,1)$ satisfies all the precedence constraints from t_k to $t'_{k'}$:

$$s(t',1) - s(t,1) \geq f(k,k') + \frac{\mu_t}{w_a \cdot \mathbb{1}} \left(\min\{H_{max}(k), H'_{max}(k,k')\} - D_a^+\langle t_k, 1\rangle + D_a^-\langle t'_{k'}, 1\rangle \right).$$

Since $D_a^-\langle t'_{k'}, 1\rangle$ (*resp.* $D_a^+ Pred\langle t_k, 1\rangle$) is a multiple of gcd_a (*resp.* gcd'_a),

$$H_{max}(k) - D_a^+\langle t_k, 1\rangle + D_a^-\langle t'_{k'}, 1\rangle = \lfloor -M_0(a) - D_a^+ Pred\langle t_k, 1\rangle + D_a^-\langle t'_{k'}, 1\rangle \rfloor_{gcd_a}$$

and

$$H'_{max}(k,k') - D_a^+\langle t_k, 1\rangle + D_a^-\langle t'_{k'}, 1\rangle = \lfloor -M_0(a) - D_a^+ Pred\langle t_k, 1\rangle + D_a^-\langle t'_{k'}, 1\rangle \rfloor_{gcd'_a}$$

Hence,

$$s(t',1) - s(t,1) \geq f(k,k') + \frac{\mu_t}{w_a \cdot \mathbb{1}} \min\left\{ \lfloor g(k,k') \rfloor_{gcd_a}, \lfloor g(k,k') \rfloor_{gcd'_a} \right\}$$

with $f(k,k') = \zeta_t(k) + \ell_t(k) - \zeta_{t'}(k')$

$g(k,k') = -M_0(a) - D_a^+ Pred\langle t_k, 1\rangle + D_a^-\langle t'_{k'}, 1\rangle$.

■

Then a minimum delay β_a, that is the maximum of the delays computed per couple of phases, can be derived to ensure that no data is consumed by t' before it is available.

Lemma 3 ([7]). *Let $a = (t,t') \in A$. All the precedence constraints caused by a are fulfilled if:*

$$s(t',1) - s(t,1) \geq \beta_a$$

with

$$\beta_a = \max_{k,k'} \left\{ f(k,k') + \frac{\mu_t}{w_a \cdot \mathbb{1}} \min\left\{ \lfloor g(k,k') \rfloor_{gcd_a}, \lfloor g(k,k') \rfloor_{gcd'_a} \right\} \right\}.$$

Lemma 3 defines a set of potential constraints, one per arc, that prunes all the precedence constraints between the executions of adjacent actors. It is then sufficient that a periodic schedule satisfies this set of potential constraints to be valid.

Theorem 1 ([7]). *A periodic schedule that verifies:*

$$\forall a = (t,t') \in A,\ s(t',1) - s(t,1) \geq \beta_a$$

is feasible.

The next step is to build an Integer Linear Program based on this set of potential constraints to determine the start times of first executions of actors (*i.e.* $s(t,1),\ t \in T$) while minimizing buffer sizes. However, when $M_0(a)$ is unknown, the non-linearity of β_a on $M_0(a)$ prevents such an ILP formulation. The following section constitute the first contribution of this paper. It is dedicated to the linearization of β_a on $M_0(a)$. This will lead, by a restriction on the values taken by $M_0(a)$, to the computation of a sufficient delay that is linear on $M_0(a)$.

IV. Linearization of the Delay β_a

A. Actors Isolation

The first disturbing part in the current formula of the minimum delay β_a is the impossibility to decouple terms depending on the phase t_k from those depending on the phase $t'_{k'}$. It disables any treatment per actor in isolation, particularly the optimization of phases scheduling as we will see in Section VI. It may also increase significantly run-times in the case of large numbers of phases.

The idea to isolate t from t' is as follows. We analyse the function $\lfloor x \rfloor_\gamma$ for different values of γ in order to break the coupling due to $min(\lfloor g(k,k') \rfloor_{gcd_a}, \lfloor g(k,k') \rfloor_{gcd'_a})$ in the formula of β_a. If we can figure out which of $\lfloor g(k,k') \rfloor_{gcd_a}$ or $\lfloor g(k,k') \rfloor_{gcd'_a}$ is more likely to coincide with $min(\lfloor g(k,k') \rfloor_{gcd_a}, \lfloor g(k,k') \rfloor_{gcd'_a})$, then we can decide which of $\beta_a^1 = \max_{k,k'}\{f(k,k') + \frac{\mu_t}{w_a \cdot \mathbb{1}} \lfloor g(k,k') \rfloor_{gcd_a}\}$ or $\beta_a^2 = \max_{k,k'}\{f(k,k') + \frac{\mu_t}{w_a \cdot \mathbb{1}} \lfloor g(k,k') \rfloor_{gcd'_a}\}$ should substitute for β_a.

Both β_a^1 and β_a^2 are greater than or equal to β_a. Then, all the precedence constraints induced by a are still respected if $s(t',1) - s(t,1) \geq \beta_a^1$ or $s(t',1) - s(t,1) \geq \beta_a^2$. However, the choice between these two values to substitute for β_a needs to be the more accurate possible, as the accuracy of computed buffer sizes depends on. Once the substitution done, terms in k and k' can be decoupled. Indeed, as $D_a^-\langle t'_{k'}, 1\rangle$ (*resp.* $D_a^+ Pred\langle t_k, 1\rangle$) is a multiple of gcd_a (*resp.* gcd'_a):

$$\beta_a^1 = \max_k F(k, M_0(a)) + \max_{k'} G(k')$$

with

$F(k, M_0(a)) = \zeta_t(k) + \ell_t(k) + \frac{\mu_t}{w_a \cdot \mathbb{1}} \lfloor -M_0(a) - D_a^+ Pred\langle t_k, 1\rangle \rfloor_{gcd_a}$

$G(k') = \frac{\mu_t}{w_a \cdot \mathbb{1}} D_a^-\langle t'_{k'}, 1\rangle - \zeta_{t'}(k')$

and

$$\beta_a^2 = \max_k F'(k) + \max_{k'} G'(k', M_0(a))$$

with

$F'(k) = \zeta_t(k) + \ell_t(k) - \frac{\mu_t}{w_a \cdot \mathbb{1}} D_a^+ Pred\langle t_k, 1\rangle$

$G'(k', M_0(a)) = -\zeta_{t'}(k') + \frac{\mu_t}{w_a \cdot \mathbb{1}} \lfloor -M_0(a) + D_a^-\langle t'_{k'}, 1\rangle \rfloor_{gcd'_a}$.

Therefore, only $\varphi(t) + \varphi(t')$ steps are needed to compute β_a^1 (*resp.* β_a^2).

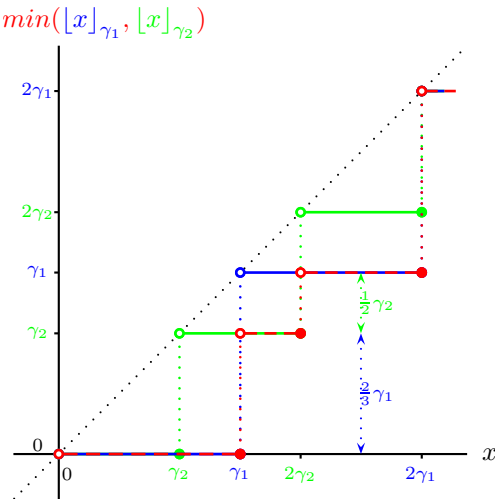

Fig. 2. The case $2\gamma_1 = 3\gamma_2$. The curve $\lfloor x \rfloor_{\gamma_1}$ in blue line , $\lfloor x \rfloor_{\gamma_2}$ in green line and $min(\lfloor x \rfloor_{\gamma_1}, \lfloor x \rfloor_{\gamma_2})$ in dashed red line.

Figure 2 shows three curves: $\lfloor x \rfloor_{\gamma_1}$ in blue, $\lfloor x \rfloor_{\gamma_2}$ with $\gamma_2 = \frac{2}{3}\gamma_1$ in green and $min(\lfloor x \rfloor_{\gamma_1}, \lfloor x \rfloor_{\gamma_2})$ in dashed red line. We can restrict the analysis to the interval $[0, 2\gamma_1[$ as $2\gamma_1$ is the least common multiple for both γ_1 and γ_2. A first look shows that neither $\lfloor x \rfloor_{\gamma_1}$ or $\lfloor x \rfloor_{\gamma_2}$ is dominated (in the sense of minimization) entirely on the whole interval. However, $\lfloor x \rfloor_{\gamma_1}$ coincides with $min(\lfloor x \rfloor_{\gamma_1}, \lfloor x \rfloor_{\gamma_2})$ on more than 83% of the interval while it is only 50% for $\lfloor x \rfloor_{\gamma_2}$. The function $\lfloor x \rfloor_{\gamma_2}$ is strictly dominated by $\lfloor x \rfloor_{\gamma_1}$ on 50% of the interval and the difference $\lfloor x \rfloor_{\gamma_2} - \lfloor x \rfloor_{\gamma_1}$ attains $\frac{2}{3}\gamma_1$. In general, we can prove the following lemma.

Lemma 4. *Let us suppose $\rho \cdot gcd'_a = \sigma \cdot gcd_a$ with $\rho > \sigma$ (i.e. $gcd_a > gcd_{a'}$). Then, $\lfloor x \rfloor_{gcd_a}$ is strictly dominant on $(1 - \frac{\sigma+1}{2\rho}) \cdot 100\%$ of an interval of $\sigma \cdot gcd_a$ and coincides with $\lfloor x \rfloor_{gcd'_a}$ on $\frac{1}{\rho} \cdot 100\%$. Moreover, the difference $\lfloor x \rfloor_{gcd'_a} - \lfloor x \rfloor_{gcd_a}$ attains $\frac{\rho-1}{\rho} gcd_a$.*

Therefore, the more gcd_a is greater than gcd'_a, the more the interval on which $\lfloor x \rfloor_{gcd_a}$ is dominant is wider. Thus, if $gcd_a > gcd'_a$, we are more inclined to substitute $min(\lfloor g(k,k') \rfloor_{gcd_a}, \lfloor g(k,k') \rfloor_{gcd'_a})$ in the formula of β_a with $\lfloor g(k,k') \rfloor_{gcd_a}$. In the following, without loss of generality, we will suppose $gcd_a > gcd'_a$. Then, the sufficient delay considered after this step of isolation is:

$$\beta_a^1 = \max_k F(k, M_0(a)) + \max_{k'} G(k').$$

B. Linearization of the Sufficient Delay β_a^1

Due to the max_k function and $\lfloor\ \rfloor_{gcd_a}$ in the formula of F, β_a^1 is non-linear and discontinuous on $M_0(a)$. To linearize β_a^1 (*i.e.* $max_k F$) on $M_0(a)$, we must first determine the critical phase $k^\star = \arg\max_k F(k, M_0(a))$. However, if $M_0(a)$ is unknown, such in the case of a backward arc or a forward arc of a feedback buffer, it is not always possible to identify a phase $k^\star$ of t such that:

$$\forall M_0(a) \in \mathbb{N}, \forall k,\ F(k, M_0(a)) \leq F(k^\star, M_0(a)).$$

Such a phase may not exist. Indeed, because $\lfloor -M_0(a) - D_a^+ Pred\langle t_k, 1\rangle \rfloor_{gcd_a}$ can have two different values depending on whether $-M_0(a) - D_a^+ Pred\langle t_k, 1\rangle$ is a multiple or not of gcd_a, there may exist two values x_1 and x_2 for $M_0(a)$ for which, there exists a couple of phases (t_{k^1}, t_{k^2}) such that: $\forall k \neq k^1,\ F(k^1, x_1) > F(k, x_1)$ and $\forall k \neq k^2,\ F(k^2, x_2) > F(k, x_2)$.

Of course, there may exist more than two critical phases. In the example of Figure 3, both k^1 and k^2 are critical as they are not dominated (in the sense of maximization) on the entire interval of $M_0(a)$ values. Phase k^1 is dominated by k^3 for some values of $M_0(a)$, however, k^3 is always strictly dominated by k^2 and thus it is not critical.

The concept of useful tokens allows to overcome this issue. In the context of buffer capacities minimization, this concept states that only some values are relevant for $M_0(a)$; some extra tokens have no effect on the value of the delay β_a^1. First, the case of a unique critical phase is studied. The set of useful tokens is identified and β_a^1 is linearized. In case of several critical phases, only one critical phase can be considered for

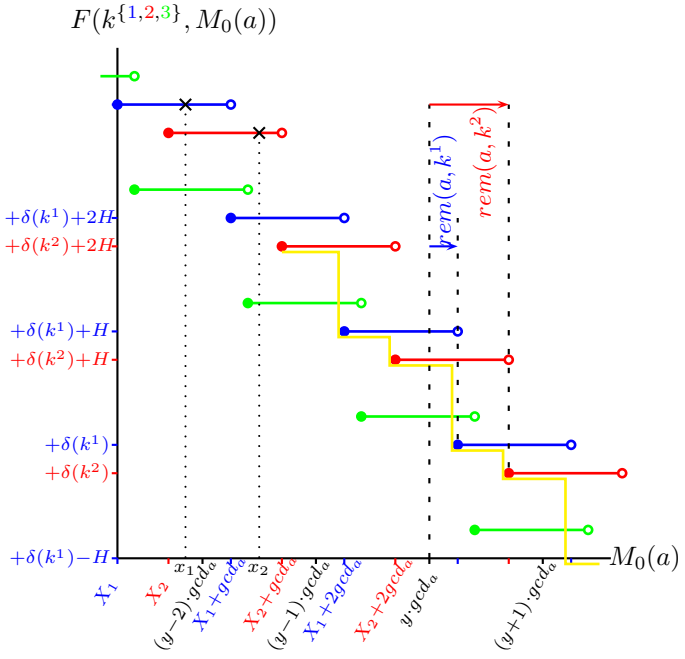

Fig. 3. An example of three phases. Phases k^1 and k^2 are critical. Phase k^3, strictly dominated by k^2, is not critical. In yellow, the staircase made of points of jump discontinuity of critical phases. Its height within an interval of length gcd_a is $H = \frac{\mu_t}{w_a \cdot \mathbb{1}} gcd_a$.

the linearization of β_a^1. The set of useful tokens must then correspond to intervals on which the selected phase is critical.

a) The critical phase is unique: Let us fix arbitrary the value of $M_0(a)$. Then, $F(k, M_0(a))$ can be computed for every phase k and the unique critical phase $k^\star = \arg\max_k F(k, M_0(a))$ can be identified in $\varphi(t)$ steps. Let us define for every phase k, $quot(a,k) = \lfloor -D_a^+ Pred\langle t_k, 1\rangle \rfloor^{gcd_a}$ $0 \leq rem(a,k) = -D_a^+ Pred\langle t_k, 1\rangle - quot(a,k) < gcd_a$, respectively the quotient and the remainder in the euclidean division of $-D_a^+ Pred\langle t_k, 1\rangle$ by gcd_a. Let us also define $m_0(a,k)$ such that

$$m_0(a,k) \cdot gcd_a = -\lfloor -M_0(a) + rem(a,k) \rfloor_{gcd_a} - gcd_a.$$

$m_0(a,k)$ belongs to $\mathbb{N} \cup \{-1\}$ (for $M_0(a) = 0$ and $rem(a,k) \neq 0$, $m_0(a,k) = -1$). By definition of $\lfloor\ \rfloor_{gcd_a}$, we can prove that $m_0(a,k) \cdot gcd_a + rem(a,k) \leq M_0(a)$. For the example of Figure 3, let us ignore the phase k^1, then the unique critical phase is k^2. If we suppose that $M_0(a) = x_2$, then $m_0(a,k^2) \cdot gcd_a + rem(a,k^\star)$ equals X_2, which is the first point of (jump) discontinuity to the left of x_2. Since the function F is right continuous, $F(k^2, x_2) = F(k^2, X_2)$. Then, the value of β_a^1 is still the same if we replace the value of $M_0(a)$ by $m_0(a,k^2) \cdot gcd_a + rem(a,k^2)$. The following lemma formalizes this idea.

Lemma 5. *$M_0(a)$ can be replaced by the less or equal quantity $m_0(a,k^\star) \cdot gcd_a + rem(a,k^\star)$, with $m_0(a,k^\star) \cdot gcd_a = -\lfloor -M_0(a) + rem(a,k^\star) \rfloor_{gcd_a} - gcd_a$, without influencing the value of β_a^1.*

Proof:
Since $k^\star$ is the critical phase, then

$$\max_k F(k, M_0(a)) = F(k^\star, M_0(a)).$$

Now, in the formula of $F(k^\star, M_0(a))$, since $quot(a,k^\star)$ is a multiple of gcd_a, then

$$\lfloor -M_0(a) - D_a^+ Pred\langle t_{k^\star}, 1\rangle \rfloor_{gcd_a} = \lfloor -M_0(a) + rem(a,k^\star) \rfloor_{gcd_a} + quot(a,k^\star).$$

However, the substitution of $M_0(a)$ with the less or equal quantity $m_0(a,k^\star) \cdot gcd_a + rem(a,k^\star)$ in $\lfloor -M_0(a) + rem(a,k^\star) \rfloor_{gcd_a}$ gives the same value:

$$\begin{aligned}\lfloor -(m_0(a,k^\star) \cdot gcd_a + rem(a,k^\star)) + rem(a,k^\star) \rfloor_{gcd_a} &= \left\lfloor \lfloor -M_0(a) + rem(a,k^\star) \rfloor_{gcd_a} + gcd_a \right\rfloor_{gcd_a} \\ &= \lfloor -M_0(a) + rem(a,k^\star) \rfloor_{gcd_a}.\end{aligned}$$

Thus, $F(k^\star, M_0(a)) = F(k^\star, m_0(a,k^\star) \cdot gcd_a + rem(a,k^\star))$. Moreover, for the other phases $k \neq k^\star$, since we supposed that $k^\star$ is the unique critical phase, then

$$F(k, m_0(a,k^\star) \cdot gcd_a + rem(a,k^\star)) < F(k^\star, m_0(a,k^\star) \cdot gcd_a + rem(a,k^\star)).$$

Therefore,

$$\begin{aligned}\max_k F(k, m_0(a,k^\star) \cdot gcd_a + rem(a,k^\star)) &= F(k^\star, m_0(a,k^\star) \cdot gcd_a + rem(a,k^\star)) \\ &= \max_k F(k, M_0(a)).\end{aligned}$$

Then, the substitution does not change the value of β_a^1. ∎

Now, if $M_0(a)$ is unknown, it follows from the previous lemma that only values such that $M_0(a) - rem(a,k^\star)$ is a multiple of gcd_a deserve to be sought for $M_0(a)$ (points of discontinuity). These values define the set of useful tokens. The following lemma provides the linear formula of β_a^1, *i.e.* β_a^1 restricted to the set of useful tokens.

Lemma 6. *The linear formula of the sufficient delay is given by:*

$$\beta_a^1 = \max_k \delta_a(k) + \max_{k'} \psi_a(k') + \frac{\mu_t}{w_a \cdot \mathbb{1}} (-m_0(a) \cdot gcd_a - gcd_a)$$

with $\delta_a(k) = \zeta_t(k) + \ell_t(k) + \frac{\mu_t}{w_a \cdot \mathbb{1}} quot(a,k)$

and $\psi_a(k') = G(k') = -\zeta_{t'}(k') + \frac{\mu_t}{w_a \cdot \mathbb{1}} D_a^- \langle t'_{k'}, 1\rangle.$

Then, the critical phase $k^\star$ is the phase that maximizes $\delta_a(k)$ and the set of useful tokens is defined by

$$M_0(a) \in \{0\} \cup \{m_0(a) \cdot gcd_a + rem(a,k^\star) | m_0(a) \in \mathbb{N}\}.$$

Proof: As a consequence of Lemma 5, for any phase k that may be the unique critical phase, $M_0(a)$ should be of the form $m_0(a) \cdot gcd_a + rem(a,k)$ with $m_0(a)$ a variable $\in \mathbb{N} \cup \{-1\}$ that is independent of any phase. Then,

$$\lfloor -M_0(a) - D_a^+ Pred\langle t_k, 1\rangle \rfloor_{gcd_a} = quot(a,k) - m_0(a) \cdot gcd_a - gcd_a.$$

As a result,

$$\begin{aligned}F(k, M_0(a)) &= \zeta_t(k) + \ell_t(k) + \frac{\mu_t}{w_a \cdot \mathbb{1}} quot(a,k) + \frac{\mu_t}{w_a \cdot \mathbb{1}} (-m_0(a) \cdot gcd_a - gcd_a) \\ &= \delta_a(k) + \frac{\mu_t}{w_a \cdot \mathbb{1}} (-m_0(a) \cdot gcd_a - gcd_a).\end{aligned}$$

Thus the linear formula of β_a^1. The critical phase $k^\star$, that is equal to $\arg\max_k F(k, M_0(a))$, is then equal to $\arg\max_k \delta_a(k)$. ∎

b) There are more than one critical phase: Let us denote by $\mathcal{S}_a$ the set of critical phases. Let us suppose the interval $M_0(a) \in [y \cdot gcd_a, (y+1) \cdot gcd_a[$, $y \in \mathbb{N}$ (*see.* Figure 3). Each phase from $\mathcal{S}_a$ behaves as the unique critical phase on a part of this interval. The points of discontinuity within the interval happen at $m_0(a,k) \cdot gcd_a + rem(a,k)$ with $m_0(a,k) = y, \forall k$. Then,

$$\forall k,\ F(k, y \cdot gcd_a + rem(a,k)) = \delta_a(k) + \frac{\mu_t}{w_a \cdot \mathbb{1}} (-y \cdot gcd_a - gcd_a).$$

Therefore, it is sufficient to compare the values $\delta_a(k)$

to identify the critical phases (for clarity, the values on the vertical axis in Figure 3 were reduced by $\frac{\mu_t}{w_a \cdot \mathbb{1}} (-y \cdot gcd_a - gcd_a)$). Intuitively, the points of discontinuity of the critical phases forms a staircase (*see.* yellow staircase in Figure 3). The other points of discontinuity that are below and/or to the left of that staircase belong to phases that are not critical. The point of discontinuity at the top and to the extreme left of the staircase (within the interval) belongs to the critical phase $k^\star$ such that $\delta_a(k^\star) = \max_k \delta_a(k)$. For the example of Figure 3, $k^\star$ is k^1. It is the critical phase with the smallest remainder. Indeed, any phase k that has a smaller remainder and $\delta_a(k) \leq \delta_a(k^\star)$ is dominated (in the sense of maximization) by $k^\star$ and thus it is not critical. More formally, a critical phase k^i must verify two conditions:
1. $\delta_a(k^\star) - \frac{\mu_t}{w_a \cdot \mathbb{1}} gcd_a < \delta_a(k^i) \leq \delta_a(k^\star)$.
2. $\forall k^j \in \mathcal{S}_a - \{k^i\},\ rem(a,k^i) < rem(a,k^j) \Leftrightarrow \delta_a(k^i) > \delta_a(k^j)$.
The first condition states that a point of discontinuity that is more than $\frac{\mu_t}{w_a \cdot \mathbb{1}} gcd_a$ below the highest point of discontinuity is too below to be on the staircase, thus its phase is not critical. Indeed, the height of the staircase, within any interval of length gcd_a, is $H = \frac{\mu_t}{w_a \cdot \mathbb{1}} gcd_a$. The second condition ensures that all dominated phases are excluded from $\mathcal{S}_a - \{k^\star\}$. This is the case of k^3, it verifies the first condition, however, $rem(a,k^3) < rem(a,k^2)$ and $\delta_a(k^3) < \delta_a(k^2)$. Then, it is dominated by k^2 and does not belong to $\mathcal{S}_a$.

Only one critical phase can be considered for the linearization of β_a^1. The critical phase we consider in the following is $k^\star$ such that $\delta_a(k^\star) = \max_k \delta_a(k)$, which is k^1 for our example. This phase is critical on the interval $[y \cdot gcd_a + \max_{k \in \mathcal{S}_a} rem(a,k), (y+1) \cdot gcd_a[$, *i.e.* starting from the point of discontinuity of the critical phase with the greatest remainder until the end of the interval (for our example, k^2 is the critical phase with the greatest remainder, *i.e.* $k^2 = \arg\max_{k \in \mathcal{S}_a} rem(a,k)$). Then, on this interval

$$\max_k F(k, M_0(a)) = F(k^\star, M_0(a)).$$

Since $F(k^\star, .)$ is right-continuous and constant on $[y \cdot gcd_a + rem(a,k^\star), (y+1) \cdot gcd_a[$, then

$$\begin{aligned} F(k^\star, M_0(a)) &= F(k^\star, y \cdot gcd_a + rem(a,k^\star)) \\ &= \max_k \delta_a(k) + \tfrac{\mu_t}{w_a \cdot \mathbb{1}} (-y \cdot gcd_a - gcd_a)\ . \end{aligned}$$

Therefore,

$$\begin{aligned} &\forall M_0(a) \in [y \cdot gcd_a + \max_{k \in \mathcal{S}_a} rem(a,k), (y+1) \cdot gcd_a[, \\ &\quad max_k F(k, M_0(a)) = \max_k \delta_a(k) + \tfrac{\mu_t}{w_a \cdot \mathbb{1}} (-y \cdot gcd_a - gcd_a)\ . \end{aligned}$$

Then, it is sufficient to limit the number of tokens to the minimum $y \cdot gcd_a + \max_{k \in \mathcal{S}_a} rem(a,k)$, which defines useful tokens for the interval $[y \cdot gcd_a, (y+1) \cdot gcd_a[,\ y \in \mathbb{N}$.

From the previous discussion, the following lemma holds.

Lemma 7. *A linear formula of the sufficient delay is given by:*
$\beta_a^1 = \max_k \delta_a(k) + \max_{k'} \psi_a(k') + \frac{\mu_t}{w_a \cdot \mathbb{1}} (-m_0(a) \cdot gcd_a - gcd_a)$
with
$M_0(a) \in \{0\} \cup \{m_0(a) \cdot gcd_a + \max_{k \in \mathcal{S}_a} rem(a,k) | m_0(a) \in \mathbb{N}\}$.

Remark 1. *In the case $gcd'_a > gcd_a$, the actors isolation step results in the sufficient delay β_a^2. The issue is then on the determination of the critical phase $k'^\star = \arg\max_{k'} G'(k', M_0(a))$. By the same procedure we get to the following results: let us define $quot'(a,k') = \lfloor D_a^- \langle t'_{k'}, 1 \rangle \rfloor^{gcd'_a}$ and $rem'(a,k') = D_a^- \langle t'_{k'}, 1 \rangle - quot'(a,k')$, then*
$\beta_a^2 = \max_k \delta'_a(k) + \max_{k'} \psi'_a(k') + \frac{\mu_t}{w_a \cdot \mathbb{1}} (-m_0(a) \cdot gcd'_a - gcd'_a)$
with $\delta'_a(k) = \zeta_t(k) + \ell_t(k) - \frac{\mu_t}{w_a \cdot \mathbb{1}} D_a^+ Pred\langle t_k, 1 \rangle$
and $\psi'_a(k') = -\zeta_{t'}(k') + \frac{\mu_t}{w_a \cdot \mathbb{1}} quot'(a,k')$.
Then, the selected critical phase $k'^\star$ is the phase that maximizes $\psi'_a(k')$ and the set of useful tokens is defined by
$M_0(a) \in \{0\} \cup \{m_0(a) \cdot gcd'_a + \max_{k' \in \mathcal{S}'_a} rem'(a,k') | m_0(a) \in \mathbb{N}\}$
with $\mathcal{S}'_a$ the set of critical phases.

This concludes the linearization step of the minimum delay β_a. The following corollary is a consequence of Theorem 1 and both actors isolation and sufficient delay linearization steps.

Corollary 1. *A periodic schedule that verifies,*
$\forall a = (t,t') \in A$ *such that* $gcd_a > gcd'_a,\ s(t',1) - s(t,1) \geq \beta_a^1$
$\forall a = (t,t') \in A$ *such that* $gcd'_a > gcd_a,\ s(t',1) - s(t,1) \geq \beta_a^2$
is feasible.

V. Buffer Sizes Minimization

Let us suppose an application modelled by a consistent strongly connected CSDFG $\mathcal{G} = (T, A)$. Each bounded buffer $b(a)$ of this application is modelled by a forward arc a and a backward arc a'. The capacity of $b(a)$ that is defined as the maximum number of data that $b(a)$ can store at a time is then given by $M_0(a) + M_0(a')$. All the storage units of $b(a)$ are of the same size (cost) $\theta(a)$, then the size of $b(a)$ is equal to $\theta(a) \times (M_0(a) + M_0(a'))$.

The throughput constrained buffer sizes minimization problem consists in computing a feasible schedule s such that:
1. The overall buffer sizes $\sum_{a \in A} \theta(a) M_0(a)$ is minimized,
2. The throughput of s, *i.e.* $\nu(s)$, is greater or equal to some predefined throughput.

In [7], using a periodic schedule and a phases scheduling $\zeta_t(k) = \sum_{l=1}^{k-1} \frac{\ell_t(l)}{\ell_t \cdot \mathbb{1}} \mu_t$, this problem is formulated as an Integer Linear Program (ILP). In this section, a generalized version of this ILP that accepts any valid phases scheduling is presented. This is allowed by the linearization step and the useful tokens concept developed in the previous section.

At this stage we consider that actors phases schedulings $\zeta_t,\ t \in T$ have already been fixed. Then, both the set of critical phases and the set of useful tokens per arc can be identified. In the previous section, it has been established that a periodic schedule is feasible if sufficient delays exist between the start times of first executions of adjacent actors (*see.* Corollary 1); the throughput constrained buffer sizes minimization problem defined above can then be formulated by the following Integer Linear Program $\Pi(\mathcal{G})$:

$$\begin{aligned} &min \textstyle\sum_{a \in A} \theta(a) M_0(a) \quad \textit{subject to} \\ &\begin{cases} \forall a = (t,t') \in A \text{ such that } gcd_a \geq gcd'_a, \\ \quad s(t',1) - s(t,1) \geq \max_k \delta_a(k) + \max_{k'} \psi_a(k') \\ \qquad + \frac{\mu_t}{w_a \cdot \mathbb{1}} (-m_0(a) \cdot gcd_a - gcd_a) \\ \quad M_0(a) \geq m_0(a) \cdot gcd_a + \max_{k \in \mathcal{S}_a} rem(a,k). \\ \forall a = (t,t') \in A \text{ such that } gcd'_a > gcd_a, \\ \quad s(t',1) - s(t,1) \geq \max_k \delta'_a(k) + \max_{k'} \psi'_a(k') \\ \qquad + \frac{\mu_t}{w_a \cdot \mathbb{1}} (-m_0(a) \cdot gcd'_a - gcd'_a) \\ \quad M_0(a) \geq m_0(a) \cdot gcd'_a + \max_{k' \in \mathcal{S}'_a} rem'(a,k'). \\ \forall a \in A,\ \ m_0(a) \in \mathbb{N} \cup \{-1\} \\ \quad M_0(a) \geq 0. \\ \forall t \in T, \quad s(t,1) \geq 0. \end{cases} \end{aligned}$$

Solving an ILP has exponential-time worst-case performance. We consider the LP-relaxation of the ILP by ignoring the integrity constraints on $m_0(a),\ a \in A$. A solution that satisfies all the constraints of the ILP is then obtained by rounding the fractional elements of the LP-relaxation solution to their nearest higher integer.

VI. Phases Scheduling Optimization

This section describes the principal contribution of this paper. An optimized phases scheduling per CSDF actor is derived such that the evaluation of sufficient buffer sizes made by the ILP $\Pi(\mathcal{G})$ is more accurate. For an actor t, the goal is to fix $\zeta_t(k), k \in \{1, \cdots, \varphi(t)-1\}$. In practice, the phases scheduling optimization step is applied before the construction of the ILP, however, to be able to motivate the usefulness of this optimization step, the ILP has been presented before. First, to introduce the approach, the case of a graph of two actors and one buffer is studied. It is shown that the producer and the consumer phases schedulings have an impact on the lower bound of the buffer capacity computed by the ILP. A Min-Max Linear Program (LP) is derived to optimize an actor phases scheduling with the aim of reducing this lower bound. Then, the approach is generalized to the case of an actor with several input/output buffers.

Let us consider a CSDFG $\mathcal{G}$ of one buffer $b(a)$ modelled by a forward arc $a = (t, t')$ and a backward arc $a' = (t', t)$. Let us suppose that $gcd_a > gcd'_a$. It follows that $gcd'_{a'} > gcd_{a'}$ since $gcd'_{a'} = gcd_a$ and $gcd_{a'} = gcd'_a$. The throughput constrained buffer size minimization problem is formulated by the ILP $\Pi(\mathcal{G})$:

$$min\ M_0(a) + M_0(a') \quad \textit{subject to}$$
$$\begin{cases} s(t',1) - s(t,1) \geq \max_k \delta_a(k) + \max_{k'} \psi_a(k') \\ \qquad\qquad + \frac{\mu_t}{w_a \cdot \mathbb{1}} \left(-m_0(a) \cdot gcd_a - gcd_a\right) \\ s(t,1) - s(t',1) \geq \max_{k'} \delta'_{a'}(k') + \max_k \psi'_{a'}(k) \\ \qquad\qquad + \frac{\mu_t}{w_a \cdot \mathbb{1}} \left(-m_0(a') \cdot gcd_a - gcd_a\right) \end{cases}$$

(obvious constraints are not represented). The first remark is that the cost (size) per unit of storage θ has been dropped from the objective function. Indeed, both arcs represent the same buffer and thus have the same cost per unit of storage, *i.e.* $\theta(a) = \theta(a')$. The capacity of $b(a)$ given by $M_0(a) + M_0(a')$ verifies

$$M_0(a) + M_0(a') \geq \big(m_0(a) + m_0(a')\big) \cdot gcd_a + \max_{k \in \mathcal{S}_a} rem(a,k) + \max_{k \in \mathcal{S}'_{a'}} rem(a',k)\ .$$

Now, by summing the potential constraints of $\Pi(\mathcal{G})$, a lower bound for $(m_0(a) + m_0(a')) \cdot gcd_a$ can be derived:

$$\big(m_0(a) + m_0(a')\big) \cdot gcd_a \geq \frac{w_a \cdot \mathbb{1}}{\mu_t} (\max_k \delta_a(k) + \max_{k'} \psi_a(k') + \max_{k'} \delta'_{a'}(k') + \max_k \psi'_{a'}(k)) - 2 \cdot gcd_a.$$

It follows that to optimize the capacity of $b(a)$, we have to minimize $\max_k \delta_a(k) + \max_{k'} \psi_a(k') + \max_{k'} \delta_{a'}(k') + \max_k \psi_{a'}(k)$ which is a linear function of ζ_t and $\zeta_{t'}$. Since terms in k and k' can be decoupled, the actor t phases scheduling optimization can be performed independently from that of the adjacent actor t'. This is enabled by the actors isolation step described in Subsection IV-A. Impacts of the remainders $\max_{k \in \mathcal{S}_a} rem(a,k)$ and $\max_{k \in \mathcal{S}'_{a'}} rem(a',k)$ are not taken into account during the optimization. It would increase considerably the complexity and run-times while the impact on the accuracy should be limited.

Let us denote by $\Delta_a = \max_k \delta_a(k)$ and by $\Psi'_{a'} = \max_k \psi'_{a'}(k)$. Phases scheduling optimization problem for the actor t can be formulated by the following Min-Max LP:

$$min\,(\Delta_a + \Psi'_{a'}) \quad \text{subject to}$$
$$\begin{cases} \forall k \in \{1, \cdots, \varphi(t)\}, & & \\ \quad \delta_a(k) = \zeta_t(k) + \ell_t(k) + \frac{\mu_t}{w_a \cdot \mathbb{1}} quot(a,k) & \leq & \Delta_a \\ \forall k \in \{1, \cdots, \varphi(t)\}, & & \\ \quad \psi'_{a'}(k) = -\zeta_t(k) + \frac{\mu_t}{w_a \cdot \mathbb{1}} quot'(a',k) & \leq & \Psi'_{a'} \\ \qquad\qquad \zeta_t(1) & = & 0 \\ \forall k \in \{2, \cdots, \varphi(t)\}, \quad \zeta_t(k) - \zeta_t(k-1) & \geq & \ell_t(k-1) \\ \qquad\qquad \zeta_t(\varphi(t)) + \ell_t(\varphi(t)) & \leq & \mu_t \end{cases}$$

The variables of the LP are $\zeta_t(k) | \forall k \in \{1, \cdots, \varphi(t)\}$, Δ_a and $\Psi'_{a'}$. This Min-Max LP will fix ζ_t while trying to minimize $\Delta_a + \Psi'_{a'} = \max_k \delta_a(k) + \max_k \psi'_{a'}(k)$. As it has the same structure, the Min-Max LP that optimizes the scheduling of t' phases is not presented.

Now we will extend the optimization technique to the case t has more than one input/output buffer. Since the optimization can be done in isolation from the other actors, the generalization is straightforward. Both the objective function and the system of linear constraints of the previous Min-Max LP can be extended to take into account the impact that a phases scheduling may have on the capacities of adjacent buffers. This impact is weighted as the adjacent buffers may have different costs per unit of storage. In the case of one buffer, the cost does not interfere as the two arcs involved model the same buffer. As a consequence, approaches that optimize buffers in isolation [3] cannot, by construction, consider the impact of a phases scheduling on adjacent buffers with different costs . The following Min-Max LP $\Pi(t)$ extends the previous linear program:

$$min \sum_{(a_i = (t,t^i), a'_i = (t^i,t))} \theta(a_i)(\Delta_{a_i} + \Psi'_{a'_i}) \quad \text{subject to}$$
$$\begin{cases} \forall a_i = (t,t^i), \forall k \in \{1, \cdots, \varphi(t)\}, & & \\ \quad \delta_{a_i}(k) = \zeta_t(k) + \ell_t(k) + \frac{\mu_t}{w_{a_i} \cdot \mathbb{1}} quot(a_i,k) & \leq & \Delta_{a_i} \\ \forall a'_i = (t^i,t), \forall k \in \{1, \cdots, \varphi(t)\}, & & \\ \quad \psi'_{a'_i}(k) = -\zeta_t(k) + \frac{\mu_t}{w_{a_i} \cdot \mathbb{1}} quot'(a'_i,k) & \leq & \Psi'_{a'_i} \\ \qquad\qquad \zeta_t(1) & = & 0 \\ \forall k \in \{2, \cdots, \varphi(t_i)\}, \quad \zeta_t(k) - \zeta_t(k-1) & \geq & \ell_t(k-1) \\ \qquad\qquad \zeta_t(\varphi(t)) + \ell_t(\varphi(t)) & \leq & \mu_t \end{cases}$$

As any linear program, Min-Max LPs are solvable in polynomial time. Advanced techniques exist to solve efficiently Min-Max LPs. Indeed, because of the wide range of domain applications, Min-Max LPs have been extensively studied during last decades [18]. In addition to the graph theory, the game theory constitutes one of the most important applications area of Min-Max linear programming. Phases scheduling can be seen as a cooperative game. Phases are then considered players trying to maximize their own payoffs. The payoffs matrix can be obtained from the constraint matrix of the Min-Max LP. When the game converges to an equilibrium an optimal solution for the Min-Max LP is obtained.

VII. Experimental Results

A. The Impact of Phases Scheduling

In this subsection, the impact of phases scheduling on sufficient buffer sizes evaluation is highlighted on an MP3 Playback application. The CSDFG $\mathcal{G}$ of this application presented in [3] is composed of four actors and three buffers. The $MP3$ actor delivers a $48kHz$ audio sample stream and the Sample Rate Converter (SRC) actor converts it to a $44.1kHz$ stream.

The actor APP models an Audio Post-Processing applied to the stream before it is converted to an analog signal by the Digital-Analog Converter (DAC). The actor $MP3$ performs its task in 39 phases during which 1152 samples are produced. The number of samples produced by each phase is given by $w_{MP3} = [0, 0, 18\text{x}32, 0, 18\text{x}32]$. Phases execution times are $\ell_{MP3} = [670, 2700, 18\text{x}40, 2700, 18\text{x}40]\mu s$. The three other actors have only one phase. The maximum throughput of the application is imposed by the DAC actor that runs periodically every $\mu_{DAC} = \ell_{DAC} = \frac{1}{44100}s$. Periods of the other actors can be subsequently derived: $\mu_{APP} = \mu_{DAC} = \frac{1}{44100}s$, $\mu_{SRC} = \mu_{APP} \cdot \frac{441}{1} = 10ms$ and $\mu_{MP3} = \mu_{SRC} \cdot \frac{1152}{480} = 24ms$. The capacity of B_3 equals 2 that is optimal.

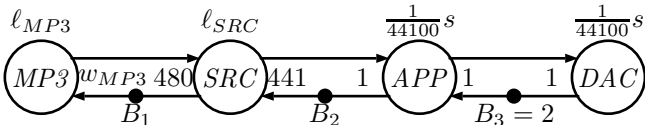

Fig. 4. MP3 Playback application [8].

In the first experiment, three different phases scheduling are applied to the actor $MP3$. The first phases scheduling executes all phases of an occurrence one after the other without interruption, *i.e.* $\zeta_{MP3}(k) = \zeta_{MP3}(k-1) + \ell_{MP3}(k-1), k \in \{2, \cdots, 39\}$. We may call it burst phases scheduling as it responds to a burst transmission criterion. This phases scheduling is not to be confused with an SDF modelling of the $MP3$ actor. The SDF modelling consumes/produces the 1152 tokens in an atomic fashion at the start/end of execution while the burst phases scheduling is still consuming/producing data gradually. See [3] for a comparison between SDF and CSDF modelling of this application. The second phases scheduling fixes $\zeta_{MP3}(k)$ to $\sum_{l=1}^{k-1} \frac{\ell_{MP3}(l)}{\ell_{MP3} \cdot \mathbb{1}} \mu_{MP3}$. We called it Time Averaged phases scheduling. It was initially used in [8] for the computation of sufficient buffer capacities. However, the ILP formulation provided by [7] offers more accurate results than [8] for this phases scheduling. The last phases scheduling is determined by our Min-Max LP applied to the 39 phases of the $MP3$ actor.

Table I lists B_1 capacities obtained for the three different phases scheduling. The comparison is made for different execution times of SRC, $\ell_{SRC} \in \{2.5, 5, 7.5, 10\}ms$. The capacities of buffer B_2 for this experiment are not reported as the same results were obtained by all the techniques and this for the different execution times of SRC. Indeed, SRC and APP are SDF actors and their single phase is scheduled at the start of the execution of the actor.

TABLE I
BUFFER B_1 CAPACITIES.

$\ell_{SRC} =$	$10ms$	$7.5ms$	$5ms$	$2.5ms$
Burst Transmission	1824	1632	1536	1440
Time Averaged [7]	1344	1248	1152	960
Min-Max LP	960	864	768	672

As might be expected, burst phases scheduling needs the biggest buffers. The buffer must bear a high bandwidth during a short time. While the average bandwidth (throughput) is the same for all the phases scheduling $\frac{1152}{24} tokens/ms$, the burst transmission bandwidth is $\frac{1152}{7.51} tokens/ms$ during $7.51ms$. The Min-Max LP delivers the best phases scheduling resulting in better buffer capacities estimation. In comparison with the Time Averaged phases scheduling, results are 40% to 50% smaller.

TABLE II
BUFFER CAPACITIES FOR THE BUFFER B_1.

$\ell_{SRC} =$	$10ms$	$7.5ms$	$5ms$	$2.5ms$
Paper [3]	1056	928	800	672
Min-Max LP	960	864	768	672
[3]+ILP	960	864	768	672

We compared results obtained by the Min-Max LP with those obtained by an alternative approach based on periodic schedules [3] (Table II). This last technique offered the best known evaluation of sufficient buffer capacities for the MP3 Playback application (5% to 28% larger than the optimum). Our technique delivers more accurate results then [3] (up to 10%).

The technique proposed in [3] is also performed in two steps: a phases scheduling optimization step called *Start time postponement* and a minimum delay computation step. We performed an experiment to identify if the accuracy of our technique for this example is due to the phases scheduling optimization or the computation of minimum delays. The experiment consists on using the phases scheduling as described in [3] and our minimum delays computation and ILP formulation. We obtained the same results as those of the Min-Max LP (*see.* Table II). This result is not sufficient to derive a conclusion about the accuracy of the phases scheduling technique proposed in [3], that is discussed in the next subsection. However, it can state that our technique to compute minimum delays, and thus buffer capacities, is more accurate. This can be explained as follows. Let us suppose a couple of adjacent actors t, t' and a buffer $b(a) = (t, t')$. The approach of [3] for the computation of the minimum delay uses linear upper bound on token production times (lub-tpt) and linear lower bound on token consumption times (llb-tct). A minimum delay is derived such that the production time of any token according to the lub-tpt is always less or equal to the consumption time of this token according to the llb-tct which guarantees that no token is consumed before it is produced. However, the smoothing effect introduced by these linear bounds may increase the real value of the minimum delay. In fact, the phase that determines the lub-tpt may not be the phase that produces the tokens that are necessary to activate the execution of the phase that determines the llb-tct. In addition, the determination of the lub-tpt needs $lcm(\varphi(t), \varphi(t'))$ steps which may result in excessive run-times. To overcome problematic run-times, authors proposed a conservative heuristic on the computation of this linear bound that ensures no token is consumed before it is produced; however, it may overestimate the real value of the minimum delays and thus sufficient buffer capacities. On the other hand, our analytical technique for the computation of the minimum delay runs in $\varphi(t) + \varphi(t')$ steps and does not

introduce these overestimations in the computation. Thus, it will always deliver as good results as [3] if not better.

B. The Impact of CSDF Modelling

1) Min-Max LP vs. Stuijk et al.[12]: We used the SDF^3 tool [19] that implements the exact technique based on model checking [12] to compute optimal values. Obtained values are up to 25% better that those of the Min-Max LP. The key of optimality of [12] lies in the self-timed execution used to schedule several occurrences of the repetition vector until a time periodicity is reached. However, as mentioned before, this technique has an exponential worst case complexity (See [7] for an example of a version of the MP3 Application with exponential run-times). On the other hand, periodic schedule semantic imposes a severe restriction on the periodicity of actors executions. We recall that an execution of an actor refers to the execution of all its phases. One of the major contribution of the Min-Max LP lies in its ability to relax the periodic schedule semantic by scheduling phases of multiple consecutive executions of an actor. The second experiment is devoted to highlight this improvement. We used the Min-Max LP to schedule all the actor phases involved in the repetition vector. The repetition vector of the MP3 Playback application is $r = [5, 12, 5292, 5292]$. In terms of executed phases, it is equal to $[39 \cdot 5, 1 \cdot 12, 1 \cdot 5292, 1 \cdot 5292] = [195, 12, 5292, 5292]$. We apply a transformation on the previous graph $\mathcal{G}$ to get a new CSDFG $\mathcal{G}^r$ in which the $MP3^r$ actor has 195 phases. SRC^r, APP^r and DAC^r have respectively 12, 5292 and 5292 identical phases. The vector w_{MP3^r} (*resp.* ℓ_{MP3^r}) is the concatenation of five vectors w_{MP3} (*resp.* ℓ_{MP3}). As a result of this transformation, all the paths in $\mathcal{G}^r$ have a weight of one and all the actors have the same period, *i.e.* $\mu^r_{MP3} = \mu^r_{SRC} = \mu^r_{APP} = \mu^r_{DAC} = \mu_{MP3} \cdot 5 = 120ms$.

Buffer capacities obtained by applying the Time Averaged phases scheduling to $\mathcal{G}^r$ actors are identical to those obtained in the first experiment. In fact, we can prove that this phases scheduling results in the same schedule and thus the same buffer capacities evaluation. Therefore, the Time Averaged phases scheduling cannot relax the periodic schedule semantic. When the phases scheduling obtained by the Min-Max LP is applied, the evaluated capacity of buffer B_1 is up to 60% smaller (*see.* $\ell_{SRC} = 5ms$). Optimal values are obtained except in the case $\ell_{SRC} = 7.5ms$ for which the overestimation drops to less than 3%.

TABLE III
MIN-MAX LP VS. OPTIMAL[12] VS. [3]

$\ell_{SRC} =$	$10ms$	$7.5ms$	$5ms$	$2.5ms$
Opt [12] (B1/B2)	960/882	576/921	480/662	480/552
[3] (B1/B2)	960/882	960/772	960/662	864/552
MM LP (B1/B2)	960/882	768/772	480/662	480/552
MM LP Overesti	0.00%	2.87%	0.00%	0.00%

Experiments show that running time increases linearly with the number of adjacent buffers and quadratically with the number of phases; *i.e.* the time complexity may be in $O(n_t \varphi(t)^2)$ with n_t the number of t adjacent buffers. Table IV gives an overview of the Min-Max LP run times.

TABLE IV
MIN-MAX LP RUN-TIMES (SECONDS).

Phases	$5 \cdot 39$	$50 \cdot 39$	$500 \cdot 39$	$5000 \cdot 39$
Run-time	$1.01 \cdot 10^{-4}$	0.0114	1.16	119

The ILP (relaxation) runs with a speed of $10^{-5}s$ per buffer and thanks to the actors isolation step, several Min-Max LPs can be executed in parallel to reduce the impact on run time of the overall treatment of phases scheduling.

2) Min-Max LP vs. Wiggers et al.[3]: This second CSDF modelling for the MP3 Application reveals an issue with the phases scheduling technique proposed in [3]. Evaluated capacity of B_1 is up to 28.5% bigger than the capacity computed with the previous model (*see.* Table III) and up to 100% bigger than the capacity obtained with the Min-Max LP phases scheduling. This behaviour can be explained as follows. The *Start time postponement* step delays the producer phases start times starting from a burst phases scheduling. For time-complexity considerations, only the scheduling of the phases that produce the multiples of gcd_a tokens are optimized. This is because tokens produced by these phases are potentially tokens that allow to activate the consumer phases. The other phases of the producer are maximally delayed while ensuring that two phases do not overlap. There is no additional mechanism to prevent from delaying too much these phases which may result in too much accumulation of tokens on the backward arc $a' = (t', t)$ since this scheduling policy delays their consumption. The number of phases effectively scheduled is then at most $\sum_{k=1}^{\varphi(t)} w_a(k)/gcd_a$. In practice, for the model proposed in the first subsection $\mathcal{G}$, $\frac{1152}{96} = 12$ phases out of 39 phases of the MP3 actor are actually optimized. Surprisingly, in the case of the second model $\mathcal{G}^r$, the number of optimized phases does not change, $\sum_{k=1}^{\varphi(MP3^r)} w_a(k)/gcd_a = \frac{5760}{480} = 12$, while the total number of phases increases to 195. That low ratio of the number of optimized phases to the total number of phases to schedule explains why the accuracy decreases. The accuracy of the phases scheduling technique proposed in [3] is subject to numerical values of transfer rates and we can easily imagine irreducible patterns of transfer rates for which the ratio is very low.

C. The Impact on Periodic Schedulability

As mentioned in the introduction, cycles are critical structures and their initialization may limit the maximum throughput achieved by an application. When a critical cycle results from feedback buffers, a solution that initializes sufficiently these buffers should allow to attain the prefixed throughput. That is the case of the H.263 Encoder [20], [21], [22]. The specification imposes that this application starts with only one frame in the feedback buffer between the Motion Compensation actor to the Motion Estimation actor (the other buffers are initially empty). While this specification is met when the application is scheduled with a self-timed execution or a periodic schedule together with the Min-Max LP phases

scheduling, two frames are necessary to be able to schedule this application periodically with the phases scheduling proposed in [3]. Derived buffer sizes are also impacted.

TABLE V
H.263 ENCODER BUFFER SIZES [21] (THROUGHPUT=15FPS).

Technique	[3]+ILP	MMLP+ILP	Optimal
Buffer Sizes (KBytes)	51.25	25.75	25.75

In addition to couples of arcs that represent buffers, arcs are added to the CSDFG of an application to model mapping decisions on platforms. In [23], authors provide a detailed CSDFG of a channel equalizer and its mapping on a multi-processor system. This application is used to reduce multipath distortion in an FM signal. Several tasks are mapped to the same processor and thus dependency arcs are added to model the fixed order of access to this shared resource. These arcs add new cycles for which the number of initial tokens is already fixed. While a solution to attain the prefixed throughput could be found for cycles that result from feedback buffers, we may not always be able to modify the initialization of cycles that result from resource considerations as the obtained graph will correspond to another mapping decisions.

The use of periodic schedule policy to solve the buffer sizes minimization problem in polynomial time comes at the cost of underestimation of the maximum throughput achieved. We analysed the impact of phases scheduling on periodic schedualability of the channel equalizer. The low run-time of our technique allows us to perform a dichotomic search over the value of the throughput to determine the maximum achievable throughput by a periodic schedule with different phases schedulings. The maximum throughput of the self-timed execution is estimated by a Maximum Cycle Mean (MCM) analysis applied to the equivalent SRDFG which has pseudo-polynomial complexity. The throughput is given in terms of the frequency of the input/output actors of the application. Results are presented in Table VI.

TABLE VI
CHANNEL EQUALIZER [23] MAXIMUM ACHIEVABLE THROUGHPUT.

Technique	[7]	[3]	MM LP	MCM
Maximum Throughput (KHz)	0	0	95.6	146.6
Buffer Sizes (Words)	-	-	108	108

First, we note that the application is not schedulable periodically with the Time Averaged and the phases scheduling proposed by [3]. As a result, no buffer capacities can be derived and no guarantee can be given on the achievable throughput by this application using these two phases schedulings. Conversely, the Min-Max LP phases scheduling achieves a maximum throughput of 95.6KHz that is 34.7% from the maximum throughput achieved by a self-timed execution. Note also that the same buffer sizes are obtained for both schedules. This experiment shows that the Min-Max LP not only contributes on better buffer sizes estimation, but it also increases periodic schedualability of applications which allows to build a fast tool to evaluate the achievable performance for different set of mapping decisions.

VIII. CONCLUSION

In this paper we presented a technique to optimize a CSDF actor phases scheduling. The aim is to increase the accuracy of periodic schedule based approaches to solve the throughput constrained buffer sizes minimization problem. We introduced the actors isolation step which enabled the formulation of this problem by a Min-Max Linear Program. We exposed the ability of this Min-Max Linear Program to relax periodic schedule semantic and how this relaxation together with our ILP formulation allow to obtain close to optimal buffer sizes while running in polynomial time. We also exposed the impact of the phases scheduling on periodic schedualability of applications.

We believe that, in the growing context of High-Level Synthesis, the increased accuracy together with the low run-time of the technique presented in this paper constitute significant advantages for a use in design exploration.

REFERENCES

[1] E. A. Lee and D. G. Messerschmitt, "Synchronous data flow," *IEEE Proceedings of the IEEE*, vol. 75, no. 9, 1987.

[2] G. Bilsen *et al.*, "Cyclo-static data flow," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 3255–3258, 1995.

[3] M. H. Wiggers *et al.*, "Efficient computation of buffer capacities for cyclo-static dataflow graphs," in *Proceedings of the 44th annual Design Automation Conference*. ACM, 2007, pp. 658–663.

[4] T. M. Parks *et al.*, "A comparison of synchronous and cycle-static dataflow," in *Proceedings of the 29th Asilomar Conference on Signals, Systems and Computers*, 1995, p. 204.

[5] K. Denolf *et al.*, "Exploiting the expressiveness of cyclo-static dataflow to model multimedia implementations," *EURASIP Journal on Advances in Signal Processing*, p. 14 pages, 2007.

[6] M. Fingeroff, *High-Level Synthesis Blue Book*. Xlibris Corporation, 2010.

[7] M. Benazouz *et al.*, "A new method for minimizing buffer sizes for cyclo-static dataflow graphs," in *8th IEEE Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia)*, 2010, pp. 11 –20.

[8] M. H. Wiggers *et al.*, "Efficient computation of buffer capacities for cyclo-static real-time systems with back-pressure," in *Proceedings of the 13th IEEE Real Time and Embedded Technology and Applications Symposium*, 2007, pp. 281–292.

[9] R. Cruz, "A calculus for network delay. i. network elements in isolation," *Information Theory, IEEE Transactions on*, vol. 37, no. 1, pp. 114 –131, Jan. 1991.

[10] O. Marchetti and A. Munier-Kordon, "Complexity results for weighted timed event graphs," *Discrete Optimization*, vol. 7, no. 3, pp. 166 – 180, 2010.

[11] A. Dasdan *et al.*, "Efficient algorithms for optimum cycle mean and optimum cost to time ratio problems," in *Design Automation Conference*, 1999, pp. 37–42.

[12] S. Stuijk *et al.*, "Throughput-buffering trade-off exploration for cyclo-static and synchronous dataflow graphs," *IEEE Trans. Comput.*, vol. 57, no. 10, pp. 1331–1345, 2008.

[13] D. D. Gajski, S. Abdi, A. Gerstlauer, and G. Schirner, *Embedded System Design: Modeling, Synthesis and Verification*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[14] B. Bailey *et al.*, *ESL Design and Verification: A Prescription for Electronic System Level Methodology*. Elsevier, 2007.

[15] E. A. Lee, "Consistency in dataflow graphs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 2, pp. 223–235, April 1991.

[16] E. Teruel *et al.*, "On weighted T-systems," in *Proocedings of the 13th Internationnal Conference on Application and Theory of Petri Nets*, vol. 616. Springer, 1992.

[17] T. H. Cormen, C. E. Leiseerson, R. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 1990.

[18] D. Ding-Zhu and P. Panos M., *Minimax and Applications*. Kluwer Academic Publishers, 1995.
[19] S. Stuijk *et al.*, "SDF^3: SDF For Free," in *Proceedings of 6th Application of Concurrency to System Design Conference*, June 2006, pp. 276–278.
[20] D. Kim, "System-level specification and cosimulation for multimedia embedded systems," Ph.D. dissertation, Seoul National University, 2003.
[21] H. Oh and S. Ha, "Fractional rate dataflow model and efficient code synthesis for multimedia applications," *ACM SIGPLAN NOTICE*, vol. 37, pp. 12–17, 2002. [Online]. Available: peace.snu.ac.kr/publications/data/17/lctes2002.pdf
[22] H. Oh, "Efficient cosynthesis from extended dataflow graphs for multimedia applications," Ph.D. dissertation, Seoul National University, 2003.
[23] A. Moonen *et al.*, "Evaluation of the throughput computed with a dataflow model - a case study." TU Eindhoven, Tech. Rep., March 2007. [Online]. Available: www.es.ele.tue.nl/esreports/esr-2007-01.pdf