A New Method for Minimizing Buffer Sizes for Cyclo-Static Dataflow Graphs.

Mohamed Benazouz, Olivier Marchetti, Alix Munier-Kordon, Thierry Michel

A New Method for Minimizing Buffer Sizes for Cyclo-Static Dataflow Graphs

Mohamed Benazouz, Olivier Marchetti, Alix Munier-Kordon and Pascal Urard †
LIP6, Université Pierre et Marie Curie, Paris (France)
† Central R&D – STMicroelectronics, Crolles (France)
Email: {Mohamed.Benazouz, Olivier.Marchetti, Alix.Munier}@lip6.fr, Pascal.Urard@st.com

***Abstract*—Several optimizations must be considered for the design of streaming applications (e.g. multimedia or network packet processing). These applications can be modelled as a set of processes that communicate using buffers. Cyclo-Static Dataflow graphs, which are an extension of Synchronous Dataflow graphs, allow to consider a large class of industrial applications.**

This paper presents an original methodology to minimize the global surface of the buffers for a Cyclo-Static Dataflow graph under a given throughput constraint. It is proved that, if the processes are periodic, each buffer introduces a linear constraint described analytically. The optimization problem is then modelled by an Integer Linear Program. A polynomial algorithm based on its relaxation provides a quasi-optimal solution for real life problems. The resolution of the optimization problem for a Reed-Solomon Decoder application is then detailed.

***Index Terms*—Buffer minimization, Cyclo-Static Dataflow graph, Periodic schedule, Linear Programing, Streaming applications.**

I. Introduction and related work

Embedded systems are becoming increasingly complex because of the consumers expectations. As example, mobile phones are now supposed to take and display photos, download and play multimedia contents, and naturally allow to hold a telephone conversation. Most of these applications consists in data stream processing and can be splitted into a set of processes performing specific treatments infinitely often, and a set of buffers for data exchanges.

Synchronous Dataflow graphs (in short SDF), introduced by Lee and Messerschmitt in [1] are widely used to model communications between processes. An application is modeled by a directed graph were nodes (*resp.* arcs) correspond to processes (*resp.* buffers). Each process consumes (*resp.* produces) data in its input (*resp.* output) buffer. Moreover, the processes production/consumption rates are fixed at compile time.

Cyclo-Static DataFlow graphs (in short CSDF) were introduced by [2] to model more complex communication scheme between two processes: each execution of a process t is decomposed into $\varphi(t) > 0$ phases, each of them sending/receiving a given number of data. A comparison between SDF and CSDF can be found in [3]. More recently, it is shown in [4] that CSDF can be even considered to model another class of channels.

The aim of the paper is to prove that the methodology developed in [5] to minimize the surface of the buffers for a minimum fixed throughput for a SDF can be extended to handle CSDF graphs. This problem was previously studied by [6] using a model checking approach: an optimal solution is sought, but the combinatorial explosion limits dramatically the size of the instances considered.

Our approach is closely related to the results developed in [7], [8]. The authors proved that, if a periodic schedule of the processes is supposed, the constraints induced by a buffer can be expressed linearly using the starting times of the first execution of the adjacent processes. An algorithm is developed to compute each equation. Linear Programming is then considered to minimize a linear function of these starting times.

In this paper, an original methodology using Linear Programming to minimize the surface of the buffers for a CSDF graph is developed. Buffers are supposed to be homogeneous, *i.e.* each buffer stores data of the same dimension. Their surface depends then linearly on the dimension of the data and the size of the buffer. We show mathematically for a periodic schedule of the phases that a buffer induces a linear inequality between the starting times of the first execution: equations are described analytically and the problem is then modeled using an Integer Linear Program. Its relaxation is solved using simplex algorithm and a quasi-optimum solution is built in polynomial time. A small practical example is then presented. Due to its low time-complexity, this methodology may be extended to solve problems with an important number of processes.

The paper is organized as follows: CSDF graphs and our notations are presented in Section 2. It is proved in Section 3 that each buffer induces a couple of linear inequalities expressed using the starting times of the first execution of the adjacent processes. Section 4 is dedicated to the formulation of the problem using Integer Linear Programming. The modeling of a Reed-Solomon Decoder application using CSDF graph

and its minimum solution are lastly presented in Section 5. Section 6 is our conclusion.

II. Cyclo-Static Dataflow graphs

A Cyclo-Static Dataflow graph (CSDF) is a directed graph where nodes model macro-tasks and arcs correspond to buffers. It is denoted by $\mathcal{G} = (T, A)$ where T (*resp.* A) is the set of nodes (*resp.* arcs).

A. Macro-tasks

Every macro-task $t \in T$ is decomposed into $\varphi(t) \in \mathbb{N} - \{0\}$ phases; for every value $k \in \{1, \cdots, \varphi(t)\}$, the kth phase of t is denoted by t_k and has a fixed duration $\ell_t(k)$. One execution of the macro-task $t \in T$ corresponds to the ordered executions of the phases $t_1, \cdots, t_{\varphi(t)}$ and has a duration $\ell_t \cdot \mathbb{1} = \sum_{k=1}^{\varphi(t)} \ell_t(k)$.

Moreover, every macro-task $t \in T$ is executed several times: for every integer $n \in \mathbb{N} - \{0\}$, $\langle t, n\rangle$ denotes 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. It is also supposed that two phases or two successive executions of a macro-task cannot overlap.

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 $Pred$.

B. Buffers

Every arc $a = (t, t') \in A$ represents a buffer $b(a)$ of unbounded size from the macro-task t to t'. $\forall k \in \{1, \cdots, \varphi(t)\}$, it is supposed that $w_k(a)$ data are produced in $b(a)$ at the end of an execution of t_k. Similarly, $\forall k' \in \{1, \cdots, \varphi(t')\}$, $v_{k'}(a)$ data are read from $b(a)$ before the execution of $t'_{k'}$. We set $w_a \cdot \mathbb{1} = \sum_{k=1}^{\varphi(t)} w_a(k)$ and $v_a \cdot \mathbb{1} = \sum_{k=1}^{\varphi(t')} v_a(k)$.

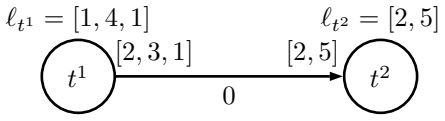

Fig. 1. An unbounded buffer $b(a)$, $a = (t^1, t^2)$. $\varphi(t^1) = 3$ and $\varphi(t^2) = 2$. The arc is labeled by the two vectors $w_a = [2, 3, 1]$, $v_a = [2, 5]$ and by the initial number of data $M_0(a) = 0$.

Figure 1 shows an unbounded buffer $b(a)$ from t^1 to t^2. t^1 (*resp.* t^2) has three (*resp.* two) phases *i.e.*, $\varphi(t^1) = 3$ (*resp.* $\varphi(t^2) = 2$). The arc is labeled by vectors of production/consumption rates, $w_a = [2, 3, 1]$ and $v_a = [2, 5]$. t^1 (*resp.* t^2) is associated with its duration $\ell_{t^1} = [1, 4, 1]$ (*resp.* $\ell_{t^2} = [2, 5]$).

A path of $\mathcal{G}$ of length $p \in \mathbb{N} - \{0\}$ is defined by a list of macro-tasks $\nu = (t^1, t^2, \cdots, t^p)$ such that for any $k \in \{1, \cdots, p-1\}$, $(t^k, t^{k+1}) \in A$. A circuit is a path such that $t^p = t^1$. The weight of a path ν is the ratio

$$W(\nu) = \prod_{a \in \nu} \frac{w_a \cdot \mathbb{1}}{v_a \cdot \mathbb{1}}.$$

Each buffer $b(a)$ has an initial number of data $M_0(a) \in \mathbb{N}$. In the example of Figure 1, the buffer is initially empty, *i.e.* $M_0(a) = 0$.

C. Schedules

A feasible schedule associated with a CSDF graph 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 starting time $s(t_k, n)$ for the nth execution of t_k such that the number of data in every buffer $a \in A$ remains non negative, *i.e.* no data is read before it is produced. The starting times of a macro-task coincide with those of its first phase, *i.e.* $s(t, n) = s(t_1, n)$.

We also consider the existence of a macro-task $t^\star \in T$ for which a throughput of value $\delta^\star$ is required. The throughput of the system for a schedule s is then defined as

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

and must verify $\delta(s) = \delta^\star$. This constraint comes from streaming applications, for which an exact input or/and output throughput is required.

A CSDF graph is said to be consistent if there exists a function M_0 such that a feasible schedule exists. Next Theorem proved in [9], [10], [11] expresses a necessary condition of consistency that is assumed to be true throughout the present paper:

Theorem 1. *If $\mathcal{G}$ is consistent, then, for every circuit c of $\mathcal{G}$, $W(c) \geq 1$.*

Rougthly speaking, for any circuit c of $\mathcal{G}$, $W(c)$ can be viewed as the production rate of data on c. So, if $W(c) < 1$, the whole number of data stored in buffers of c decreases after a finite firing sequence and therefore it leads to a deadlock situation.

D. Precedence constraint

The set of constraints induced by an arc $a = (t, t')$ on executions of macro-tasks t and t' may be expressed as classical precedence constraints. More formally, it is said that a 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 $(n, n') \in (\mathbb{N} - \{0\})^2$ if the two following conditions hold:

1) $\langle t'_{k'}, n'\rangle$ may be executed at the completion of $\langle t_k, n\rangle$;
2) $Pred\langle t'_{k'}, n'\rangle$ may be executed before the end of $\langle t_k, n\rangle$ but not $\langle t'_{k'}, n'\rangle$.

Let us define $D_a^+\langle t_k, n\rangle$ as the total number of data produces by t in the buffer $b(a)$ at the completion of $\langle t_k, n\rangle$. Then, it verifies the sequence

$$D_a^+\langle t_k, n\rangle = D_a^+ Pred\langle t_k, n\rangle + w_a(k)$$

with the initialization $D_a^+\langle t_{\varphi(t)}, 0\rangle = 0$. Similarly, the number of data consumed by t' in the buffer $b(a)$ at the completion of $\langle t'_{k'}, n'\rangle$ is defined by the sequence

$$D_a^-\langle t'_{k'}, n'\rangle = D_a^- Pred\langle t'_{k'}, n'\rangle + v_a(k')$$

with the initialization $D_a^-\langle t'_{\varphi(t')}, 0\rangle = 0$.

Functions D_a^+ and D_a^- may be used to build a precedence constraint between the executions of the macro-tasks. For buffer $b(a)$ in Figure 1, we have $D_a^+\langle t_1^1, 3\rangle = 13$, $D_a^+\langle t_2^1, 3\rangle = 16$, $D_a^-\langle t_1^2, 2\rangle = 9$ and $D_a^-\langle t_2^2, 2\rangle = 14$. Since, $D_a^+\langle t_2^1, 3\rangle \geq D_a^-\langle t_2^2, 2\rangle$, $\langle t_2^2, 2\rangle$ can be executed at the completion of $\langle t_2^1, 3\rangle$. As, $D_a^-\langle t_2^2, 2\rangle > D_a^+\langle t_1^1, 3\rangle \geq D_a^-\langle t_1^2, 2\rangle$, $\langle t_1^2, 2\rangle$ can be executed after $\langle t_1^1, 3\rangle$ but not $\langle t_2^2, 2\rangle$. Thus, there exists a precedence constraint from $\langle t_2^1, 3\rangle$ to $\langle t_2^2, 2\rangle$.

The following lemma provides a mathematical criterion that catches this intuitive definition of a precedence constraint between two executions.

Lemma 1. *Let $a = (t, t') \in A$. There exists 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 $(n, n') \in (\mathbb{N} - \{0\})^2$ iff :*

$$w_a(k) > M_0(a) + D_a^+\langle t_k, n\rangle - D_a^-\langle t'_{k'}, n'\rangle \geq \max\{0, w_a(k) - v_a(k')\}.$$

Proof: According to the definition of a precedence constraint, the first condition implies

$$M_0(a) + D_a^+\langle t_k, n\rangle - D_a^-\langle t'_{k'}, n'\rangle \geq 0.$$

Since $Pred\langle t'_{k'}, n'\rangle$ may be executed before the end of $\langle t_k, n\rangle$,

$$M_0(a) + D_a^+ Pred\langle t_k, n\rangle - D_a^- Pred\langle t'_{k'}, n'\rangle \geq 0$$

and thus

$$M_0(a) + D_a^+\langle t_k, n\rangle - D_a^-\langle t'_{k'}, n'\rangle \geq w_a(k) - v_a(k').$$

Lastly, once $\langle t_k, n\rangle$ and $\langle t'_{k'}, n'\rangle$ are executed, the number of remaining data in a is less than $w_a(k)$, otherwise $\langle t'_{k'}, n'\rangle$ can be executed before the completion of $\langle t_k, n\rangle$. Thus,

$$w_a(k) > M_0(a) + D_a^+\langle t_k, n\rangle - D_a^-\langle t'_{k'}, n'\rangle$$

which concludes the proof. ■

For the example pictured by Figure 1, we get $D_a^+\langle t_2^1, 3\rangle = 16$ and $D_a^-\langle t_2^2, 2\rangle = 14$. Since $w_a(2) = 3$ and $v_a(2) = 5$, the following inequality is true:

$$3 > D_a^+\langle t_2^1, 3\rangle - D_a^-\langle t_2^2, 2\rangle \geq \max\{0, 3 - 5\}.$$

By Lemma 1, there exists a precedence constraint from $\langle t_2^1, 3\rangle$ to $\langle t_2^2, 2\rangle$.

Now, let us note for every arc $a = (t, t') \in A$,

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

where $\gcd$ is the greatest common divisor of a given list of non negative integers. For every integer α, we also set $\lfloor\alpha\rfloor^{gcd_a} = \left\lfloor \frac{\alpha}{gcd_a} \right\rfloor \cdot gcd_a$. The following lemma refines the upper bound from Lemma 1.

Lemma 2. *Let $a = (t, t') \in A$ and the couple of executions $\langle t_k, n\rangle$ and $\langle t'_{k'}, n'\rangle$ with $k \in \{1, \cdots, \varphi(t)\}$, $k' \in \{1, \cdots, \varphi(t')\}$ and $(n, n') \in (\mathbb{N} - \{0\})^2$. We set $H_{max}(k) = -\lfloor M_0(a)\rfloor^{gcd_a} - \lfloor D_a^+ Pred\langle t_k, 1\rangle\rfloor^{gcd_a} - gcd_a + D_a^+\langle t_k, 1\rangle$ and $H_{min}(k, k') = \max\{0, w_a(k) - v_a(k')\} - M_0(a)$.*

Then, there exists a precedence constraint from $\langle t_k, n\rangle$ to $\langle t'_{k'}, n'\rangle$, iff :

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

Proof: By definition of D_a^+, $\forall n \in \mathbb{N} - \{0\}$

$$D_a^+\langle t_k, n\rangle = D_a^+\langle t_{\varphi(t)}, n-1\rangle + D_a^+\langle t_k, 1\rangle$$

and

$$D_a^+\langle t_k, 1\rangle = D_a^+ Pred\langle t_k, 1\rangle + w_a(k).$$

The left inequality of Lemma 1 becomes

$$-(M_0(a) + D_a^+ Pred\langle t_k, 1\rangle) > D_a^+\langle t_{\varphi(t)}, n-1\rangle - D_a^-\langle t'_{k'}, n'\rangle.$$

Since $D_a^+\langle t_{\varphi(t)}, n-1\rangle - D_a^-\langle t'_{k'}, n'\rangle$ is divisible by gcd_a, the strict inequality may be replaced by:

$$-(\lfloor M_0(a)\rfloor^{gcd_a} + \lfloor D_a^+ Pred\langle t_k, 1\rangle\rfloor^{gcd_a} + gcd_a) \geq D_a^+\langle t_{\varphi(t)}, n-1\rangle - D_a^-\langle t'_{k'}, n'\rangle.$$

The left inequality of the lemma is obtained by adding $D_a^+\langle t_k, 1\rangle$. The right part of the inequality is a consequence of Lemma 1, which concludes the proof. ■

E. Bounded buffers

In a CSDF graph, an arc a is associated with a buffer $b(a)$ with a non-limited size, *i.e.* the number of data stored simultaneously in $b(a)$ may be infinite. However, this hypothesis is unacceptable for real-life systems. Stuijk *et al.* [6] noticed that a buffer $b(a)$ with a bounded size from t to t' may be modeled by adding a reverse arc $a' = (t', t)$ in the associated CSDF graph with, for every $k \in \{1, \cdots, \varphi(t)\}$, $v_{a'}(k) = w_{a'}(k)$ and for every $k' \in \{1, \cdots, \varphi(t')\}$, $w_{k'}(a') = v_{k'}(a)$ (see Figure 2). The size of the buffer $b(a)$ is then equal to the sum $M_0(a) + M_0(a')$.

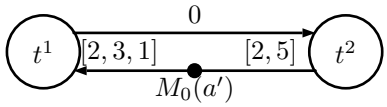

Fig. 2. A bounded buffer $b(a)$.

Without loss of generality, it is assumed that the application is modelled using a connected CSDF graph. Now, if all buffers have a bounded size, the graph obtained by adding reverse arcs is strongly connected (*i.e.* for every couple of macro-tasks $(t', t') \in T^2$, there exists a path from t to t') and is said symmetrical.

The following theorem holds for symmetrical graphs:

Theorem 2. *If $\mathcal{G}$ is consistent and symmetrical, then, for every circuit c of $\mathcal{G}$, $W(c) = 1$.*

Proof: Let us consider a circuit c of $\mathcal{G}$. Since $\mathcal{G}$ is symmetrical, the path c' constructed using reverse arcs of c is also a circuit. By Theorem 1, since $\mathcal{G}$ is consistent, we get $W(c) \geq 1$. Now, if $W(c) > 1$ then $W(c') = \frac{1}{W(c)} < 1$ and thus $\mathcal{G}$ is inconsistent. ∎

The following corollary is used to derive the minimum throughput of macro-tasks.

Corollary 1. *Let $(t, t') \in T^2$. If $\mathcal{G}$ is consistent and symmetrical, then, all paths between t and t' have the same weight.*

Proof: Let us suppose that there exists two disjoint paths, ν_1 and ν_2, from t to t' with two different weights $W(\nu_1) \neq W(\nu_2)$. Since the graph is symmetrical, we may construct a circuit c by concatenating the path ν_1 with the reverse path of ν_2. So, $W(c) = \frac{W(\nu_1)}{W(\nu_2)}$. By Theorem 2, $\mathcal{G}$ is consistent implies $W(c) = 1$ which leads to a contradiction because this involves $W(\nu_1) = W(\nu_2)$. ∎

In the following, it is supposed that the graph is symmetrical.

III. Periodic schedules

Our study is limited to periodic schedules as defined by Wiggers *et al.*[8]. An execution of a macro-task t is scheduled periodically every μ_t time units. Starting times of phases $t_1, \cdots, t_{\varphi(t)}$ are spread over μ_t using their time execution.

More formally, s is a periodic schedule if every task $t \in T$ is associated with a period μ_t 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) $\forall k \in \{2, \cdots, \varphi(t)\}, s(t_k, n) = s(t_{k-1}, n) + \dfrac{\ell_t(k-1)}{\ell_t \cdot \mathbb{1}}\mu_t$.

This definition ensures that two successive phases do not overlap. Note that the throughput of a periodic schedule is exactly $\dfrac{1}{\mu_{t^\star}}$.

A. A sufficient condition of existence for a periodic schedule

The following theorem characterizes a periodic schedule such that all precedence constraints as defined in Section II-D are fulfilled. Every arc $a = (t, t')$ induces a relationship between the couple of periods $(\mu_t, \mu_{t'})$. A minimum delay between the first phases starting times $s(t', 1) - s(t, 1)$ such that no data is consumed by t' before it is produced by t is also expressed. Our equations are similar to [8]. However, our values β_a may be smaller and are evaluated analytically on a smallest set of relevant values. Our values are minimum and computed faster, leading to a better computation of buffers sizes.

Theorem 3. *There exists a set of rationals $\{\beta_a, a \in A\}$ such that, every periodic schedule which verifies:*

1) *$\mu_{t^\star} = \frac{1}{\delta^\star}$ and $\forall t \in T - \{t^\star\}$, $\mu_t = \mu_{t^\star}.W(\nu_{tt^\star})$ where $\nu_{tt^\star}$ is a path of $\mathcal{G}$ from t to $t^\star$.*
2) *$\forall a = (t, t') \in A$,*

$$s(t', 1) - s(t, 1) \geq -\frac{\mu_t}{w_a \cdot \mathbb{1}}\lfloor M_0(a) \rfloor^{gcd_a} + \beta_a$$

is feasible.

Proof: 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 $(n, n') \in (\mathbb{N} - \{0\})^2$. By definition,

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

Since s is periodic, this equation becomes

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

with

$$f(k, k') = \frac{\sum_{l=1}^{k-1} \ell_t(l)}{\ell_t \cdot \mathbb{1}}\mu_t + \ell_t(k) - \frac{\sum_{l=1}^{k'-1} \ell_{t'}(l)}{\ell_{t'} \cdot \mathbb{1}}\mu_{t'}.$$

Let us define now

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

By definition of D_a^+ and D_a^-

$$H(\langle k, n\rangle, \langle k', n'\rangle) = (n-1)w_a \cdot \mathbb{1} - (n'-1)v_a \cdot \mathbb{1} + g(k, k')$$

with $g(k, k') = D_a^+\langle t_k, 1\rangle - D_a^-\langle t'_{k'}, 1\rangle$. It is deduced that

$$n - 1 = \frac{1}{w_a \cdot \mathbb{1}}\left(H(\langle k, n\rangle, \langle k', n'\rangle) + (n'-1)v_a \cdot \mathbb{1} - g(k, k')\right)$$

and thus,

$$s(t', 1) - s(t, 1) \geq f(k, k') - (n'-1)\left(\mu_{t'} - \frac{\mu_t}{w_a \cdot \mathbb{1}}v_a \cdot \mathbb{1}\right) + \frac{\mu_t}{w_a \cdot \mathbb{1}}\left(H(\langle k, n\rangle, \langle k', n'\rangle) - g(k, k')\right).$$

According to Lemma 2, for any couple (n, n') and for every k', $H(\langle k, n\rangle, \langle k', n'\rangle) \leq H_{max}(k)$, and then the right part of the previous inequality is less or equal to

$$r(k, k', n') = f(k, k') - (n'-1)\left(\mu_{t'} - \frac{\mu_t}{w_a \cdot \mathbb{1}}v_a \cdot \mathbb{1}\right) + \frac{\mu_t}{w_a \cdot \mathbb{1}}\left(H_{max}(k) - g(k, k')\right).$$

Thus, to preserve the minimum delay between $s(t', 1)$ and $s(t, 1)$, it is sufficient to consider the delay $r(k, k', n')$. Then, the new considered inequality is

$$s(t', 1) - s(t, 1) \geq r(k, k', n').$$

This inequality must be true for any value n', so $\mu_{t'} - \frac{\mu_t}{w_a \cdot \mathbb{1}}v_a \cdot \mathbb{1} \geq 0$ and then $\frac{\mu_{t'}}{v_a \cdot \mathbb{1}} \geq \frac{\mu_t}{w_a \cdot \mathbb{1}}$. Since the graph is symmetrical, there exists a circuit c that includes a. By

Theorem 2, since $\mathcal{G}$ is consistent, $W(c) = 1$, and then $\frac{\mu_{t'}}{v_a \cdot \mathbb{1}} = \frac{\mu_t}{w_a \cdot \mathbb{1}}$. Thus $\mu_t = \frac{w_a \cdot \mathbb{1}}{v_a \cdot \mathbb{1}} \mu_{t'}$. So,

$$\forall a = (t, t'), \mu_t = \frac{w_a \cdot \mathbb{1}}{v_a \cdot \mathbb{1}} \mu_{t'},$$

and then, for a path $\nu_{tt^\star}$, we get

$$\mu_t = W(\nu_{tt^\star}) \mu_{t^\star}.$$

Now, by Corollary 1, all paths from t to $t^\star$ have the same weight $W(\nu_{tt^\star})$, thus the previous equality always holds. So the first part of the theorrem is proved.

Now, by replacing $r(k, k', n')$ and $H_{max}(k)$ by their values in the precedent inequality we get

$$s(t',1) - s(t,1) \geq -\frac{\mu_t}{w_a \cdot \mathbb{1}} \lfloor M_0(a) \rfloor^{gcd_a} + f(k,k') + \frac{\mu_t}{w_a \cdot \mathbb{1}} \left(-\left(\left\lfloor D_a^+ \langle t_{k-1}, 1 \rangle \right\rfloor^{gcd_a} - gcd_a + D_a^+ \langle t_k, 1 \rangle - g(k,k') \right) \right)$$

and thus

$$s(t',1) - s(t,1) \geq -\frac{\mu_t}{w_a \cdot \mathbb{1}} \lfloor M_0(a) \rfloor^{gcd_a} + f(k,k') + \frac{\mu_t}{w_a \cdot \mathbb{1}} \left(-\left\lfloor D_a^+ \langle t_{k-1}, 1 \rangle \right\rfloor^{gcd_a} + D_a^- \langle t'_{k'}, 1 \rangle - gcd_a \right).$$

This inequality must be true $\forall k \in \{1, \cdots, \varphi(t)\}$ and $\forall k' \in \{1, \cdots, \varphi(t')\}$. Thus, it must be true for the right term equal to $-\frac{\mu_t}{w_a \cdot \mathbb{1}} \lfloor M_0(a) \rfloor^{gcd_a} + \beta_a$ with

$$\beta_a = \max_{k,k'} \left\{ \frac{\sum_{l=1}^{k-1} \ell_t(l)}{\ell_t \cdot \mathbb{1}} \mu_t + \ell_t(k) - \frac{\sum_{l=1}^{k'-1} \ell_{t'}(l)}{\ell_{t'} \cdot \mathbb{1}} \mu_{t'} + \frac{\mu_t}{w_a \cdot \mathbb{1}} \left(-\left\lfloor D_a^+ \langle t_{k-1}, 1 \rangle \right\rfloor^{gcd_a} + D_a^- \langle t'_{k'}, 1 \rangle - gcd_a \right) \right\}.$$

■

B. Computation of the sufficient condition

In this subsection, we evaluate the time complexity of the computation of periods and equations as defined in the previous subsection for a given graph $\mathcal{G} = (A, T)$.

Computation of β_a: Let $a = (t, t')$ an arc of $\mathcal{G}$ and let β_a be the value as defined in the proof of Theorem 3. We observe that terms depending respectively of k and k' can be splitted to obtain:

$$\beta_a = \max_k \left\{ \frac{\sum_{l=1}^{k-1} \ell_t(l)}{\ell_t \cdot \mathbb{1}} \mu_t + \ell_t(k) - \frac{\mu_t}{w_a \cdot \mathbb{1}} \left\lfloor D_a^+ \langle t_{k-1}, 1 \rangle \right\rfloor^{gcd_a} \right\} + \max_{k'} \left\{ -\frac{\sum_{l=1}^{k'-1} \ell_{t'}(l)}{\ell_{t'} \cdot \mathbb{1}} \mu_{t'} + \frac{\mu_t}{w_a \cdot \mathbb{1}} \left(D_a^- \langle t'_{k'}, 1 \rangle - gcd_a \right) \right\}$$

and thus, only $\varphi(t) + \varphi(t')$ steps are needed to evaluate β_a.

Computation of $\mu_t, t \in T$: As seen in the proof of Theorem 3, periods of two adjacent macro-tasks t and t' with $a = (t, t') \in A$ verify $\mu_t = \frac{w_a \cdot \mathbb{1}}{v_a \cdot \mathbb{1}} \mu_{t'}$. Starting from $t^\star$ for which the period $\mu_{t^\star}$ is set, the period of every macro-task t can be computed by a Depth-First Search algorithm [12]. Let us consider $\deg(t)$, the degree of $t \in T$. The computation of periods may take $O(\sum_{t \in T} \deg(t))$ time units which is linear in the size of the graph.

IV. Problem formulation and resolution

Let us suppose a symmetrical CSDF graph $\mathcal{G} = (T, A)$ in which every buffer $b(a)$ is modeled using a couple of arcs (a, a'). The size of $b(a)$ equals to $M_0(a) + M_0(a')$ as seen in Section II-E. This size can be set or not by the designer. In the first case, values $M_0(a)$ and $M_0(a')$ can be unknown. A buffer $b(a)$ is initialized if its initial number of data $M_0(a)$ is set.

A consequence of Theorem 3 is that, for any arc $a \in A$, $M_0(a)$ is a multiple of gcd_a. Thus, if $M_0(a)$ is set, it can be replaced by $\lfloor M_0(a) \rfloor^{gcd_a}$. Otherwise, only multiples of gcd_a are sought for $M_0(a)$. Let us denote by A_1 (*resp.* A_2) the set of arcs for which the value of M_0 is known (*resp.* unknown).

Moreover, buffers are homogeneous, *i.e.* data stored in any buffer $b(a)$ have all the same dimension $\theta(a)$.

For a given throughput, our problem may be formulated by the following Integer Linear Program:

$$min \left(\sum_{a \in A} \theta(a) M_0(a) \right) \quad \textit{subject to}$$

$$\begin{cases} \forall a = (t,t') \in A_1, \\ \qquad s(t',1) - s(t,1) \geq \beta_a - \frac{\mu_t}{w_a \cdot \mathbb{1}} \lfloor M_0(a) \rfloor^{gcd_a} \\ \forall a = (t,t') \in A_2, \\ \qquad s(t',1) - s(t,1) \geq \beta_a - \frac{\mu_t}{w_a \cdot \mathbb{1}} M_0(a) \\ \forall a = (t,t') \in A_2, M_0(a) = m_0(a) \cdot gcd_a \\ \forall a = (t,t') \in A_2, m_0(a) \in \mathbb{N} \\ \forall t \in T, \qquad s(t,1) \geq 0 \end{cases}$$

The first (*resp.* second) inequality expresses the sufficient condition associated with an initialized (*resp.* uninitialized) arc $a \in A_1$ (*resp.* $a \in A_2$) following Theorem 3. The other constraints restrict the values that $M_0(a)$, $a \in A_2$, can take to multiples of gcd_a.

This problem is a generalization of an NP-Hard problem [13]. In order to compute a good solution efficiently, we first solve the linear program relaxation by removing the integrity constraints on the values of $M_0(a)$, $a \in A_2$. Then, to get a feasible solution, for every arc $a \in A_2$, we round $M_0(a)$ to the next greater multiple of gcd_a.

V. Experimental Results

Our method was tested on two particular industrial applications. The first one concerns a Reed Solomon Decoder application (*RSD*). The second one is an MP3 Playback model extracted from [8].

A. RSD application

It is used to detect and correct errors that may occur during wireless communications. The input of our application is a frame of 896 bytes, composed of 864 data bytes and 32 parity bytes. The output is 4 frames of 216 bytes called codewords. Each codeword is associated with the number of errors that were detected in it and whether they were all corrected or not. This determines the status of the frame received (accepted or rejected).

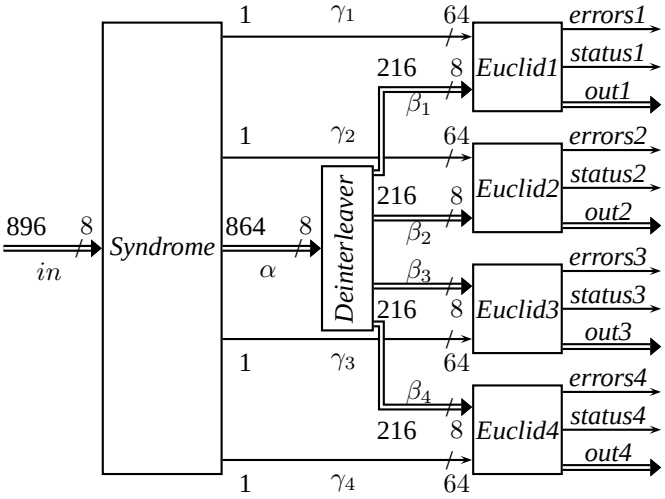

Fig. 3. Block diagram of a Reed-Solomon Decoder. Each arc a represents a buffer and is labeled by the dimension $\theta(a)$ expressed in bits and the number of data transferred during the treatment of one frame.

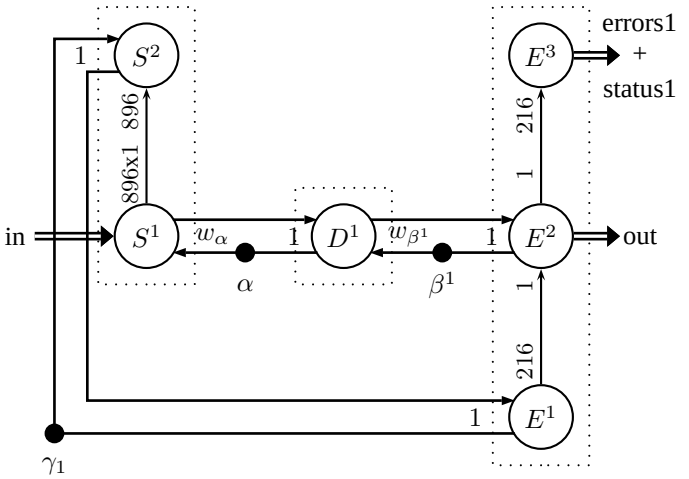

Fig. 4. The CSDF graph modelling the 3 blocks (Syndrome, Deinterleaver and Euclid) and the channels between these blocks. $w_\alpha = [864\text{x}1, 32\text{x}0]$ and $w_{\beta_1} = [1, 0, 0, 0]$.

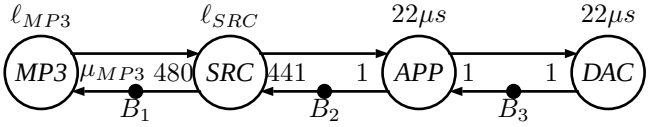

Fig. 5. MP3 Playback, $\mu_{MP3} = [0, 0, 18\text{x}32, 0, 18\text{x}32]$. Durations: $\ell_{MP3} = [670, 2700, 18\text{x}40, 2700, 18\text{x}40]\mu s$, $\ell_{SRC} \in \{2.5, 5, 7.5, 10\}ms$.

A block diagram of this application is shown in Figure 3.

To decode Reed-Solomon codes, an Euclidean decoding algorithm is used and it is implemented by an *Euclid* block. The *Syndrome* block performs the syndrome calculation using 32 parity bytes.

To enhance the throughput, four *Euclid* blocks are used in parallel to decode an interlaced data frame. The *Deinterleaver* block is used to deinterlace a frame of 864 bytes (buffer α) into 4 codewords of 216 bytes (buffers β^1 to β^4). Each codeword is treated by a separate *Euclid* block able to detect 7 errors and to correct at most 3. To perform its task, an *Euclid* block needs a syndrome of 8x8 bits which is delivered by the *Syndrome* block (buffers γ_1 to γ_4).

The period of the system is needed to be 1152 cycle time. The duration of one cycle depends on the technology used.

Figure 4 shows a cyclo static modelling of this application. Because of symmetry only the first *Euclid* block is represented. Also, due to space constraints, several macro-tasks have been merged.

The *Syndrome* block is composed of two macro-tasks S^1 and S^2. S^1 reads a frame of 864 data bytes and 32 parity bytes. It writes data bytes one by one (cycle by cycle) on the buffer α. Once all parity bytes are read, S^2 computes the syndrome and writes it on buffer γ_1, which takes 7 cycles time. Then, $\varphi(S^1) = 896, w_\alpha = [864\text{x}1, 32\text{x}0]$ and $\forall k \in \{1, \cdots, 896\},\ \ell_{S^1}(k) = 1$. $\varphi(S^2) = 1$ and $\ell_{S^2}(1) = 7$.

The unique macro-task of the *Deinterleaver* block has 4 phases (as many as the number of *Euclid* blocks). Every phase takes 1 cycle time *i.e.* , $\forall k \in \{1, \cdots, 4\},\ \ell_{D^1}(k) = 1$. During the phase i, it writes the data just read from buffer α on the buffer β^i and nothing on the three others.

The *Euclid* block is composed of three macro-tasks. They all have one phase *i.e.* , $\varphi(E^1) = \varphi(E^2) = \varphi(E^3) = 1$. $\ell_{E^1}(1) = 89$, $\ell_{E^2}(1) = 1$ and $\ell_{E^3}(1) = 3$.

Our algorithm runs on a $2.3Ghz$ AMD processor and Linux based system. The solver used to resolve the linear program relaxation is *GLPK* [14].

The algorithm computes several solutions of the same minimum cost 1001 bytes. Two of them are shown in Table I. The first solution sets the size of buffer α to its minimum which is 33 bytes, and sizes of buffers $\beta_i, i \in \{1, \cdots, 4\}$ are set to 234 bytes. The second solution does the opposite and reduces the amount of buffers between the *Deinterleaver* block and *Euclid* blocks to its minimum 4x1 bytes. In this case, the minimum size of α is 965 bytes. Adopting a solution rather than the other depends on architectural choices. Our designers preferred merging memory so they opted for the second solution.

TABLE I
BUFFER SIZES FOR THE *RSD* APPLICATION.
$Cost(bytes) = \alpha + (\beta_1 + \beta_2 + \beta_3 + \beta_4) + 8(\gamma_1 + \gamma_2 + \gamma_3 + \gamma_4)$.

Solution	α	$\beta_i,\ i \in \{1 \cdots 4\}$	$\gamma_i,\ i \in \{1 \cdots 4\}$	*Cost*
1	33	234	1	1001
2	965	1	1	1001

B. MP3 Playback application

The *MP3* playback model presented in [8] allows us to compare our results with [6], [8].

We obtain the same buffer sizes as in [8] for different execution time of the converter SRC (See Table II). This artefact is due to the application structure (a chain). Their

objective function minimizes the sum of the first starting times of macro-tasks which does not coincide with the sum of buffer sizes for general graphs.

However our algorithm is faster ($10^{-5}s$ vs $10^{-2}s$) due to the analytical technique presented here for the computation of β_a. This speed factor is to be appreciated in large systems with hundreds or thousands of processes and also when several iterations have to be performed, as in Parameterized CSDF [15].

TABLE II
BUFFER SIZES FOR THE MP3 PLAYERBACK.

$\ell(SRC) =$	10	7.5	5	2.5
B_1	1376	1280	1152	1024
B_2	882	772	662	552
B_3	2	2	2	2
Sum	2260	2054	1816	1578

Model checking approach computes optimal buffer sizes using earliest schedule. Numerical values obtained by [6] are about 23% to 59% better. The flaw of this last technique is its scalability. An improved *MP3* playback is presented to highlight this point: the application presented in Figure 6 is able to read and mix two independent *MP3* input streams. Sizes of buffers B_4 and B_5 are obviously the same as buffers B_1 and B_2. Model checking technique failed to solve this new instance. This proves that exact techniques based on state-space exploration can not deal with this combinatorial optimization problem even for small applications.

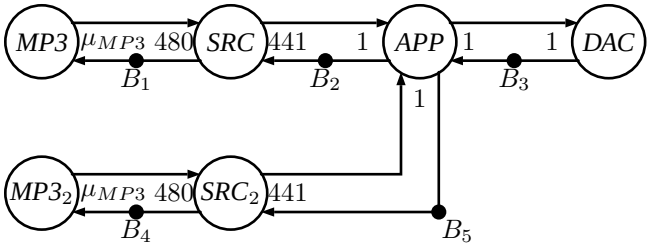

Fig. 6. The mix of two streams. $MP3 \equiv MP3_2$ and $SRC \equiv SRC_2$.

VI. CONCLUSION

It is proved in this paper that our methodology developed in [5] may be extended to CSDF graphs. Moreover, a small real-life problem and its resolution are presented, showing that several minimum solutions can be computed. All our results are mathematically proved, and general heterogeneous buffers may be considered.

However, we are convinced that our results may be improved if the starting times of the phases are not supposed to be periodic. If an analytical expression of the equation is possible in this case, the experimental results should be greatly improved.

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, M. Engels, R. Lauwereins, and J. Peperstraete, "Cyclo-static data flow," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 3255–3258, 1995.

[3] T. M. Parks, J. L. Pino, and E. A. Lee, "A comparison of synchronous and cycle-static dataflow," in *ASILOMAR '95: Proceedings of the 29th Asilomar Conference on Signals, Systems and Computers (2-Volume Set)*. Washington, DC, USA: IEEE Computer Society, 1995, p. 204.

[4] K. Denolf, M. Bekooij, J. Cockx, D. Verkest, and H. Corporaal, "Exploiting the expressiveness of cyclo-static dataflow to model multimedia implementations," *EURASIP Journal on Advances in Signal Processing*, no. Article ID 84078, p. 14 pages, 2007. [Online]. Available: http://dx.doi.org/10.1155/2007/84078

[5] M. Benazouz, O. Marchetti, A. Munier-Kordon, and U. Pascal, "A new approach for minimizing buffer capacities with throughput constraint for embedded system design," *AICCSA '10, International Conference on Computer Systems and Applications, http://hal.archives-ouvertes.fr/hal-00368648/fr/*, 2010.

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

[7] M. H. Wiggers, M. J. G. Bekooij, and G. J. M. Smit, "Efficient computation of buffer capacities for cyclo-static dataflow graphs," in *DAC '07: Proceedings of the 44th annual Design Automation Conference*. New York, NY, USA: ACM, 2007, pp. 658–663.

[8] M. H. Wiggers, M. J. G. Bekooij, P. G. Jansen, and G. J. M. Smit, "Efficient computation of buffer capacities for cyclo-static real-time systems with back-pressure," in *RTAS '07: Proceedings of the 13th IEEE Real Time and Embedded Technology and Applications Symposium*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 281–292.

[9] R. M. Karp and R. E. Miller, "Properties of a model for parallel computations: Determinacy, termination, queueing," *SIAM*, vol. 14, no. 63, pp. 1390 – 1411, 1966.

[10] A. Munier, "Régime asymptotique optimal d'un graphe d'événements temporisé généralisé: application à un problème d'assemblage," *RAIRO-Automatique Productique Informatique Industrielle*, vol. 27, no. 5, pp. 487–513, 1993.

[11] E. Teruel, P. Chrzastowski-Wachtel, J. M. Colom, and M. Silva, "On weighted T-systems," in *Proocedings of the 13th Internationnal Conference on Application and Theory of Petri Nets 1992, Lecture Notes in Computer Science*, vol. 616. Springer, 1992.

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

[13] O. Marchetti, "Dimensionnement des mémoires pour systèmes embarqués," Ph.D. dissertation, Université Pierre et Marie Curie, 2006.

[14] "GLPK – GNU Linear Programming Kit, http://www.gnu.org/software/glpk/."

[15] S. Sriram and S. S. Bhattacharyya, *Embedded Multiprocessors: Scheduling and Synchronization*. New York, NY, USA: Marcel Dekker, Inc., 2000.