Single-Rate Approximations of Cyclo-Static Synchronous Dataflow Graphs

Robert de Groote, Philip K.F. Hölzenspies, Jan Kuper and Gerard J.M. Smit
Department of Electrical Engineering, Mathematics and Computer Science
University of Twente, Enschede, The Netherlands
{e.degroote, p.k.f.holzenspies, j.kuper, g.j.m.smit}@utwente.nl

ABSTRACT

Exact analysis of synchronous dataflow (SDF) graphs is often considered too costly, because of the expensive transformation of the graph into a single-rate equivalent. As an alternative, several authors have proposed *approximate* analyses. Existing approaches to approximation are based on the *operational semantics* of an SDF graph.

We propose an approach to approximation that is based on *functional semantics.* This generalises earlier work done on multi-rate SDF graphs towards cyclo-static SDF (CSDF) graphs. We take, as a starting point, a mathematical characterisation, and derive two transformations of a CSDF graph into HSDF graphs. These HSDF graphs have the same size as the CSDF graph, and are approximations: their respective temporal behaviours are optimistic and pessimistic with respect to the temporal behaviour of the CSDF graph. Analysis results computed for these single-rate approximations give *bounds* on the analysis results of the CSDF graph. As an illustration, we show how these single-rate approximations may be used to compute bounds on the buffer sizes required to reach a given throughput.

Categories and Subject Descriptors

F.1 [**Computation by Abstract Devices**]: Models of Computation

General Terms

Algorithms, Theory.

Keywords

Synchronous Dataflow, Transformation, Approximation.

1. INTRODUCTION

Synchronous dataflow (SDF) [13] is a popular model of computation, used to conservatively model real-time stream processing applications. The model is attractive, because it allows for analysis of a modelled application's temporal behaviour. Such analyses allow for e.g., the derivation of static order schedules, computation of throughput, verification of latency constraints, and optimisation of buffer sizes under a throughput constraint. The model is particularly suitable in the domain of real-time embedded systems, as analyses provide *guarantees* with respect to performance.

Different varieties of SDF exist, such as homogeneous (or single-rate) SDF (HSDF) [13], multi-rate SDF (MRSDF, or simply SDF) [13] and cyclo-static SDF (CSDF) [2]. These models differ in their succinctness, where (of these three) HSDF is the least, and CSDF the most succinct. In an HSDF graph, all actors produce data at the same rate, whereas in MRSDF these rates may vary per actor and per channel. The CSDF model is a generalisation of MRSDF, and allows for actors with periodically varying behaviour. As a result, CSDF models can capture streaming applications with greater accuracy than MRSDF graphs can, which leads to a reduction of the resource usage of implementations following this model [15].

Efficient analysis techniques for HSDF graphs exist; the throughput and associated static order schedule of such a graph may be computed in polynomial time [5]. These techniques can not be applied directly to MRSDF and CSDF graphs, however; for these graphs, a transformation into an *equivalent* HSDF graph is required. The size of such an equivalent HSDF graph may, in the worst case, be exponential in the size of the original graph, making analyses that involve the construction of these equivalent HSDF graphs very expensive in terms of computation time and memory.

Instead of constructing an equivalent HSDF graph, some authors have proposed state-space exploration of a self-timed execution of the graph as an analysis method. As a self-timed execution of an SDF graph will eventually settle in a periodically repeating schedule, a graph's throughput can be determined from its self-timed execution. This approach may be tailored to the optimisation of buffer sizes under a throughput constraint. Although they work quite efficiently in practice, these methods suffer from the same problem as those based on the construction of an equivalent HSDF graph.

As an alternative to exact analysis, approaches have been proposed to compute approximations on performance characteristics. These approximations involve the construction of linear bounds on the execution schedules of the actors in the graph. With these bounds, analysis and optimisation problems may be formulated as linear programs, which may be solved efficiently.

Existing approaches to exact and approximate analysis of SDF graphs may be characterised as *operational*: they are

SCOPES'14, June 10 - 11 2014, Sankt Goar, Germany
Copyright 2014 ACM 978-1-4503-2941-5/14/06...$15.00.
http://dx.doi.org/10.1145/2609248.2609249

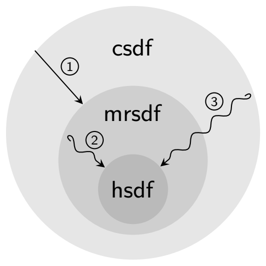


Figure 1: Transformations between different classes of dataflow: 1. exact [7], 2. approximate [6], 3. approximate (this paper).

derived from the graph's temporal behaviour, i.e., its execution schedule. Instead, we propose a *functional* approach, in which we give mathematical abstractions of the dependencies of actors. We define so-called *predecessor functions*, which describe the times actors are enabled for their k^{th} execution in terms of its dependencies. By bounding these functions, we can describe systems that model the original system either conservatively, or optimistically, giving bounds on performance from *both* sides. To our knowledge and at the time of writing, our approach is still the only one to also provide optimistic bounds, which bounds the degree of over-dimensioning, based on the conservative analysis.

In earlier work, we have presented a single-rate approximation [6], where MRSDF graphs are rewritten into HSDF graphs of precisely the same size. Also, we have presented an exact transformation from CSDF graphs into MRSDF graphs [7], in which every actor is expanded into a number of actors equal to the number of phases. This paper presents a single-rate approximation for CSDF, i.e., an approximate transformation that rewrites a CSDF graph into an HSDF graph. The resulting set of transformations is depicted in Figure 1.

The transformations from our earlier work can be composed to first transform a CSDF graph to MRSDF (exactly, growing by the number of phases) and then to HSDF (approximately, not growing). The method proposed in this paper gives a rougher approximation than this composition of previous methods, but is constant in the size of the input graph. For MRSDF graphs, however, this method improves upon our earlier approximation method. The main contribution of this paper, therefore, is not to obtain improved accuracy, but rather to expand the sound mathematical characterisation for the temporal behaviour of MRSDF graphs presented in our earlier work to the class of CSDF graphs.

We discuss existing approximation methods in more detail in Section 2. We introduce briefly the class of CSDF graphs and our notations (Section 3), before we discuss the deduction of predecessor functions (Section 4). Using these, we give our single-rate approximation of CSDF graphs (Section 5) and evaluate its efficacy by means of applying it to a well-studied case of SDF analysis, namely buffer-capacity estimation (Section 6).

2. RELATED WORK

Literature on synchronous dataflow can be categorised into *exact* and *approximate* methods. Exact analysis of (MRSDF and) CSDF graphs exploit the fact that the (self-timed) schedule of a consistent SDF graph, after an initial transient phase, follows a repetitive pattern, which is composed of several so-called *iterations* [9]. Such an iteration may be explicitly represented by transforming the CSDF graph into an equivalent HSDF graph, which has a single actor for each individual firing in the iteration [8]. For these HSDF graphs, efficient analysis techniques are available [5, 12]. However, as the length of a single iteration, in the worst case, grows exponentially in the size of the CSDF graph, approaches that rely on the construction of an equivalent HSDF graph are often considered too costly for practical use [10, 17].

As an efficient alternative, approaches that explore the state-space of a self-timed execution for its periodicity have become the common approach to analysing SDF graphs [10]. This approach may be tailored to the optimisation of buffer sizes under a throughput constraint [17]. Although in the worst case, these approaches suffer from the same problem that prohibits methods based on equivalent HSDF graphs, experiments suggest that they run much faster, primarily because they avoid the costly MRSDF-to-HSDF transformation [10, 17].

Rather than performing an exact analysis, an approximate model of the SDF graph may be constructed and analysed with much less effort [6, 11]. Such an approximate model gives a conservative estimation of the graph's performance; the actual performance is never worse than the performance computed from the model. This perfectly suits the typical aim of dataflow analysis, which is to give *guarantees* on the properties of a real-time system. However, the error made by such an approximation may be considerable, and lead to a costly over-dimensioning of the system at hand. Methods to bound this error are only known for MRSDF, but not for CSDF [6]. The transformation presented in this paper generalises [6] towards CSDF graphs. Furthermore, by explicitly taking into account the greatest common divisor of the cumulative production and consumption of tokens onto and from a channel, this work improves upon [6].

Many of the existing approaches to approximate analysis of SDF graphs work by capturing the analysis problem in a linear program. Although a linear program may be solved in polynomial time when using sophisticated techniques or clever pivoting rules, our proposed derivation of single-rate approximations allows well-established analysis algorithms to be applied. Furthermore, the structural correspondence between the CSDF graph and its single-rate approximation allows results computed for the latter, to be translated back to the CSDF graph in a trivial way.

The accuracy of approximating required buffer sizes for a minimum throughput in a CSDF graph improves if the individual phases of an actor are considered. This finding is presented in [1], where schedules of actors and phases of actors are captured in a so-called MIN-MAX linear program, which may be solved in polynomial time. A different approach to considering the individual phases of an actor is presented in [7], where a CSDF graph is transformed into an equivalent MRSDF graph. The goal of this paper is not to achieve maximum accuracy in buffer sizing, but rather to give a mathematical characterisation of the temporal behaviour of a CSDF graph. If maximum accuracy is desired, we propose to transform the subgraph of interest into MRSDF, and apply MRSDF-specific techniques to obtain tighter bounds.

3. CYCLO-STATIC DATAFLOW GRAPHS

A cyclo-static dataflow (CSDF) graph is a directed graph, where the vertices are called actors and the edges are called channels. Actors model tasks, channels model data dependencies between tasks. Actors may *fire*, which models the execution of the task represented by that actor. Execution of a task requires data to be available on incoming channels, and produces data on outgoing channels. This data is modelled by *tokens*. In a synchronous dataflow graph, the number of tokens produced onto and consumed from channels is specified by *rates*.

In a CSDF graph, actors have cyclically varying behaviour: each actor cycles through a fixed number of *phases*. The number of tokens produced onto or consumed from a channel, as well as the execution time of the actor, depends on the current phase of the actor. We use the term *period* to refer to the number of phases of an actor, and denote the period of actor v by φ_v. The phase of the actor can be derived from the actor's *firing index* in a straightforward way: the behaviour of the actor during its k^{th} firing is given by its $(k \bmod_1 \varphi_v)^{\text{th}}$ phase[1].

Each actor v has an associated *execution time vector*, which we denote by $T_v = [t_1, \ldots, t_{\varphi_v}] \in \mathbb{N}^{\varphi_v}$. At the start of an execution, an actor consumes data from its incoming channels. The completion of the k^{th} execution occurs $t_{k \bmod_1 \varphi_v}$ time units after said execution started, and involves the production of tokens onto its outgoing channels. For the sake of brevity, we write $\tau_v(k)$ to denote the execution time of the k^{th} firing of actor v.

Each channel vw has an initial, integer number of tokens, denoted δ_{vw}. Furthermore, two vectors are associated with each channel vw. These vectors are the channel's production rate vector, denoted $P^+_{vw} = [p^+_1, \ldots, p^+_{\varphi_v}] \in \mathbb{N}^{\varphi_v}$, and consumption rate vector, denoted $P^-_{vw} = [p^-_1, \ldots, p^-_{\varphi_w}] \in \mathbb{N}^{\varphi_w}$. The k^{th} firing of actor v produces $p^+_{k \bmod_1 \varphi_v}$ tokens onto the channel, whereas the k^{th} firing of actor w consumes $p^-_{k \bmod_1 \varphi_w}$ tokens from the channel. We write $\rho^+_{vw}(k)$ and $\rho^-_{vw}(k)$ as respective shorthand notations. If, on each of an actor's incoming channels, the number of tokens is at least the channel's consumption rate, the actor may start an execution and is said to be *enabled*.

We often need to refer to the total number of tokens produced or consumed by an actor in one period. We therefore denote the number of tokens produced onto channel vw in one period of v by $P^{\Sigma+}_{vw} = \sum_{i=1}^{\varphi_v} \rho^+_{vw}(i)$, and the number of tokens consumed, in one period of w, from channel vw by $P^{\Sigma-}_{vw} = \sum_{i=1}^{\varphi_w} \rho^-_{vw}(i)$.

Finally, relative primality of production and consumption rates plays an important role in the approximation of CSDF graphs (see Section 5). The vector g associates a CSDF channel with the greatest common divisor of the sums of the rate vectors associated with that channel:

$$g_{vw} = \gcd\left(P^{\Sigma+}_{vw}, P^{\Sigma-}_{vw}\right). \tag{1}$$

Cyclic dependencies in a CSDF graph limit the frequency at which actors may fire. This frequency depends on the rates and tokens associated with the channels that compose these cycles. For a given channel vw, actor w completes, on average, $P^{\Sigma+}_{vw}\varphi_w$ firings for every $P^{\Sigma-}_{vw}\varphi_v$ completed firings of actor v. Let the *gain* of a channel be defined as:

$$\text{gain}_{vw} = \frac{P^{\Sigma-}_{vw}\varphi_v}{P^{\Sigma+}_{vw}\varphi_w},$$

and the gain of a *path* be the product of the gains of the channels that compose the path.

If the gain of each cycle equals one, then the graph is said to be *consistent* [6]. The firing times of actors in a *consistent* CSDF graph follow a repetitive pattern [2]. If, furthermore, the graph is strongly connected, then the number of tokens that accumulates on a channel during execution, is bounded [9].

For a consistent CSDF graph, an integer vector q exists such that, for every channel vw, the following, so-called *balance equation* holds:

$$\frac{q_v}{\varphi_v} P^{\Sigma+}_{vw} = \frac{q_w}{\varphi_w} P^{\Sigma-}_{vw}. \tag{2}$$

with the restriction that for each actor v, q_v is an integer multiple of φ_v [2]. Vector q is commonly referred to as the graph's *repetition vector*, and gives rise to the definition of a *graph iteration*: in a single graph iteration, actor v fires precisely q_v times. Note that due to the added restriction, the repetition vector entries are not necessarily relatively prime.

Furthermore, a minimal integer vector s exists, such that, for each actor v, the following, so-called *flow conservation equation* holds:

$$\frac{s_{uv}}{\varphi_v} P^{\Sigma-}_{uv} = \frac{s_{vw}}{\varphi_v} P^{\Sigma+}_{vw} \in \mathbb{N}. \tag{3}$$

Vector s is commonly referred to as a *P-semiflow* in the context of Petri Nets [19]. In the context of dataflow graphs, we refer to it as *flow normalisation vector*. The flow normalisation vector gives the ratio between the number of tokens that flows through channels in a single iteration.

We combine the repetition and flow normalisation vectors into the following constant, which we denote $\mathcal{N}$:

$$\mathcal{N} = \frac{q_v P^{\Sigma+}_{vw}}{\varphi_v} s_{vw}. \tag{4}$$

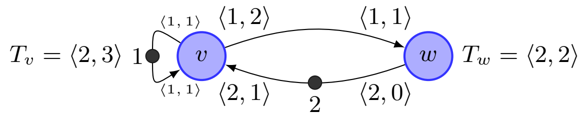
If we multiply the rate vectors associated with a channel with the corresponding entry in the flow normalisation vector, then an integer k exists such that in k iterations, precisely $\mathcal{N}$ tokens "flow through" each channel in the graph. We use the constant $\mathcal{N}$ as a scaling factor in the construction of the single-rate approximations, as outlined in Section 5.

We conclude this section with an illustration of the structural invariants, using an example CSDF graph, depicted in Figure 2. Each of the two actors in the graph has a period of 2. A repetition vector q that satisfies the balance equations is given by $q_v = 4, q_w = 6$. The smallest integer vector s that satisfies the flow conservation equations is given by $s_{vv} = 3$ and $s_{vw} = s_{wv} = 2$. As a result, the constant $\mathcal{N}$ for the graph equals 12.

4. THE PREDECESSOR FUNCTION

A CSDF graph is an example of a *discrete event system* [4, 12]. In such a system, the firing times of actors are interdependent; an actor can not start a firing before the actors it depends on have fired (at least) a given number of times. In synchronous dataflow, these dependencies are given by the graph's channels, and the details of a dependency of an actor

[1]We write $k \bmod_1 n$ as a shorthand notation for $(k-1) \bmod n + 1$, with the mod operator defined conventionally as: $a \bmod b = a - b\left\lfloor \frac{a}{b} \right\rfloor$.



(a) CSDF graph

Figure 2: An example of a consistent CSDF graph.

can be determined from the annotations (rates and tokens) on the incoming channels of that actor.

These dependencies may be elegantly described by a set of recurrent equations. We write $t_w(k)$ to denote the *completion time* of the k^{th} firing of actor w. The dependency of $t_w(k)$ on the completion time of connected upstream actors then satisfies:

$$t_w(k) = \max_{vw} \{t_v(k - \gamma(k))\} + \tau_w(k). \tag{5}$$

That is, w *starts* its k^{th} firing as soon as each predecessor v has completed $(k-\gamma(k))$ firings. Actor w needs an additional $\tau_w(k)$ time units to *complete* this execution. Note that we take the maximum over all *incoming channels* of w in the graph, and omit the quantification of vw in the equation, for convenience.

Equation (5) is commonly named a *dater* equation [12], as it defines the *times* at which actors fire. Function $\gamma(k)$ is called the *shift function* [4]. If $\gamma(k)$ is a constant integer, then (5) is a so-called *shift-invariant system*[2]. For shift-invariant discrete event systems, many analysis techniques are available [5, 12].

For dataflow systems, the shift function that is associated with a channel vw captures the fact that a certain, minimal number of tokens must have been produced by a *producing* actor, before the *consuming* actor can start its next execution. We therefore rewrite (5) into the following, slightly more convenient form:

$$t_w(k) = \max_{vw} \{t_v(\pi_{vw}(k))\} + \tau_w(k), \tag{6}$$

where $\pi_{vw}(k)$ is the *predecessor function*, which gives the number of firings predecessor v must have completed, such that enough tokens are available to let actor w start its k^{th} firing. Note that π_{vw} must be *non-decreasing* to obey the semantics of a dataflow graph: dataflow channels represent unbounded FIFO buffers, and to guarantee functional correctness, tokens may not "overtake" each other [14]. The predecessor function must thus be such that the order in which tokens are consumed does not violate the order in which tokens are produced.

There is a strong correspondence between different classes of SDF graphs and the form of the associated predecessor function [6, 7]. Multi-rate and cyclo-static SDF graphs correspond to so-called *periodically shift-varying* systems [6], and an HSDF graph may be represented by a shift-invariant system (and vice versa). The number of tokens on an HSDF channel is then simply equal to the amount of shift in the associated predecessor function [6, 8].

The interchangeability between a synchronous dataflow graph and a set of dater equations allows transformations of an SDF graph to be stated in terms of algebraic manipulations of the predecessor function. Transforming an SDF graph into HSDF thus corresponds to rewriting (6) into a shift-invariant form. A mathematical characterisation of the predecessor function associated with a channel involves the channel's production and consumption rates, and its initial number of tokens. Consider the following function Δ_{vw}, which gives the number of tokens on a channel vw, as a function of the number of firings of the actors connected by that channel:

[2]A shift-invariant system is the discrete analogue of a time-invariant system

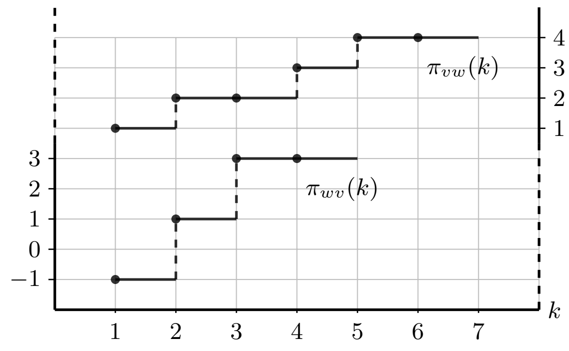


Figure 3: A graphical representation of a single period of the predecessor functions of channels vw and wv of the example CSDF graph from Figure 2.

$$\Delta_{vw}(i,j) = \delta_{vw} + \sum_{l=1}^{i} \rho^{+}_{vw}(l) - \sum_{l=1}^{j} \rho^{-}_{vw}(l). \tag{7}$$

The general form of the predecessor function associated with a channel in a synchronous dataflow graph may be expressed in terms of (7), in the following way:

$$\pi_{vw}(k) = \min \{m | \Delta_{vw}(m,k) \geq 0\}, \tag{8}$$

which says that the minimal number of firings of actor v that must precede the k^{th} firing of actor w is such that a minimum, non-negative number of tokens is available on channel vw.

An alternative formulation that is semantically equivalent is:

$$\pi_{vw}(k) = \max \{m | \Delta_{vw}(m,k) < 0\} + 1, \tag{9}$$

which says that the required number of firings of actor v is the first firing *after* its latest firing that does *not* enable the k^{th} firing of actor w.

The graph of a predecessor function of a CSDF channel is periodic and follows a staircase-like pattern. Figure 3 depicts the predecessor functions of two of the three channels of the graph shown in Figure 2.

4.1 Computing the predecessor function

The two formulations of the predecessor function given in the previous section are not very suitable for *computing* the predecessor of a given firing. We therefore rewrite (8) and (9) into a more suitable form, using the following identities:

$$\left\lceil \frac{n}{m} \right\rceil = \min \left\{ k \;\middle|\; km \geq n \right\}, \tag{10a}$$

$$\left\lfloor \frac{n}{m} \right\rfloor = \max \left\{ k \;\middle|\; km \leq n \right\}, \tag{10b}$$

$$\left\lceil \frac{n}{m} \right\rceil = \left\lfloor \frac{n+m-1}{m} \right\rfloor. \tag{10c}$$

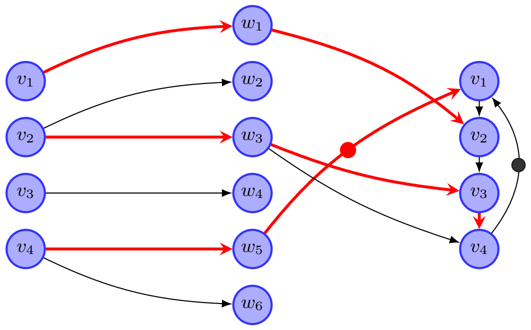


Figure 4: Equivalent HSDF graph of the example CSDF graph from Figure 2. The graph's critical cycle is shown in red.

Let $m = m'\varphi_v + i$, with $m' = \left\lfloor \frac{m}{\varphi_v} \right\rfloor$. Using this substitution, we may rewrite the min-formulation of (8) into the following form:

$$\begin{aligned} &\pi_{vw}(k) \\ &\quad = \min\left\{ m'\varphi_v + i \,\middle|\, m' P^{\Sigma+}_{vw} + \Delta_{vw}(i,k) \geq 0 \right\} \\ &\quad = \min_{i<\varphi_v}\left\{ \min\left\{ m' \,\middle|\, m' P^{\Sigma+}_{vw} \geq -\Delta_{vw}(i,k) \right\} \varphi_v + i \right\} \\ &\quad \overset{(10a)}{=} \min_{i<\varphi_v}\left\{ \left\lceil \frac{-\Delta_{vw}(i,k)}{P^{\Sigma+}_{vw}} \right\rceil \varphi_v + i \right\}. \end{aligned} \tag{11}$$

Determining the number of times an actor v must fire to produce sufficient tokens onto channel vw, such that actor w can fire k times, involves iterating the phases of v. Computing the predecessor function $\pi_{vw}(k)$ may thus be done in time proportional to φ_v.

In a similar fashion, we may rewrite the max-formulation of (9) into:

$$\begin{aligned} &\pi_{vw}(k) - 1 \\ &\quad = \max_{i<\varphi_v}\left\{ \max\left\{ m' \,\middle|\, m' P^{\Sigma+}_{vw} + \Delta_{vw}(i,k) < 0 \right\} \varphi_v + i \right\} \\ &\quad \overset{(10b)}{=} \max_{i<\varphi_v}\left\{ \left\lfloor \frac{-1-\Delta_{vw}(i,k)}{P^{\Sigma+}_{vw}} \right\rfloor \varphi_v + i + 1 \right\} \\ &\quad \overset{(10c)}{=} \max_{i<\varphi_v}\left\{ \left\lceil \frac{-\Delta_{vw}(i,k)}{P^{\Sigma+}_{vw}} \right\rceil \varphi_v + i \right\} - \varphi_v. \end{aligned} \tag{12}$$

4.2 Equivalent HSDF graphs

Using the predecessor function, we may transform a (consistent) CSDF graph into an equivalent HSDF graph, where equivalence pertains to the graphs' self-timed schedules. For such an equivalent HSDF graph, we need to compute the precedence relations between the invocations of communicating actors [2]. The predecessor function gives precisely those relations: the last token consumed by the k^{th} firing of actor w is produced by firing $\pi_{vw}(k)$ of actor v, and hence the equivalent HSDF graph contains edges from HSDF actor $v_{\pi_{vw}(k) \bmod_1 q_v}$ to HSDF actor $w_{k \bmod_1 q_w}$. The number of tokens on such an edge signifies the number of graph iterations that separate the production and consumption of the last token.

An example of such an HSDF graph representation is given in Figure 4. For the sake of clarity, the figure depicts the equivalent HSDF graph in an unrolled fashion, with all channels (except those that correspond to the self-loop of v) pointing from left to right.

The throughput of the CSDF graph of Figure 2(a) is equal to the throughput of its equivalent HSDF graph, which is the inverse of the latter graph's *maximum cycle ratio* [5,16]. The graph's maximum cycle ratio equals 16 and is attained by its critical cycle, shown in red.

5. APPROXIMATING CSDF BY HSDF

The approach that we take to transforming a CSDF graph into an approximating HSDF graph involves two main steps. As a first step, we derive linear bounds on the predecessor function associated with each channel in the CSDF graph. Here we differ from existing approaches: where bounds are conventionally constructed based on operational semantics (i.e., firing schedules), we take the mathematical characterisation, from which these operational semantics follow, as a starting point. We use basic integer arithmetic to derive a lower linear bound from the min-formulation, given by (11), and derive an upper bound, in a similar fashion, from the predecessor function's max-formulation (12).

These linear bounds change the discrete system into a continuous one, also called *fluid* by some authors [3]. This continuous system is not shift-invariant, as vertices fire at different rates. Such a system may still be analysed by formulating and solving a linear program [11]. However, we take an extra step to transform the system into one that is shift-invariant. This step involves changing the system's counting units.

As a result of this final step, the system may be represented by an HSDF graph. As such, existing, well-known analysis techniques may be applied, and analysis results obtained for the HSDF graph can naturally be translated back to the CSDF graph.

5.1 Prelude

The derivation of linear bounds on the predecessor function involves computing the maximum and minimum of the following set:

$$\left\{ \frac{km}{n} - \left\lceil \frac{km-d}{n} \right\rceil \,\middle|\, k \in \mathbb{Z} \right\}, \tag{13}$$

with constants $m, n \in \mathbb{N}$ and $d \in \mathbb{Z}$. Using basic integer arithmetic, this set may be reformulated as:

$$\left\{ \frac{d - (d-km) \bmod n}{n} \,\middle|\, k \in \mathbb{Z} \right\}. \tag{14}$$

Minimum and maximum values of this set are attained at the respective maximum and minimum possible value of $(d - km) \bmod n$. The minimum of the set thus satisfies:

$$\begin{aligned} \min_{k\in\mathbb{Z}}\left\{ \frac{km}{n} - \left\lceil \frac{km-d}{n} \right\rceil \right\} &= \frac{d - (n - \gcd(m,n) + d \bmod \gcd(m,n))}{n} \\ &= \frac{\gcd(m,n)\left\lceil \frac{d+1}{\gcd(m,n)} \right\rceil}{n} - 1, \end{aligned} \tag{15}$$

and the maximum value of the set is given by:

$$\max_{k\in\mathbb{Z}}\left\{\frac{km}{n}-\left\lceil\frac{km-d}{n}\right\rceil\right\}=\frac{d-d \bmod \gcd(m,n)}{n} = \frac{\gcd(m,n)\left\lfloor\frac{d}{\gcd(m,n)}\right\rfloor}{n}. \tag{16}$$

In the remainder of this section, identities (15) and (16) are used to derive linear bounds on the CSDF predecessor function.

5.2 Bounding the Predecessor Function

The predecessor function associated with a CSDF channel follows a repetitive staircase pattern. Tight upper and lower linear bounds may be constructed by choosing an appropriate slope and intercept. These linear bounds thus have the following form:

$$\pi^{\mathrm{up}}_{vw}(k)=\alpha_{vw}k-\beta^{\mathrm{up}}_{vw} \tag{17a}$$

$$\pi^{\mathrm{lo}}_{vw}(k)=\alpha_{vw}k-\beta^{\mathrm{lo}}_{vw}. \tag{17b}$$

The (average) slope of the staircase pattern is given by the ratios of the repetition vector entries associated with the actors connected by the channel, in the following way:

$$\begin{aligned}
&\pi_{vw}(k+mq_w)-1\\
&=\max_{\substack{i<\varphi_v\\ j<\varphi_w}}\left\{\left(\left\lceil\frac{\left(k'+m\frac{q_w}{\varphi_w}\right)P^{\Sigma-}_{vw}-\Delta_{vw}(i,j)}{P^{\Sigma+}_{vw}}\right\rceil-1\right)\varphi_v+i\right\}\\
&\overset{(2)}{=}\max_{\substack{i<\varphi_v\\ j<\varphi_w}}\left\{\left(\left\lceil\frac{k'P^{\Sigma-}_{vw}-\Delta_{vw}(i,j)}{P^{\Sigma+}_{vw}}\right\rceil+m\frac{q_v}{\varphi_v}-1\right)\varphi_v+i\right\}\\
&=\pi_{vw}(k)+mq_v-1.
\end{aligned}$$

We thus set the slope of the linear bounds on the channel's predecessor function to $\alpha_{vw}=\frac{q_v}{q_w}$.

Computation of the intercepts, β^{up} and β^{lo}, requires more effort. These intercepts should be chosen such that the linear bounds are *tight*, i.e., the minimum error between the predecessor function and its linear bound must be zero.

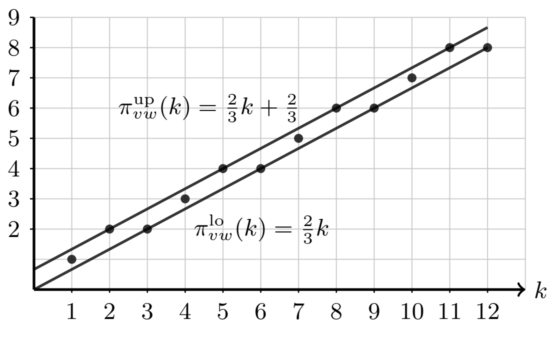
5.2.1 *Constructing an upper bound*

We construct a tight linear upper bound by choosing β^{up} such that, for all $k\in\mathbb{N}$, we have $\pi^{\mathrm{up}}_{vw}(k)\geq\pi_{vw}(k)$, and for at least one k, $\pi^{\mathrm{up}}_{vw}(k)=\pi_{vw}(k)$. This is achieved by setting β^{up} to:

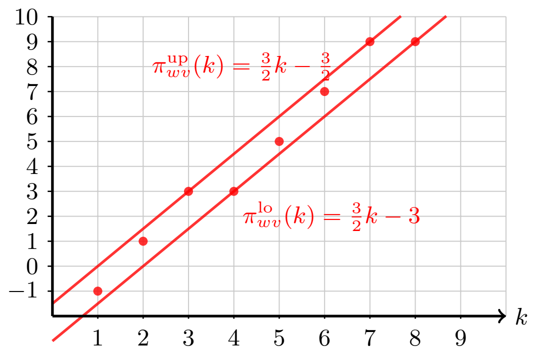
$$\beta^{\mathrm{up}}_{vw}=\min_k\left\{\alpha_{vw}k-\pi_{vw}(k)\right\}. \tag{18}$$

Computing β^{up}_{vw} using the above formulation may take considerable time, as the length of a single period of the predecessor function depends on the numerical values of the production and consumption rate vectors of vw. The following rewriting steps reduce the complexity of this computation to one that depends solely on the lengths of the production and consumption rate vectors. As a first step, we expand (18) into:

$$\begin{aligned}
&\beta^{\mathrm{up}}_{vw}-\varphi_v+1\\
&=\min_k\left\{\frac{q_v}{q_w}k-\max_{i<\varphi_v}\left\{\left\lceil\frac{-\Delta_{vw}(i,k)}{P^{\Sigma+}_{vw}}\right\rceil\varphi_v+i\right\}\right\}\\
&=\min_k\left\{\min_{i<\varphi_v}\left\{\left(\frac{kP^{\Sigma-}_{vw}}{\varphi_wP^{\Sigma+}_{vw}}-\left\lceil\frac{-\Delta_{vw}(i,k)}{P^{\Sigma+}_{vw}}\right\rceil\right)\varphi_v-i\right\}\right\}.
\end{aligned}$$



(a) Two periods of π_{vw}



(b) Two periods of π_{wv}

Figure 5: Tight linear lower and upper bounds on the predecessor functions of Figure 3.

For the sake of compactness, we introduce the following shorthand notation:

$$\psi_{vw}(i,j,k)=\left\lceil\frac{kP^{\Sigma-}_{vw}-\Delta_{vw}(i,j)}{P^{\Sigma+}_{vw}}\right\rceil. \tag{19}$$

We further reduce the complexity of the computation by exploiting the periodicity of $\Delta_{vw}(i,k)$ in k. Substituting $k'\varphi_w+j$ for k gives:

$$\begin{aligned}
&\beta^{\mathrm{up}}_{vw}+1\\
&=\min_{k'}\min_{\substack{i<\varphi_v\\ j<\varphi_w}}\left\{\left(\frac{k'P^{\Sigma-}_{vw}}{P^{\Sigma+}_{vw}}-\psi_{vw}(i,j,k')\right)\varphi_v-i+j\frac{q_v}{q_w}\right\}+\varphi_v\\
&=\min_{\substack{i<\varphi_v\\ j<\varphi_w}}\left\{\min_{k'}\left\{\frac{k'P^{\Sigma-}_{vw}}{P^{\Sigma+}_{vw}}-\psi_{vw}(i,j,k')\right\}\varphi_v-i+j\frac{q_v}{q_w}\right\}+\varphi_v\\
&=\min_{\substack{i<\varphi_v\\ j<\varphi_w}}\left\{\frac{g_{vw}\varphi_v\left\lceil\frac{\Delta_{vw}(i,j)+1}{g_{vw}}\right\rceil}{P^{\Sigma+}_{vw}}-i+j\frac{q_v}{q_w}\right\}.
\end{aligned}$$

Putting everything together gives the following linear predecessor function:

$$\begin{aligned}
\pi^{\mathrm{up}}_{vw}(k)&=\frac{q_v}{q_w}k\\
&-\min_{\substack{i<\varphi_v\\ j<\varphi_w}}\left\{\frac{g_{vw}\varphi_v\left\lceil\frac{\Delta_{vw}(i,j)+1}{g_{vw}}\right\rceil}{P^{\Sigma+}_{vw}}-i+j\frac{q_v}{q_w}-1\right\}.
\end{aligned}\tag{20}$$

A tight linear bound on the predecessor function associated with CSDF channel vw may thus be computed in time proportional to the product of the periods of actors v and

w, i.e., $\varphi_v\varphi_w$. Note that in case the greatest common divisor g_{vw} equals one, the bound may be computed in time proportional to $\varphi_v + \varphi_w$, by separating the terms in i from the terms in j.

5.2.2 *Constructing a lower bound*

We may construct a lower bound, in a way similar to the construction of an upper bound, described above. Rather than using the max-formulation of the predecessor function, we now use the min-formulation. For the lower bound, we compute the intercept β^{lo}_{vw} using:

$$\beta^{\text{lo}}_{vw} = \max_k \left\{\alpha_{vw} k - \pi_{vw}(k)\right\}, \tag{21}$$

which we expand into:

$$\begin{aligned}
&\beta^{\text{lo}}_{vw} \\
&= \max_k \left\{ \frac{q_v}{q_w} k - \min_{i<\varphi_v} \left\{ \left\lceil \frac{-\Delta_{vw}(i,k)}{P^{\Sigma+}_{vw}} \right\rceil \varphi_v + i \right\} \right\} \\
&= \max_k \left\{ \max_{i<\varphi_v} \left\{ \left(\frac{k P^{\Sigma-}_{vw}}{\varphi_w P^{\Sigma+}_{vw}} - \left\lceil \frac{-\Delta_{vw}(i,k)}{P^{\Sigma+}_{vw}} \right\rceil \right) \varphi_v - i \right\} \right\}.
\end{aligned}$$

Again, by substituting $k'\varphi_w + j$ for k, and using the shorthand notation (19), we rewrite this into:

$$\begin{aligned}
&\beta^{\text{lo}}_{vw} \\
&= \max_{k'} \max_{\substack{i<\varphi_v \\ j<\varphi_w}} \left\{ \left(\frac{k' P^{\Sigma-}_{vw}}{P^{\Sigma+}_{vw}} - \psi_{vw}(i,j,k') \right) \varphi_v - i + j\frac{q_v}{q_w} \right\} \\
&= \max_{\substack{i<\varphi_v \\ j<\varphi_w}} \left\{ \max_{k'} \left\{ \frac{k' P^{\Sigma-}_{vw}}{P^{\Sigma+}_{vw}} - \psi_{vw}(i,j,k') \right\} \varphi_v - i + j\frac{q_v}{q_w} \right\} \\
&= \max_{\substack{i<\varphi_v \\ j<\varphi_w}} \left\{ \frac{g_{vw}\varphi_v \left\lfloor \frac{\Delta_{vw}(i,j)}{g_{vw}} \right\rfloor}{P^{\Sigma+}_{vw}} - i + j\frac{q_v}{q_w} \right\}.
\end{aligned}$$

This gives the following tight linear lower bound on the CSDF predecessor function:

$$\begin{aligned}
\pi^{lo}_{vw}(k) &= \frac{q_v}{q_w} k \\
&- \max_{\substack{i<\varphi_v \\ j<\varphi_w}} \left\{ \frac{g_{vw}\varphi_v \left\lfloor \frac{\Delta_{vw}(i,j)}{g_{vw}} \right\rfloor}{P^{\Sigma+}_{vw}} - i + j\frac{q_v}{q_w} \right\}.
\end{aligned} \tag{22}$$

5.3 Changing counting units

The system that we derived in the previous section can not directly be represented as an HSDF graph, because it is not a shift-invariant system. Shift-invariance signifies that firing times progress at the same rate. As the rates at which CSDF actors fire are interrelated through the graph's repetition vector, counting the number of *iterations* an actor has completed, rather than its completed firings, yields a shift-invariant system.

Such a change in counting is obtained by scaling the domain and codomain of (20) and (22). We apply the following affine transformation to each predecessor function:

$$\dot{\pi}_{vw}(k) = \frac{1}{q_v} \pi_{vw}(k q_w). \tag{23}$$

This gives the following predecessor functions:

$$\begin{aligned}
&\dot{\pi}^{\text{up}}_{vw}(k) \\
&= k - \min_{\substack{i<\varphi_v \\ j<\varphi_w}} \left\{ \frac{g_{vw}\varphi_v \left\lceil \frac{\Delta_{vw}(i,j)+1}{g_{vw}} \right\rceil}{q_v P^{\Sigma+}_{vw}} - \frac{i+1}{q_v} + \frac{j}{q_w} \right\},
\end{aligned} \tag{24}$$

$$\begin{aligned}
&\dot{\pi}^{\text{lo}}_{vw}(k) \\
&= k - \max_{\substack{i<\varphi_v \\ j<\varphi_w}} \left\{ \frac{g_{vw}\varphi_v \left\lfloor \frac{\Delta_{vw}(i,j)}{g_{vw}} \right\rfloor}{q_v P^{\Sigma+}_{vw}} - \frac{i}{q_v} + \frac{j}{q_w} \right\}.
\end{aligned} \tag{25}$$

These predecessor functions are *time-invariant*, but not shift-invariant, as the intercept in (24) and (25) is in general not an integer. To cope with this, we scale the shift by a constant factor. This is analogous to multiplying the number of tokens on each channel in an HSDF with the same factor. Multiplying each channel's number of tokens by a factor f scales the cycle ratio of each cycle (and thus the graph's maximum cycle ratio) by a factor of $\frac{1}{f}$. Multiplication with the factor $\mathcal{N}$ gives the following linear upper bound:

$$\begin{aligned}
&\ddot{\pi}^{\text{up}}_{vw}(k) = k - \\
&s_{vw} \min_{\substack{i<\varphi_v \\ j<\varphi_w}} \left\{ g_{vw} \left\lceil \frac{\Delta_{vw}(i,j)+1}{g_{vw}} \right\rceil - \frac{(i+1)P^{\Sigma+}_{vw}}{\varphi_v} + \frac{j P^{\Sigma-}_{vw}}{\varphi_w} \right\}.
\end{aligned}$$

Note that the min-term in this expression is indeed an integer, by definition of the flow conservation vector, (3). In a similar fashion, we derive the following linear lower bound:

$$\begin{aligned}
&\ddot{\pi}^{\text{lo}}_{vw}(k) = k - \\
&s_{vw} \max_{\substack{i<\varphi_v \\ j<\varphi_w}} \left\{ g_{vw} \left\lfloor \frac{\Delta_{vw}(i,j)}{g_{vw}} \right\rfloor - \frac{i P^{\Sigma+}_{vw}}{\varphi_v} + \frac{j P^{\Sigma-}_{vw}}{\varphi_w} \right\}.
\end{aligned} \tag{26}$$

5.4 Single-Rate Approximations

The predecessor functions constructed in the previous section give rise to two sets of dater equations, each of which forms a shift-invariant system. As a final step, we need to replace each actor's execution time vector by a scalar. A *pessimistic* system is obtained by replacing each execution time vector by its maximum, and, similarly, by replacing said vectors by their minima, we obtain an *optimistic* system:

$$\hat{t}_w(k) = \max_{vw} \hat{t}_v\left(\ddot{\pi}^{\text{up}}_{vw}(k)\right) + \max_i \tau_w(i) \tag{27a}$$

$$\check{t}_w(k) = \max_{vw} \check{t}_v\left(\ddot{\pi}^{\text{lo}}_{vw}(k)\right) + \min_i \tau_w(i). \tag{27b}$$

These sets of dater equations may be represented by HSDF graphs [4,8,12]. As an example, Figures 6(a) and 6(b) depict the optimistic and pessimistic HSDF graph approximations of the example CSDF graph from Figure 2(a).

Actors in the resulting HSDF graph represent *partial* iterations. This is due to the change in counting units, which was necessary to obtain shift-invariance. The relation between the number of HSDF actor firings and the number of firings of the corresponding CSDF actor depends on the repetition vector entry associated with the actor: q_v firings of CSDF actor v correspond to $\mathcal{N}$ firings of HSDF actor v.

For example, 3 firings of actor v in the pessimistic HSDF graph of Figure 6(b), correspond to a single firing of CSDF actor v in Figure 2(a). Using this correspondence, we can derive a schedule for a CSDF graph from the schedule of its

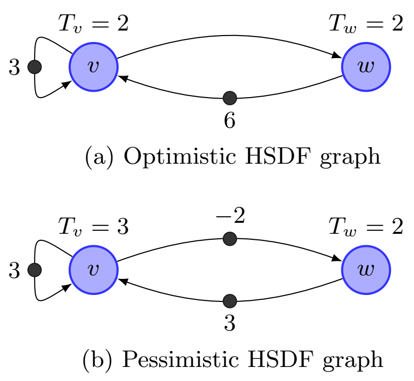


Figure 6: Optimistic and pessimistic single-rate approximations derived from the CSDF graph from Figure 2.

pessimistic (or optimistic) HSDF approximation. Likewise, we can compute bounds on the throughput of a CSDF graph by computing the throughputs of the two HSDF approximations, and multiplying these by $\mathcal{N}$. For the optimistic HSDF graph of Figure 6(a), we find that the throughput (which is the inverse of the graph's maximum cycle ratio) equals $\frac{4}{6}$, the critical cycle being vwv. For the pessimistic HSDF graph, the critical cycle is again vwv, but now limits the throughput to 5. Analysis of the approximations thus gives that the throughput lies between $12 \cdot \frac{4}{6} = 8$ and $12 \cdot 5 = 60$ (recall that the exact throughput of the graph, computed from its equivalent HSDF graph, equals 16).

Note that, unlike the pessimistic bound, the optimistic throughput bound pertains to the *model*, not to the modelled application. In this sense, the optimistic bound primarily gives an indication of the *error* of the estimated throughput; if this error is (too) large, an exact transformation into a larger graph, which exposes interactions between individual phases of the CSDF actors, may give a more accurate estimation [7].

5.5 Complexity of the Transformation

The transformation produces an HSDF graph that has the same size as the original CSDF graph. Each CSDF actor is transformed into an HSDF actor that has as a scalar execution time, which is either the minimum or the maximum of the execution time vector of the CSDF actor. The complexity of transforming the CSDF actors is thus $O(|V|\varphi^{\max})$, where $\varphi^{\max}$ is the maximum actor period, taken over all CSDF actors.

Each channel in the CSDF graph is transformed into a single channel in the resulting HSDF graph. This involves computing a scaled linear bound on the channel's predecessor function, as is described in the previous section. The computation of the intercept of this linear bound has the highest cost in this step; it requires iterating the phases of the actors connected by the channel and hence has complexity $O((\varphi^{\max})^2)$.

As a result, the complexity of the entire transformation is $O(\varphi^{\max}(|V| + \varphi^{\max}|E|))$. As the size of the input is $O(\varphi^{\max}(|V| + |E|))$, the complexity of the transformation is polynomial in the size of the CSDF graph.

5.6 Symbolic tokens

In the context of SDF graphs, an important optimisation problem is the minimisation of the number of initial tokens, under a throughput constraint. In this problem, the number of initial tokens on a channel is denoted by a variable. When transforming a CSDF graph into a single-rate approximation, the intercept of the linear bound depends on the number of initial tokens. This dependency is non-linear if the divisor g_{vw} of a channel vw is greater than one.

In order to construct a single-rate approximation in which a channel's initial tokens remain variable, an assumption on these initial tokens must be made. We may factor out the initial tokens δ_{vw} of a channel vw, if the following *congruence relation* is satisfied:

$$\delta_{vw} \equiv r_{vw} \pmod{g_{vw}}, \tag{28}$$

with $0 \leq r_{vw} < g_{vw}$. In the following section, we illustrate how the choice of the remainder r_{vw} affects the optimisation problem.

6. ESTIMATING BUFFER CAPACITIES

In this section, we show how the optimistic and pessimistic single-rate approximations of a CSDF graph can be used to compute necessary and sufficient buffer sizes under a throughput constraint. As a case study, we use a model of an MP3 playback application, which has served as a case study in several other studies as well [20, 21].

The application consists of three tasks, each of which is modelled by a single CSDF actor: Actor MP3 models an MP3 decoder that processes a 48 kHz variable bitrate MP3 file, and the sample rate converter (modelled by actor SRC) converts this to a 44.1 kHz stream to match the frequency of the digital-to-analog-converter, which is modelled by the actor labelled DAC. Communication channels between the tasks are FIFO buffers; their finite capacity is modelled by a reverse channel, with a number of initial tokens equal to the buffer's capacity. To leave these buffer capacities unspecified, the number of tokens on these reversed channels are expressed symbolically. The model is depicted in Figure 7.

The worst-case execution time of the ten different phases of the sample rate converter task are (in order): 136577, 133824, 133760, 133750, 133748, 133863, 133844, 133955, 133882, and 133862 clock cycles. The MP3 decoder task has a worst-case execution of 1603621 clock cycles, and the digital-to-analog-converter samples its input periodically every 5000 clock cycles. The latter gives a minimal required throughput; in order not to stall the DAC actor, data should arrive in time. Because the repetition vector entry of the DAC actor is 5292, the cycle time of the graph should thus, at most, be $5000 \cdot 5292$.

The throughput of the graph depends on the capacity of the two buffers between the three tasks, i.e., the variables d_1 and d_2. If we approximate these buffer sizes required to reach the desired throughput, using the techniques from [20], we find that d_1 must be at least 1536 and d_2 must be at least 90. The analysis does not provide an error margin. Exact analysis, using the SDF3 tool [18], reveals that $d_1 = 1152$ and $d_2 = 65$ are the optimal choices for the buffer capacities.

If we apply the transformation presented in this paper, then we need an additional constraint on d_1, since the sums of the production and consumption rate vectors associated with the channels connecting MP3 and SRC are not relatively

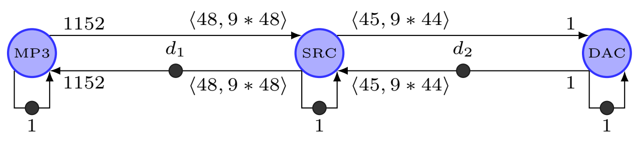
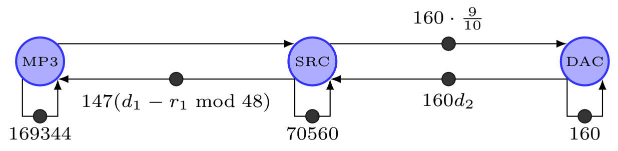
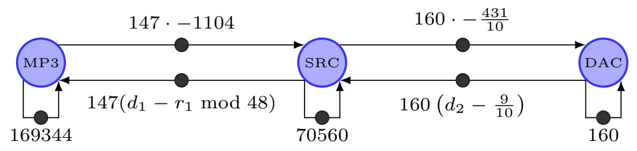


Figure 7: A Cyclo-Static dataflow graph model of an MP3 playback application. The capacities of the buffers between the tasks are captured by (integer) variables d_1 and d_2.



(a) Optimistic HSDF graph approximation.



$147 \cdot -1104$
$160 \cdot -\frac{431}{10}$
MP3
SRC
DAC
$147(d_1 - r_1 \bmod 48)$
$160\left(d_2 - \frac{9}{10}\right)$
169344
70560
160

(b) Pessimistic HSDF graph approximation.

Figure 8: Single-rate approximations obtained from the CSDF graph from Figure 7.

prime. This constraint is given as the following congruence relation:

$$d_1 \equiv r_1 \pmod{96}. \tag{29}$$

With this constraint, the term $d_1 - r_1$ may be factored out of the rounding functions (floor and ceiling) that occur in the linear bounds. By furthermore exploiting the fact that the rates on the channels connecting MP3 and SRC are constant, the pessimistic predecessor function with the channel marked with d_1 tokens can be simplified into:

$$\begin{aligned}\pi(k) &= k - 147 \min_{i<10} \left\{ 96 \left(\left\lceil \frac{d_1 + 48i + 1}{96} \right\rceil - \frac{i+1}{2} \right) \right\} \\ &= k - 147 \min \left\{ \left\lfloor \frac{r_1}{96} \right\rfloor + \frac{1}{2}, \left\lfloor \frac{r_1 + 48}{96} \right\rfloor \right\} + d_1 - r_1 \\ &= k - 147 \left(d_1 - r_1 + 48 \left\lfloor \frac{r_1}{48} \right\rfloor \right) \\ &= k - 147 \left(d_1 - r_1 \bmod 48 \right),\end{aligned}$$

and the optimistic predecessor function can be rewritten into the same expression:

$$\begin{aligned}\pi(k) &= k - 147 \max_{i<10} \left\{ 96 \left(\left\lfloor \frac{\lfloor \frac{r_1}{48} \rfloor + i}{2} \right\rfloor - \frac{i}{2} \right) + d_1 - r_1 \right\} \\ &= k - 147 \left(d_1 - r_1 \bmod 48 \right).\end{aligned}$$

The optimistic and pessimistic single-rate approximations are depicted in Figure 8. The scaling factor $\mathcal{N}$ used in the transformation equals 846720. Since the maximum allowed cycle time of the CSDF graph is $5000 \cdot 5292$, the maximum cycle ratio of the single-rate approximations must not be larger than $\frac{5000 \cdot 5292}{846720} = \frac{125}{4}$.

The minimum required size of the buffers can be obtained from the cycle ratios of the two cycles. The cycle ratio of the buffer between SRC and DAC equals:

$$\lambda = \frac{136577 + 5000}{160(d_2 - 44)} \leq \frac{125}{4}.$$

The minimum integer value of d_2 that satisfies this is 73.

For the buffer between MP3 and SRC, we need to consider two possible choices for r_1, namely $r_1 = 0$ and $r_1 = 48$. Regardless of our choice, the constraint under which we optimise d_1 is the same:

$$\lambda = \frac{1603621 + 136577}{147(d_1 - 1104)} \leq \frac{125}{4}.$$

The minimum value of d_1 that satisfies this constraint is $d_1 = 1482.82$. Under the constraint that $r_1 = 0$, the smallest integer value of d_1 is 1536, whereas if we take $r_1 = 48$, we find $d_1 = 1488$. The minimum capacity for the buffer between the MP3 task and the SRC task is thus 1488.

The computed buffer capacities are derived from the pessimistic HSDF graph. That is, they are *sufficient*; with the computed capacities, the actual throughput may be better than required, but not worse. If we conduct the same analysis on the optimistic HSDF graph approximation, depicted in Figure 8(a), then we obtain *necessary* buffer sizes.

For the optimistic case, the constraints on the cycle ratios are slightly different. Again, regardless of our choice for r_1 (either 0 or 48), we have:

$$\lambda = \frac{1603621 + 133748}{147 d_1} \leq \frac{125}{4}.$$

The minimum (real) value of d_1 that satisfies the constraint is 378.2. If we choose $r_1 = 0$, we obtain the minimal integer value that satisfies the constraint: $d_1 = 384$.

For the other buffer, we have:

$$\frac{133748 + 5000}{160(d_2 + \frac{9}{10})} \leq \frac{125}{4},$$

which gives as a minimal integer value $d_2 = 27$.

The necessary and sufficient buffer capacities lie quite far apart. This is due to the fact that in single-rate (HSDF) graphs, throughput is composable, whereas this is not the case in multi-rate or cyclo-static SDF graphs. In the single-rate approximations, the throughput of two adjoint cycles is the minimum of the throughputs of the individual cycles, whereas generally, in an SDF graph, the throughput of two adjoint cycles may be lower than the throughputs of the individual cycles.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we present a method to approximate a cyclo-static synchronous dataflow (CSDF) graph by two homogeneous or single-rate dataflow graphs. Whereas existing approaches derive from the operational semantics of a CSDF graph (i.e., the graph's execution schedule), we take a *functional* approach: we capture the temporal behaviour of a CSDF graph in a set of so-called *predecessor functions*, from which the operational semantics follow. The presented transformation generalises and improves upon earlier work done on MRSDF graphs [6]. Analysis results computed for the single-rate approximations provide bounds on the exact results obtained on the CSDF graph.

The transformation has polynomial time complexity, and produces HSDF graphs that have the same size as the CSDF graph. Given that polynomial-time algorithms exist for the analysis of HSDF graphs, the transformation offers a rough but quick estimation of the performance characteristics of a CSDF graph. To our knowledge, our approach is still the only one to also provide *optimistic* bounds. This quantifies the degree of over-dimensioning that is due to the conservative bound.

We give an application of these single-rate approximations to the well-known problem of optimising buffer capacities under a throughput constraint. We furthermore illustrate how congruence relations on buffer capacities are involved in solving this problem.

In future work, we plan to integrate approximate analyses, such as presented in this paper, with exact (partial) transformations such as outlined in [7]. Such an integrated approach should lead to an *incremental* analysis of SDF graphs, where inaccurate analysis results, obtained from single-rate approximations, are improved in a stepwise fashion. The presented mathematical characterisation, in the form of predecessor functions, and its approximation, provides an essential basis for the targeted approach.

Acknowledgement

This research is partly supported by EU projects SENSATION (FP7-ICT-2011-8, grant agreement no. 318490) and POLCA (FP7-ICT-2013-10, grant agreement no. 610686).

8. REFERENCES

[1] M. Benazouz and A. Munier-Kordon. Cyclo-static DataFlow phases scheduling optimization for buffer sizes minimization. In *Proceedings of the 16th International Workshop on Software and Compilers for Embedded Systems (M-SCOPES)*, pages 3–12, New York, NY, USA, June 2013. ACM Press.

[2] G. Bilsen, M. Engels, R. Lauwereins, and J. Peperstraete. Cyclo-Static dataflow. *IEEE Transactions on Signal Processing*, 44(2):397–408, 1996.

[3] G. Cohen, S. Gaubert, and J. Quadrat. Timed-event graphs with multipliers and homogeneous min-plus systems. *IEEE Transactions on Automatic Control*, 43(9):1296–1302, 1998.

[4] G. Cohen, G. J. Olsder, and J.-P. Quadrat. *Synchronization and linearity*. Wiley New York, 1992.

[5] A. Dasdan. Experimental analysis of the fastest optimum cycle ratio and mean algorithms. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 9(4):385–418, 2004.

[6] R. de Groote, P. K. F. Hölzenspies, J. Kuper, and H. J. Broersma. Back to Basics: Homogeneous Representations of Multi-Rate Synchronous Dataflow Graphs. In *Proceedings of the 11th ACM-IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, pages 35–46, Oct. 2013.

[7] R. de Groote, P. K. F. Hölzenspies, J. Kuper, and G. J. M. Smit. Multirate Equivalents of Cyclo-Static Synchronous Dataflow Graphs. In *Proceedings of the 14th International Conference on Application of Concurrency to System Design (ACSD)*, June 2014. to appear.

[8] R. de Groote, J. Kuper, H. Broersma, and G. J. Smit. Max-Plus Algebraic Throughput Analysis of Synchronous Dataflow Graphs. In *Proceedings of the 38th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 29–38. IEEE, Sept. 2012.

[9] A. H. Ghamarian, M. Geilen, T. Basten, B. D. Theelen, M. R. Mousavi, and S. Stuijk. Liveness and Boundedness of Synchronous Data Flow Graphs. In *Proceedings of the 6th International Conference on Formal Methods in Computer Aided Design (FMCAD)*, pages 68–75. IEEE, Aug. 2006.

[10] A. H. Ghamarian, M. C. W. Geilen, S. Stuijk, T. Basten, B. D. Theelen, M. R. Mousavi, A. J. M. Moonen, and M. J. G. Bekooij. Throughput Analysis of Synchronous Data Flow Graphs. In *Proceedings of the 6th International Conference on Application of Concurrency to System Design (ACSD)*, pages 25–36. IEEE, June 2006.

[11] J. P. Hausmans, S. J. Geuns, M. H. Wiggers, and M. J. Bekooij. Compositional temporal analysis model for incremental hard real-time system design. In *Proceedings of the 10th ACM International Conference on Embedded Software (EMSOFT)*, pages 185–194, New York, NY, USA, Oct. 2012. ACM Press.

[12] B. Heidergott, G. J. Olsder, and J. van der Woude. *Max Plus at Work: modeling and analysis of synchronized systems*. Princeton University Press, 2006.

[13] E. Lee and D. Messerschmitt. Synchronous data flow. *Proceedings of the IEEE*, 75(9):1235–1245, 1987.

[14] E. Lee and T. Parks. Dataflow process networks. *Proceedings of the IEEE*, 83(5):773–801, May 1995.

[15] A. Moonen, M. Bekooij, R. van den Berg, and J. van Meerbergen. Practical and Accurate Throughput Analysis with the Cyclo Static Dataflow Model. In *Proceedings of the 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 238–245. IEEE, Oct. 2007.

[16] R. Reiter. Scheduling Parallel Computations. *Journal of the ACM*, 15(4):590–599, Oct. 1968.

[17] S. Stuijk, M. Geilen, and T. Basten. Throughput-Buffering Trade-Off Exploration for Cyclo-Static and Synchronous Dataflow Graphs. *IEEE Transactions on Computers*, 57(10):1331–1345, Oct. 2008.

[18] S. Stuijk, M. C. W. Geilen, and T. Basten. SDF^3: SDF For Free. In *Proceedings of the 6th International Conference on Application of Concurrency to System Design*, pages 276–278. IEEE Computer Society Press, Los Alamitos, CA, USA, June 2006.

[19] E. Teruel, P. Chrzastowski-Wachtel, J. M. Colom, and M. Silva. On Weighted T-Systems. *Lecture Notes In Computer Science; Vol. 616*, page 348, 1992.

[20] M. Wiggers, M. Bekooij, and G. Smit. Efficient Computation of Buffer Capacities for Cyclo-Static Dataflow Graphs. In *Proceedings of the 44th ACM/IEEE Design Automation Conference (DAC)*, pages 658–663. IEEE, June 2007.

[21] M. H. Wiggers. *Aperiodic Multiprocessor Scheduling for Real-Time Stream Processing Applications*. PhD thesis, University of Twente, Enschede, The Netherlands, June 2009.