Back to Basics: Homogeneous Representations of Multi-Rate Synchronous Dataflow Graphs

Robert de Groote, Philip K.F. Hölzenspies, Jan Kuper, Hajo Broersma
Department of Electrical Engineering, Mathematics and Computer Science
University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands
{*e.degroote, p.k.f.holzenspies, j.kuper, h.j.broersma*}*@utwente.nl*

***Abstract*—Exact temporal analyses of multi-rate synchronous dataflow (MRSDF) graphs, such as computing the maximum achievable throughput, or sufficient buffer sizes required to reach a minimum throughput, require a homogeneous representation called a homogeneous synchronous dataflow (HSDF) graph. The size of such an HSDF graph may, in the worst case, be exponential in the size of the MRSDF graph. In this paper, we revisit the transformation from MRSDF to HSDF, and show how this transformation may be done either exactly or approximately. The approximate transformation gives both an *optimistic* and a *pessimistic* HSDF graph, each of which has the same size as the MRSDF graph. We furthermore show how strict lower and upper bounds on throughput, or on the buffer sizes required to reach a minimum throughput, may be obtained from these approximating graphs.**

I. Introduction

Synchronous dataflow (SDF) [1] is a popular class of models of computation and is used to model the temporal aspects of signal processing and streaming multimedia applications [2]. Synchronous dataflow models are particularly useful in the design of systems where worst-case guarantees on performance must be given. An SDF graph allows for the explicit modelling of (limited) system resources such as buffer space and processing elements. The class of SDF graphs that we focus on in this paper are called multi-rate SDF (MRSDF). In an MRSDF graph, actors may differ in the rates at which they produce and consume data.

This paper focuses on the *timing analysis* of MRSDF graphs. If an MRSDF graph represents an application with a specific allocation of system resources, then timing analysis of an MRSDF graph allows a designer to compare the performance of different mappings of the application onto hardware, thereby guiding the search for a design that meets the requirements and constraints at hand.

Exact timing analysis of an MRSDF graph has a complexity that is exponential in the graph's size [2], [3]. This exponential complexity is due to the fact that a *homogeneous representation* (called a *homogeneous* SDF or HSDF graph) of the MRSDF graph needs to be constructed and analysed.

Timing analysis is not always required to be exact; estimates or bounds may be sufficient to assist the designer in exploring possible designs. Therefore, approximate methods have been proposed in literature [4], [5]. The approximation given by these methods is *conservative*, i.e., the computed performance is never better than the actual performance, which suffices in cases where guarantees must be given. However, the designs that are obtained with only conservative approximations may be severely over-dimensioned, as the (unknown) error made by the approximation may be substantial.

Surprisingly, the existing approximate methods do not follow naturally from the exact methods. This "misalignment" is the reason that, in this paper, we reconsider the exact and approximate analysis of the temporal behaviour of an MRSDF graph. Our approach starts by revisiting the construction of HSDF graphs, as used in exact analysis. We provide a mathematical basis for these HSDF graphs, by formulating the timing constraints that govern the MRSDF graph's schedule in *max-plus algebra* (Section V).

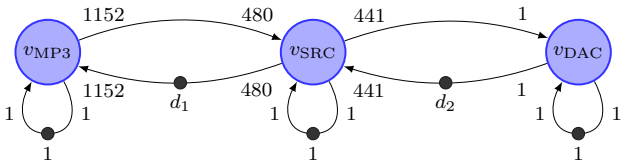
From this mathematical characterisation, we show how *both* a conservative *and an optimistic* approximation naturally follow, and how these approximations may again be represented by (homogeneous) SDF graphs (Section VI). The conservative and optimistic approximations give bounds on the throughput of the modelled application, or may be used to derive lower and upper bounds on buffer sizes required to reach a minimum throughput (Section VIII).

In summary, the main contributions, both theoretical and practical of this paper are:

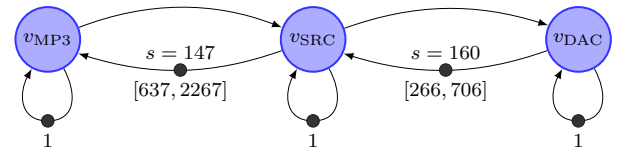
- A mathematical formulation of the temporal behaviour of an MRSDF graph, which provides a solid basis for understanding the exact analysis of MRSDF graphs.
- Six different ways to transform a multi-rate SDF graph into a homogeneous SDF graph, two of which are exact, and four of which are approximate, and their purpose for design space exploration.
- An application of the presented approximations to the problem of determining bounds on the buffer sizes required to reach a minimum throughput.

II. Basic Idea

To illustrate the use of the approximating HSDF graphs presented in this paper, we give a brief example. Consider the MRSDF graph model (taken from [5]) of an MP3 playback application, given in Figure 1(a). The 3 actors



(a) MRSDF graph model



(b) Approximate HSDF graph model, with bounds on buffer sizes

Figure 1. An MP3 playback application, modelled as an MRSDF graph, and the result of the approximate analysis presented in this paper.

(from left to right) represent the MP3 decoder, sample-rate converter and digital-to-analog converter (DAC) tasks. The MP3 decoder processes a 48 kHz MP3 file, which is transformed, by the sample-rate converter, into a 44.1 kHz stream to match the frequency of the DAC. The self-loops in this graph model the fact that only a single instance of each of the 3 tasks may run at any given time.

The 3 tasks communicate through FIFO buffers. Since there is no limit on the number of tokens on an edge in an MRSDF graph, modelling such buffers with a single edge would imply these buffers are unbounded. As a common modelling technique, these buffers are therefore modelled by a *pair* of edges, using a backward edge to model back pressure. The number of tokens placed onto such a backward edge is equal to the available space in the FIFO buffer. Variables d_1 and d_2 thus represent the capacities of the buffers between the tasks.

The (worst-case estimates of the) execution times of the MP3 decoder and sample-rate converter are 1603621 and 1320974 clock cycles, respectively. The DAC samples its input every 5000 clock cycles; its execution time is thus set to 5000. The sampling rate of the DAC gives a throughput constraint: the data rate must be such that it keeps up with the DAC.

The approximate analysis that is presented in this paper involves the transformation of the MRSDF graph of Figure 1(a) into both an optimistically and pessimistically approximating HSDF graph. Together, these two graphs give bounds on the required buffer sizes, as is depicted in Figure 1(b). The analysis (see Section VIII for more details) bounds the required buffer size d_1 between 637 and 2267, and d_2 between 266 and 706. This means that the maximum error made in these buffer size estimations is $\frac{2267-637}{637} = 256\%$ and $\frac{706-266}{266} = 165\%$, respectively.

Furthermore, the analysis assigns weights to buffers. These weights are denoted by s in Figure 1(b), and quantify the effect the buffer has on the graph's throughput. The cheapest way to increase the MP3 playback application's throughput is thus by increasing the size of the buffer between the sample-rate converter and the digital-to-analog converter, as the weight of this buffer (160) is higher than the other buffer's weight (147).

A time consuming, exact analysis (using the SDF^3 tool [6]) of the MRSDF graph reveals that, in order to reach the required throughput, d_1 must be at least 1536, and d_2 at least 517. By choosing the upper bounds of the required buffer sizes, we thus overestimate this minimum by factors of 48% (for d_1) and 37% (for d_2).

III. Related Work

Multi-rate synchronous dataflow (MRSDF) graphs, introduced in [1], are closely related to a class of Petri Nets, called *weighted timed event graphs* [7], *timed event graphs with multipliers* [8] or *weighted T-Systems* [9]. Similar approaches to throughput analysis of these graphs, each by means of a transformation into a homogeneous representation (an HSDF graph MRSDF or timed event graph [10]) have been described in [1], [2], [11], [12], and in [13], [14]. Throughput analysis of homogeneous representations (HSDF graphs or timed event graphs) is well-studied, and a number of efficient algorithms are available [15]–[19].

The main problem with an approach that transforms the graph into a homogeneous representation is the worst-case size of the latter, which is exponential in the size of the original graph. Even though the regular structure of an HSDF graph allows for efficient analysis, in which the graph need not be stored explicitly [12], this HSDF-based approach is considered too costly for practical use [3]–[5], [11]. As a result, state-space exploration methods are considered a more efficient way to analyse an MRSDF graph, since it avoids the costly MRSDF-to-HSDF transformation [3]. A disadvantage of the state-space exploration method, however, is that it does not give any deeper mathematical understanding of the analysis of MRSDF graphs.

A different way to derive a homogeneous representation of an MRSDF graph is presented in [20], where the number of actors in the resulting HSDF graph is determined by the total number of tokens in the MRSDF graph. Their transformation requires a symbolic simulation of the MRSDF graph, whereas the transformations described in this paper are computed statically.

A completely different approach to cope with the difficulties posed by throughput analysis of MRSDF graphs is to approximate the graph's throughput rather than performing an exact computation. Several authors [4], [5], [21] deal with the problem of approximating the throughput of an MRSDF graph by constructing an approximate model that may be analysed with much less effort. The model proposed in [4] is based on these approximations. These conservative approximate models have proven to be particularly useful

when determining the minimum buffer capacities necessary to achieve a guaranteed minimum throughput. The existing work on throughput approximation has two main disadvantages. First of all, the models are not represented as dataflow graphs, because the entities in the model have different firing rates. As a result, (approximate) throughput computation is formulated and solved as a linear program. Because the approximating models that we present are HSDF graphs, the outcome of approximate throughput analysis is *more detailed*: apart from bounds on throughput, we obtain the critical cycle of the system, as well.

A second disadvantage of existing methods is that only a *conservative* bound is derived. Although having only a conservative estimate is sufficient when worst-case guarantees must be given, it makes judging the error of the bound prohibitively hard.

We compute both a conservative and an optimistic bound on the throughput of an MRSDF graph and show that the error in the estimation may be considerable even for relatively small graphs. We show how this model follows directly from a max-plus algebraic formulation of an MRSDF graph. We show (in Sect. VI) how conservative, as well as optimistic bounds may be derived from an HSDF graph that approximates the MRSDF graph in terms of performance (i.e., throughput).

Related to the work on throughput approximation is the work done on Homogeneous Timed Event Graphs with Multipliers (TEGMs) in [8], where an optimistic rather than a conservative approximation is obtained, based on a min-plus algebraic formulation of the behaviour of the graph. The resulting graph is similar to one of the four approximating graphs that we derive using a max-plus algebraic formulation, in Section VI-A.

IV. Terminology

We assume that the reader is familiar with standard MRSDF terminology (such as actor, channel, firing, production/consumption rates, etc), we only define a few MRSDF notions that are relevant for this paper.

We denote the production rate of actor u on channel uv by ρ^+_{uv} and the consumption rate of v on channel uv by ρ^-_{uv}. A synchronous dataflow graph in which all rates are equal (i.e., one), is called a *homogeneous* SDF (HSDF) graph. Each channel uv has an initial number of tokens, which we denote by δ_{uv}.

When *firing*, an actor v instantaneously consumes from its inputs (ρ^-_{uv} for every incoming channel uv), waits an *execution time*, τ_v, and then produces onto its outputs (ρ^+_{vw} tokens for every outgoing channel vw). An actor may only fire if each of its incoming channels contains sufficiently many (as specified by the consumption rate) tokens.

A. Consistency and structural invariance

A cycle $C = v_1 \dots v_n$ (with $v_n = v_1$) in an MRSDF graph is *consistent* if the product of all production rates $\rho^+_{v_i v_{i+1}}$ is equal to the product of all consumption rates $\rho^-_{v_i v_{i+1}}$. An MRSDF graph is consistent if every cycle in the graph is consistent. Note that several authors define consistency of an MRSDF graph in terms of existence of the repetition vector (see below) [1], [2], [12], [22].

In a consistent MRSDF graph, two structural graph invariants exist.The first invariant is the graph's repetition vector, which we denote by q. The repetition vector associates each actor v in the graph with an integer q_v, such that, if each actor fires q_v times, the number of tokens produced onto each channel is equal to the number of tokens consumed from it. In other words, the repetition vector satisfies the *balance equations*:

$$q_u \rho^+_{uv} = q_v \rho^-_{uv}, \tag{1}$$

for each channel uv in the MRSDF graph. In Petri Net literature, the repetition vector is sometimes referred to as *T-semiflow* (where 'T' refers to *transitions*) [9].

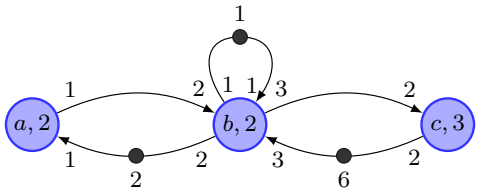
The repetition vector defines a *graph iteration*: if each actor v fires precisely q_v times, we say that one graph iteration is executed. Note that the balance equations are satisfied only *locally*: in a single iteration, the number of tokens that 'flow' through one channel may be different for another channel. The relative average speed at which tokens flow through two subsequent channels uv and vw, is given by the ratio of production rate ρ^+_{vw} and consumption rate ρ^-_{uv}. This gives the second invariant of an MRSDF graph: if we multiply the token flow of each channel uv with an integer normalisation factor s_{uv}, each channel has an equal token flow. The normalisation vector s thus satisfies the *flow conservation equations*:

$$s_{uv} \rho^-_{uv} = s_{vw} \rho^+_{vw}. \tag{2}$$

In Petri Net literature, vector s is called a *P-semiflow* (where 'P' stands for *places*) [9].

Note that for any consistent graph we have $s_{uv}\rho^+_{uv} q_u = s_{uv}\rho^-_{uv} q_v$. In other words, in each iteration of a consistent MRSDF graph, the normalised number of tokens produced onto and consumed from each channel is the same. This quantity is a derived invariant, which we denote by $\mathcal{N} = s_{uv}\rho^+_{uv} q_u$.

An example MRSDF graph is depicted in Fig. 2. The graph contains 2 initial tokens on channel ba, 1 on channel bb, and 6 on channel cb. Each actor is annotated with its execution time: 2 time units for actors a and b, 3 time units for actor c. The graph is consistent: a graph iteration consists of 4 firings of actor a, 2 firings of actor b and 3 firings of actor c. Hence, the repetition vector of the graph is $\langle 4, 2, 3 \rangle$. In order to satisfy the flow conservation equations, we need to apply the following normalisation factors: 3 for channels ab and ba, 2 for channels bc and cb and 6 for channel bb. If we combine the rates, repetition vector and normalisation vector, we find $\mathcal{N} = 12$.



(a) MRSDF graph

Figure 2. An example MRSDF graph with 3 actors and 5 channels.

B. Throughput and cycle time

The *throughput* of a consistent MRSDF graph is commonly defined as the average number of graph iterations that are executed per unit of time. In other words, the throughput $Th(\mathcal{G})$ of an MRSDF graph $\mathcal{G}$ is quantified in terms of the completed number of firings at time t, denoted $F(t)$, and normalised by the graph's repetition vector:

$$Th(\mathcal{G}) = \frac{1}{q_v} \lim_{t\to\infty} \frac{F_v(t)}{t}, \tag{3}$$

for any actor v. The inverse of throughput gives the *cycle time*: the average time required to complete a single iteration.

Alternatively, throughput may be expressed in terms of *normalised token flow*: the throughput of an MRSDF graph is quantified in terms of the number of tokens that passed a channel, normalised by the factor s. If $C_{uv}(t)$ denotes the total number of tokens that have passed channel uv at time t, the graph's throughput $Th(G)$ is defined as follows:

$$Th(G) = \frac{s_{uv}}{\mathcal{N}} \lim_{t\to\infty} \frac{C_{uv}(t)}{t}, \tag{4}$$

for any channel uv.

Throughout this paper, we assume that MRSDF graphs are strongly connected. Note that the throughput of an MRSDF graph can be computed from the individual throughputs of the graph's strongly connected components, in a straightforward way [22].

C. HSDF graphs and max-plus algebra

From a system-theoretic perspective, a synchronous dataflow graph can be regarded as a linear system [10], [23]. The algebra needed to describe such a system mathematically is max-plus algebra (note that, alternatively, min-plus algebra may be used equally well). We give a very brief introduction to max-plus algebra, limited to the details necessary to describe the temporal behaviour of HSDF graphs (we cover MRSDF graphs in Section V). The interested reader may refer to [10], [17], [23].

In max-plus algebra we can express *synchronisation* and *delay* with respectively the operators $\oplus$ and $\otimes$, where $a \oplus b = \max(a, b)$ and $a \otimes b = a + b$. Using these operators, we may describe the temporal behaviour of an HSDF graph as a max-plus system. In the system, we let $t_v(k)$ denote the time at which actor v completes its k^{th} firing. These firing times then satisfy the following, so-called *dater equation*[1]:

$$t_v(k) = \bigoplus_u t_u\left(k - \delta_{uv}\right) \otimes \tau_v. \tag{5}$$

In this equation, δ_{uv} is called the *shift* between v and u [10].

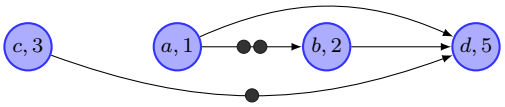


Figure 3. An example, small, fragment of an HSDF graph

As an example, consider the HSDF graph shown in Figure 3, with actors a, b, c and d. The execution of actor d depends on the executions of the other 3 actors, whereas b depends solely on a. The dependencies are formally captured by the following two max-plus equations:

$$t_b(k) = t_a(k-2) \otimes 2$$
$$t_d(k) = (t_a(k) \oplus t_b(k) \oplus t_c(k-1)) \otimes 5.$$

Max-plus systems such as defined by (5), are linear and shift-invariant [10]. In fact, any max-plus system describing an HSDF graph is linear and shift-invariant, and any linear and shift-invariant max-plus system may be graphically represented by an HSDF graph [20].

Linear and shift-invariant max-plus systems have an *eigenvalue*, commonly denoted by λ. The eigenvalue represents the fastest possible speed at which the system can operate: after an initial *transient phase* the system becomes periodic, and every k^{th} firing is equally shifted forward in time compared to the $(k-\sigma)^{\text{th}}$ firing, where σ is referred to as the system's cyclicity (see [10], [17] for details). In max-plus terms, this is formally stated as:

$$t(k_0 + c\sigma) = t(k_0) + c\sigma\lambda. \tag{6}$$

The eigenvalue λ may be computed by finding the *critical cycle* in the HSDF graph: The critical cycle is the simple cycle for which the ratio between the sum of the actor execution times and the tokens on the channels is maximal, where the maximum is taken over all simple cycles in the graph. Formally, if $\mathcal{G}$ is the HSDF graph representation of a max-plus system S, then the eigenvalue of the system satisfies:

$$\lambda(S) = \max_{\mathcal{C}\in\mathcal{G}} \frac{\sum_{uv\in\mathcal{C}} \tau_u}{\sum_{uv\in\mathcal{C}} \delta_{uv}}. \tag{7}$$

The right-hand side of (7) is called the *maximum cycle ratio* (MCR) of $\mathcal{G}$ [17]. There are many efficient algorithms available to compute the MCR of an HSDF graph, in polynomial time [15], [16], [18].

[1]For brevity, in dater equations throughout the paper, we assume that u is restricted to the set of predecessors of v.

V. Exact Homogeneous representations of MRSDF graphs

Temporal analysis of a multi-rate SDF graph requires a transformation of the MRSDF graph into a homogeneous SDF graph as a first step [1], [2]. The resulting HSDF graph may then be analysed for its maximum cycle ratio. The de facto transformation, first described in [1] and implemented in state-of-the-art SDF analysis tools such as [6], differs from the approach taken in this paper. We discuss this difference in Section V-C.

The transformation of an MRSDF graph into an HSDF graph may, from a system-theoretic viewpoint, be understood as the representation of a *shift-variant* system by a *shift-invariant* one [23]. Max-plus equations that describe the temporal behaviour of an MRSDF graph are generally not shift-invariant (although they still are linear, [10]). However, in case the MRSDF graph is *consistent* (which is true for deadlock-free and bounded MRSDF graphs [22]), this shift-variance is periodic. For example, in the graph shown in Figure 2, the number of times actor c can fire after actor b has completed its k^{th} execution, depends on k.

A periodic shift-variant systems allows for a *finite* shift-invariant representation: in Figure 2, the number of times actor c can fire after actor b has completed its k^{th} execution, depends on whether k is even or odd.

Each of the two structural invariants that hold for consistent MRSDF graphs, namely the repetition vector (q) and the normalisation vector (s), identifies a different kind of periodicity. These two kinds lead to two different shift-invariant max-plus systems, and thus HSDF graphs, as is explained in the following two sections.

A. Firings and the Actor perspective

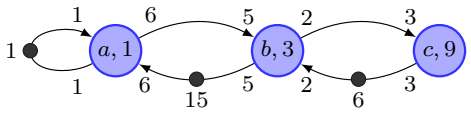
In a synchronous dataflow graph, an actor fires as soon as (we assume a self-timed execution), on each of its incoming channels, the number of tokens as specified by the consumption rate, is available. If we let $t_v(k)$ denote the time at which actor v *completes* its k^{th} execution, then the following equation captures the relation between the firing times of actors connected by a channel:

$$t_v(k) = \bigoplus_u t_u \left(\left\lceil \frac{k\rho^-_{uv} - \delta_{uv}}{\rho^+_{uv}} \right\rceil \right) \otimes \tau_v, \tag{8}$$

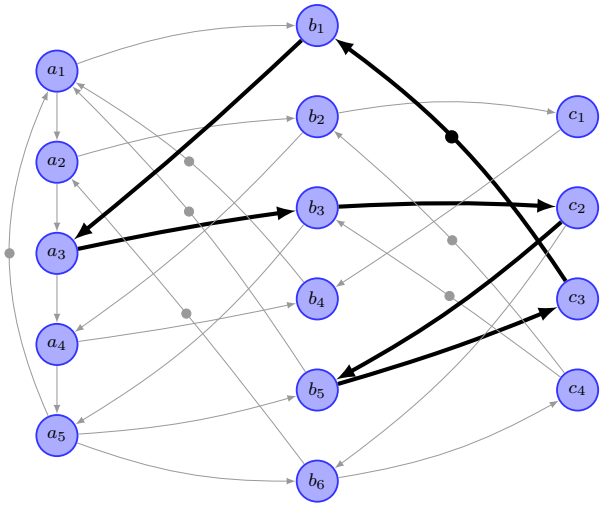
for each actor v in the MRSDF graph.

We call this system of equations (one equation for each actor) the *actor perspective*. It is a periodically shift-variant system; the amount of shift changes periodically with the firing index, k. If we apply the balance equations (1), the amount of shift may be captured modulo q_v, the repetition-vector entry of v:

$$t_v(k + mq_v) = \bigoplus_u t_u \left(\left\lceil \frac{k\rho^-_{uv} - \delta_{uv}}{\rho^+_{uv}} \right\rceil + mq_u \right) \otimes \tau_v. \tag{9}$$



(a) MRSDF graph



(b) Full HSDF graph (actor perspective)

Figure 4. An MRSDF graph (top), and its homogeneous representation (i.e., an HSDF graph) based on the actor perspective. The critical cycle is indicated by thick channels..

By changing the set of variables, i.e. introducing variables v_i with $1 \leq i \leq q_v$, where v_i is related to v such that:

$$t_{v_i}(k) = t_v(i + kq_v),$$

a shift-invariant system is obtained [12]. This shift-invariant system may be graphically represented by an HSDF graph, where a token indicates a shift of *one graph iteration*. This HSDF graph is equivalent to the original MRSDF graph in terms of its throughput. Note that the number of actors in the HSDF graph is equal to the sum of the entries in the repetition vector, which may be exponential in the size of the MRSDF graph [2].

Figure 4 gives an example of such an HSDF graph representation. The critical cycle in the HSDF graph shown in Figure 4(b) is $\mathsf{b}_1\mathsf{a}_3\mathsf{b}_3\mathsf{c}_2\mathsf{b}_5\mathsf{c}_3\mathsf{b}_1$, with a maximum cycle ratio of 28 (and thus a throughput of $1/28$).

B. Tokens and the Channel perspective

The actor perspective describes the interrelationships between actor firing times. We now take another approach and describe, again using max-plus algebra, the interdependencies between the times at which *tokens* are produced onto and consumed from channels. Recall that tokens are produced onto a channel in packets, the size of which is specified by the channel's production rate.

Tokens are produced by firings of the channel's source actor. The precondition for the production of a token onto a

channel is the availability of sufficient tokens, as specified by the consumption rates on the actor's upstream (i.e., incoming) channels. Based on this behavioural specification, we can construct a max-plus model that describes the evolution of token production and consumption times. In this model, we let $t^+_{vw}(k)$ denote the time at which the k^{th} token is produced onto channel vw, and $t^-_{vw}(k)$ denote the time at which the k^{th} token arrives (i.e., is available for consumption) at the input port associated with channel vw. The time at which a token is produced onto channel vw is then related to the token arrival time at upstream channels in the following way:

$$t^+_{vw}(k) = \bigoplus_u t^-_{uv}\left(\rho^-_{uv}\left\lceil\frac{k}{\rho^+_{vw}}\right\rceil\right)\otimes\tau_v, \tag{10a}$$

$$t^-_{vw}(k) = t^+_{vw}\left(k-\delta_{uv}\right). \tag{10b}$$

We call this (shift-variant) system (two equations for each channel) the *channel perspective*. The amount of shift, which is independent of k in (10b), but varies with k in (10a), is again periodically varying. In order to rewrite (10) as a shift-invariant system, we make use of the property that, in each iteration, on channel vw the same number (i.e., $q_v\rho^+_{vw}$) of tokens are produced and consumed and apply the balance equations, (1):

$$t^+_{vw}(k+mq_v\rho^+_{vw}) = \bigoplus_u t^-_{uv}\left(\rho^-_{uv}\left\lceil\frac{k}{\rho^+_{vw}}\right\rceil + mq_u\rho^+_{uv}\right)\otimes\tau_v.$$

In the channel perspective, an HSDF graph representation of an MRSDF graph thus consists of representing each channel vw in the MRSDF graph by actors vw_i, with $1\le i\le q_v\rho^+_{vw}$. In the corresponding max-plus system, the time at which the k^{th} token is produced onto channel vw_i, denoted $t^+_{vw_i}(k)$, is then related to (10) as follows:

$$t^+_{vw_i}(k) = t^+_{vw}\left(i+kq_v\rho^+_{vw}\right).$$

This system of equations may be more compactly represented by having *one* equation per channel, rather than two, in the following way:

$$t_{vw}(k) = \bigoplus_u t_{uv}\left(\rho^-_{uv}\left\lceil\frac{k}{\rho^+_{vw}}\right\rceil - \delta_{uv}\right)\otimes\tau_v.$$

It is clear that the channel perspective results in an HSDF graph that is always larger than the HSDF graph obtained by taking the actor perspective. The HSDF graph that compactly represents the channel perspective of the MRSDF graph depicted in Fig.4, for example, would consist of $30+30+12+12+5=89$ actors. For the sake of compactness, we have chosen to omit this HSDF graph.

C. Relation with existing MRSDF to HSDF transformations

In the introductory paper on synchronous dataflow [1], a transformation from MRSDF into HSDF is presented, which differs from the transformation described above. The transformation given in [1] represents each token that is produced by each firing of an actor during a single iteration, by a single channel in the HSDF graph. The resulting HSDF graph is thus a *multigraph*, which, in terms of equations such as (8) contains a great deal of redundancy. Several authors have adopted the transformation (e.g., [2], [3], [13], [24]) and have presented ways to reduce the multigraph to a simple graph, while maintaining the throughput. We remark that when taking a mathematical approach to modelling the temporal behaviour of an MRSDF graph, such as outlined above, these reductions are unnecessary. Regretfully, the popularity of the transformation as originally presented in [1], the redundancy of which severely degrades the efficiency of MCR analysis algorithms, has distracted research from a deeper mathematical understanding of MRSDF graphs.

One may argue that the transformation outlined in [1] seems natural when one considers an HSDF graph as a graph in which each actor produces precisely one token per firing, rather than a graph representing a shift-invariant max-plus system. We may describe their perspective on the transformation of an MRSDF graph into an equivalent HSDF graph as a "token-based" perspective, as opposed to the actor and channel perspectives presented earlier. However, we firmly believe that, if the purpose of the transformation is to analyse the resulting graph for its temporal behaviour, it is appropriate to model the *times* at which tokens are produced and consumed (or the times at which actors fire), rather than the individual tokens themselves.

VI. Approximate Homogeneous representations of MRSDF graphs

As was described in the previous section, the cost of having an exact HSDF graph representation is the graph's size, which may be exponential in the size of the original graph. If we give up the requirement of having an exact representation, we can obtain HSDF representations that are much smaller. Sections VI-A and VI-B show how an MRSDF graph may be *approximately* described by an HSDF graphs, in different ways, depending on whether the actor or channel perspectives is used as a basis. These different HSDF graphs are equal in terms of the bounds, on the original MRSDF graph's throughput, that they derive, which is proven in Section VI-C.

A. Approximate shift-invariance: Actor perspective

In order to obtain a shift-invariant representation, we must identify the system's shift-invariant properties. The most obvious shift-invariant property of an MRSDF graph is given by the graph iteration: on average, the number of completed firings of an actor v, normalised by q_v, is the same for each actor.

We therefore describe the times at which an actor completes its κ^{th} group of firings, as a max-plus algebraic linear

function of the times at which other actors complete (parts of) their iterations. We denote the time at which actor v completes its κ^{th} group of firings by $\tilde{t}_v(\kappa)$. Note that κ must be a rational number (rather than an integer). We restrict the domain of $\tilde{t}$ to those rationals κ such that $\kappa\mathcal{N} \in \mathbb{Z}$.

Since actors can only execute an integer number of firings, we define each actor-specific function $\tilde{t}_v$ for those rationals κ such that $\kappa q_v \in \mathbb{Z}$ (recall that the repetition-vector entry, q_v, of each actor v, is a divisor of $\mathcal{N}$). In other words, each t_v maps the completion time of individual firings of v to a time, but the firing index is represented as a fraction.

The time at which actor v completes its κ^{th} iteration (i.e., its $(\kappa q_v)^{\text{th}}$ firing) now satisfies:

$$\tilde{t}_v(\kappa) = \bigoplus_u \tilde{t}_u \left(\frac{1}{q_u} \left\lceil \frac{\kappa q_v \rho^-_{uv} - \delta_{uv}}{\rho^+_{uv}} \right\rceil \right) \otimes \tau_v. \tag{11}$$

This set of equations does not represent a shift-invariant system, due to the ceiling function. We may approximate the ceiling function, using either a lower or upper linear bound. A lower linear bound is obtained by simply discarding the ceiling function. To obtain an upper linear bound, we make use of the identity $\lceil \frac{m}{n} \rceil = \lfloor \frac{m+n-1}{n} \rfloor$ (where $m \in \mathbb{Z}$ and $n \in \mathbb{N}$), and obtain:

$$\tilde{t}_v(\kappa) = \bigoplus_u \tilde{t}_u \left(\frac{1}{q_u} \left\lfloor \frac{\kappa q_v \rho^-_{uv} - (\delta_{uv} - \rho^+_{uv}) - 1}{\rho^+_{uv}} \right\rfloor \right) \otimes \tau_v.$$

Using an upper linear bound on the floor operator now gives the following shift-invariant system:

$$\hat{t}_v(\kappa) = \bigoplus_u \hat{t}_u \left(\kappa - \frac{\delta_{uv} - \rho^+_{uv} + 1}{q_u \rho^+_{uv}} \right) \otimes \tau_v,$$

which satisfies the domain restriction of t. Finally, we scale the domain of t by a factor of $\mathcal{N}$ and substitute k for $\kappa\mathcal{N}$ to obtain a discrete-time, shift-invariant system:

$$\hat{t}_v(k) = \bigoplus_u \hat{t}_u \left(k - s_{uv} \left(\delta_{uv} - \rho^+_{uv} + 1 \right) \right) \otimes \tau_v. \tag{12}$$

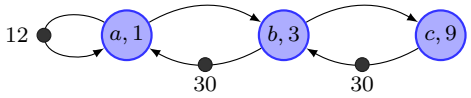
In a similar fashion, by taking a lower linear bound on the ceiling operator in (11), we obtain an optimistic shift-invariant system:

$$\check{t}_v(k) = \bigoplus_u \check{t}_u \left(\kappa - s_{uv} \delta_{uv} \right) \otimes \tau_v. \tag{13}$$

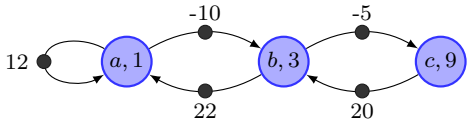
We shall refer to the systems defined by (12) and (13) as *pessimistic* and *optimistic actor-firing systems*, respectively. Note that, due to the scaling of the domain by a factor of $\mathcal{N}$, the eigenvalues of these systems have been scaled by a factor of $\mathcal{N}^{-1}$.

Each of these systems can be graphically represented by an HSDF graph. The HSDF graphs derived from the two firing systems have the same structure as the original MRSDF graph.

As an example, Figure 5 shows how the MRSDF graph depicted in Figure 4 is approximated by the HSDF graph



(a) Optimistic HSDF graph



(b) Pessimistic HSDF graph

Figure 5. The optimistic and pessimistic approximating HSDF graphs based on the actor perspective.

representations of the optimistic and pessimistic actor-firing systems. In the pessimistic HSDF graph, the respective cycle ratios of cycles aa, aba and bcb, are $\frac{1}{12}$, $\frac{1}{3}$ and $\frac{4}{5}$. Cycle bcb is thus the critical cycle, with a maximum cycle ratio of $\frac{4}{5}$. Scaling this by $\mathcal{N} = 60$ gives an upper bound on the MRSDF graph's cycle time of 48.

In the optimistic HSDF graph, cycle bcb is critical as well, with an MCR of $\frac{2}{15}$, which gives a lower bound on the cycle of time of the MRSDF graph of 24. Recall that the graph's exact cycle time, as obtained from the exact HSDF representation (see Figure 4), is equal to 28: the lower bound thus underestimates the exact cycle time by 14%, and the over-approximation is off by 71%.

B. Approximate shift-invariance: Channel perspective

In a similar fashion, an approximate shift-invariant system can be derived from the channel perspective, (10). Similar to the grouping of actor firings into graph iterations in the actor-perspective, in the channel perspective we group tokens into *quanta*. The size of these quanta are chosen such that over time, the average number of quanta that are produced onto each channel, is the same for each channel in the graph. Each channel uv thus has a different integer quantum size, which we denote by Q_{uv}, such that the following holds:

$$\frac{q_v \rho^-_{uv}}{Q_{uv}} = \frac{q_v \rho^+_{vw}}{Q_{vw}}. \tag{14}$$

The definition of Q that satisfies this, is given by:

$$Q_{uv} = \frac{\mathcal{N}}{s_{uv}} \tag{15}$$

This gives the following system, where $\tilde{t}^+_{vw}(\kappa)$ and $\tilde{t}^-_{vw}(\kappa)$ denote the time at which κ token quanta have been produced onto and consumed from vw, respectively:

$$\tilde{t}^+_{vw}(\kappa) = \bigoplus_u \tilde{t}^-_{uv} \left(\frac{\rho^-_{uv}}{Q_{uv}} \left\lceil \frac{k Q_{vw}}{\rho^+_{vw}} \right\rceil \right) \otimes \tau_v \tag{16a}$$

$$\tilde{t}^-_{vw}(\kappa) = \tilde{t}^+_{vw} \left(k - \frac{\delta_{vw}}{Q_{uv}} \right). \tag{16b}$$

In this system, the domain for $\tilde{t}^+$ and $\tilde{t}^-$ is given by those rationals κ such that $\kappa\mathcal{N} \in \mathbb{Z}$. Taking an upper linear bound on the ceiling function and applying (2) (flow conservation) gives:

$$\hat{t}^+_{vw}(\kappa) = \bigoplus_u \hat{t}^-_{uv}\left(k - \left(\frac{1}{Q_{vw}} - \frac{\rho^-_{uv}}{Q_{uv}}\right)\right) \otimes \tau_v.$$

We now multiply the number of tokens with the constant $\mathcal{N}$, and substitute k for $\kappa\mathcal{N}$, to make time discrete:

$$\hat{t}^+_{vw}(k) = \bigoplus_u \hat{t}^-_{uv}\left(k - \left(s_{vw} - s_{uv}\rho^-_{uv}\right)\right) \otimes \tau_v, \tag{17a}$$

$$\hat{t}^-_{vw}(k) = \hat{t}^+_{vw}(k - s_{vw}\delta_{vw}). \tag{17b}$$

The system that is composed of these equations gives a pessimistic description of the flow of token quanta. In a similar fashion, the following optimistic description may be derived by taking a lower linear bound on the ceiling function in (16a):

$$\check{t}^+_{vw}(k) = \bigoplus_u \check{t}^-_{uv}(k) \otimes \tau_v, \tag{18a}$$

$$\check{t}^-_{vw}(k) = \check{t}^+_{vw}(k - s_{vw}\delta_{vw}). \tag{18b}$$

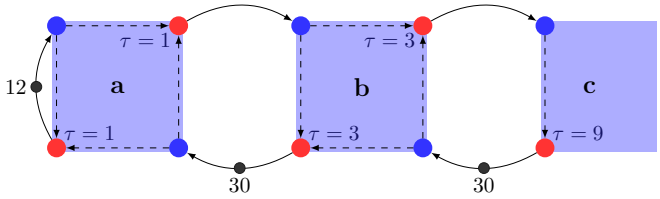
We refer to systems (17) and (18) as the *pessimistic* and *optimistic token-flow system*, respectively. The throughputs of these systems specify the maximum rate at which token quanta can flow through the graph.

As an example, Figure 6 gives the optimistic and pessimistic HSDF versions of the MRSDF graph from Figure 2, based on the channel perspective. Each actor in the HSDF graph represents a single port of the original MRSDF actor. For the sake of compactness, actors in the HSDF graph are drawn as dots; their names and execution time can be derived from the MRSDF actor that they describe.

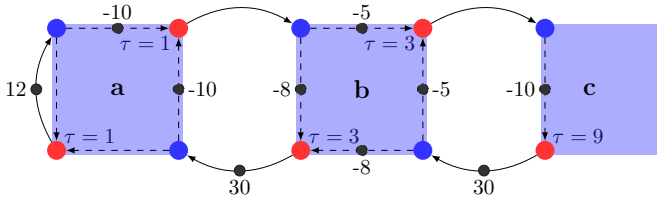
The example HSDF graphs contain more cycles than the HSDF graphs based on the actor perspective, shown in Figure 5. Recall that in both the optimistic and the pessimistic HSDF graphs shown in Figure 5, cycle bcb is critical. In Figure 2, cycle $\mathsf{cb}^+\ \mathsf{cb}^-\ \mathsf{bc}^+\ \mathsf{bc}^-\ \mathsf{cb}^+$ is critical in both HSDF graphs. The bounds on the MRSDF graph's cycle time, derived from these graphs, are 24 and 48, identical to the bounds derived from the HSDF graphs of Figure 5. The next section shows that the bounds derived in the two different perspectives are equivalent for all MRSDF graphs.

C. Equivalence of the two approximations

Each of the four different HSDF graphs presented in the previous two sections gives a bound on the cycle time of the MRSDF graph that they approximate. The two different perspectives result in graphs that have different sizes: the HSDF graphs based on the actor perspective have the same size and structure as the original MRSDF graph, whereas the channel perspective results in larger HSDF graphs. Despite these differences in structure, it can be shown that the bounds



(a) Optimistic HSDF graph



(b) Pessimistic HSDF graph

Figure 6. The optimistic and pessimistic HSDF graph representations of the approximation of MRSDF graph from Figure 4(a), using the channel perspective. Actors that represent an input port are coloured blue, actors that represent an output port are coloured red. Only the output ports have a non-zero execution time, denoted by τ.

on cycle-time, obtained from these two perspectives, are in fact equivalent.

To prove this, we let $\mathcal{G}$ be an arbitrary, consistent MRSDF graph, and let $\mathcal{H}_{Ch}$ and $\mathcal{H}_A$ be the pessimistic HSDF graph representations of $\mathcal{G}$ based on the channel and actor perspective, respectively. We show that these graphs give equivalent bounds on the cycle time of $\mathcal{G}$, by showing that there is a correspondence between cycles of $\mathcal{H}_{Ch}$ and $\mathcal{H}_A$, and that corresponding cycles have the same cycle ratio.

First of all, we simplify the pessimistic token-flow system by substituting $s_{uv}\rho^-_{uv}$ for $s_{vw}\rho^+_{vw}$ (flow conservation, (2)), and eliminating the $\hat{t}^-(\cdot)$ variables. This gives the following system:

$$\hat{t}^+_{vw}(k) = \bigoplus_u \hat{t}^+_{uv}\left(k - s_{vw}\left(\delta_{vw} - \rho^+_{vw} + 1\right)\right) \otimes \tau_v.$$

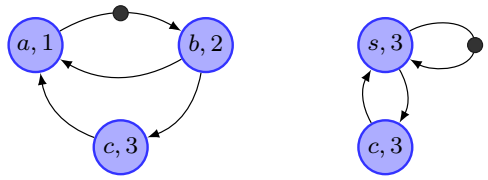
Each edge (uv, vw) in $\mathcal{H}_{Ch}$ maps to an edge uv in $\mathcal{H}_A$. A cycle $\mathcal{C}_{Ch} = v_1v_2, v_2v_3, \ldots, v_{n-1}v_n$, with $v_n = v_1$, thus corresponds to a cycle $\mathcal{C}_A = v_1, v_2, \ldots, v_n$ in $\mathcal{H}_A$. Each cycle $\mathcal{C}_{Ch}$ in $\mathcal{H}_{Ch}$ then has the following cycle ratio:

$$\lambda(\mathcal{C}_{Ch}) = \frac{\sum_{(uv,vw)} \tau_v}{\sum_{(uv,vw)} s_{vw}\left(\delta_{vw} - \rho^+_{vw} + 1\right)}.$$

Thus, corresponding cycles have the same cycle ratio:

$$\lambda(\mathcal{C}_{Ch}) = \lambda(\mathcal{C}_A) = \frac{\sum_{uv} \tau_v}{\sum_{uv} s_{uv}\left(\delta_{uv} - \rho^+_{uv} + 1\right)}.$$

In a similar fashion, it can be shown that the two different optimistic systems give equivalent maximum cycle ratios as well.



(a) Hierarchical HSDF graph (b) Failed composition

Figure 7. An example HSDF graph that is not composable: merging actors a and b into a composite actor, s, introduces deadlock in the graph, as cycle scs has no tokens.

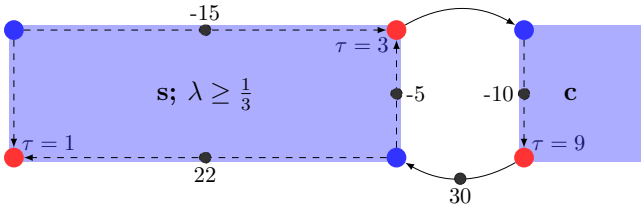


Figure 8. Composability in the channel perspective: composing components a and b in Figure 6(b) yields a new component, s, that has a shortest possible cycle time of $\frac{1}{3}$.

D. Positive Cycles and Deadlock

In the pessimistic HSDF graphs, channels may contain a negative amount of tokens. These negative tokens are a consequence of the fact that an actor may require several tokens to be consumed from one channel, before a single token can be produced onto another channel. When the number of tokens on channels in the original MRSDF graph is relatively low, the negative tokens introduced by the pessimistic approximation may lead to cycles with a negative number of tokens in the pessimistic HSDF graph. These cycles have a negative cycle ratio, which indicates a negative throughput: actors on the cycle can only keep firing if time would run backwards.

Time does not run backwards, and thus cycles with a negative number of tokens indicate possible deadlock. The presence of such cycles is a *necessary condition* for deadlock: if the pessimistic HSDF graph contains no such cycles, the original MRSDF graph is guaranteed to be free of deadlock.

VII. Composability in the Actor and Channel Perspective

When designing complex systems in an incremental fashion, a desirable property is that the analysis of the system may be performed in the same incremental manner. A system that may be analysed in an incremental, or stepwise fashion is called *composable*: the properties of the composition of two subsystems can be derived from the properties of the subsystems considered in isolation.

It is well known that MRSDF graphs are, generally, not composable [25]. That is, given two SDF graphs, it is not generally possible to derive the properties (e.g., liveness, throughput) of the composition of the two graphs from the properties of the individual graphs.

For HSDF graphs, composition fails when the graph is hierarchical. Consider for example the small HSDF graph depicted in Figure 7. The graph is deadlock-free: the two cycles, aba and abca both contain a single token. If we compose actors a and b into one actor, s, the (temporal) behaviour of the graph is changed: cycle csc now contains no tokens and thus the graph is deadlocked.

If we change the level of abstraction at which composition is performed, i.e., specify rules that forbid certain HSDF actor compositions, an HSDF graph may be analysed in a composable manner. The proper level of abstraction to achieve this, is the *component level* [4]. In the component-view, different *ports* that belong to the same MRSDF actor are grouped together. Ports can be connected by internal (between ports of the same component) and external (between ports of different components) connections. A composition of two components involves grouping the ports of the components together in the new super-component. As a result, connections between the involved components become internal connections, which may be hidden [4].

The approximating HSDF graphs derived from the channel perspective in Section VI-B naturally fit such a component-view: each port (the point at which a channel is connected to an actor) of an MRSDF actor is modelled by one HSDF actor, and the channels in the HSDF graph give the connections between the ports. Composition of two components may result in the creation of *internal cycles*: cycles in which every actor is a port of the same component. An internal cycle poses a limit on the maximum speed at which the component may operate; this cycle's cycle ratio gives the shortest possible cycle time (i.e., maximum possible throughput), at which each of the ports that lie on the cycle may produce and consume tokens (see Figure 8). This is composable: the shortest possible cycle time of the composition of two components, is equal to the *maximum* of the two shortest possible cycle times (in other words, one component forms a *bottleneck* for the other).

VIII. Computing Sufficient and Necessary Buffer Sizes

Buffer sizes affect the throughput of a streaming application. Computing the buffer sizes that are required to meet a certain minimal throughput means determining the minimum amount of initial tokens that must be placed on channels. For an MRSDF graph, this problem is known to be NP-Complete [26], and finding an exact solution requires significant computation time, even for relatively simple graphs. We therefore choose to compute bounds on the required buffer sizes, using the approximating HSDF

graphs.

It is well known (see, for example, [10]) that the maximum cycle ratio of an HSDF graph may be computed by solving a linear program. This involves rewriting each max-plus expression,

$$t_v(k) = \bigoplus_u t_u(k - \delta_{uv}) \otimes \tau_v,$$

as a number of greater-than constraints, one for each of the terms in the expression, in the following way:

$$t_v \geq t_u - \delta_{uv}\lambda + \tau_v,$$

for all channels uv in the HSDF graph. The objective of the linear program is set to minimise the graph's cycle time, λ, which is equivalent to computing the graph's maximum cycle ratio.

Instead of computing the maximum cycle ratio, we treat it as a constant (set to the required maximum cycle ratio), and let the buffer sizes be the variables. If we let the objective of the linear program be to minimise the sum of the buffer sizes, the program's solution then gives the buffer sizes that are sufficient to obtain the required maximum cycle ratio.

Since the linear shift-invariant system is an approximation, computing an exact solution to the integer linear programming problem does not make much sense. We therefore solve its linear programming relaxation instead. The (non-integer) solution to the linear program derived from the optimistic approximation, gives buffer sizes that are a lower bound on the required buffer sizes. This lower bound can be considered a necessary buffer size: if we were to choose buffer sizes *smaller* than the lower bound, we are certain that the throughput constraint will not be met. We may obtain an integer solution by taking the *floor* (i.e., round towards negative infinity) of the non-integer solution.

On the other hand, if we solve the pessimistic linear program, the computed buffer sizes are an upper bound on the required sizes, sufficient to attain the required throughput. For the pessimistic approximation, an integer solution is obtained by taking the *ceiling* (i.e., round towards positive infinity) of the non-integer solution.

As an example, we compute buffer sizes for the example MRSDF graph shown in the beginning of this paper, in Figure 1(a). The throughput requirement for this graph stems from the sample rate of the digital-to-analog converter, which must be able to start an execution every 5000 clock cycles. The required cycle time is thus $5000 \times q_{v_{\text{DAC}}}$. We must scale this cycle time down by $\mathcal{N}$= 846720, which gives $\lambda = \frac{125}{4}$.

The "optimistic" linear program for this graph is given by:

$$\begin{aligned}
\textbf{minimize} \quad & d_1 + d_2 \\
\text{s.t.} \quad & t_{v_{\text{MP3}}} \geq t_{v_{\text{SRC}}} - 147d_1\lambda + 1603621 \\
& t_{v_{\text{SRC}}} \geq t_{v_{\text{MP3}}} + 1320974 \\
& t_{v_{\text{SRC}}} \geq t_{v_{\text{DAC}}} - 160d_2\lambda + 1320974 \\
& t_{v_{\text{DAC}}} \geq t_{v_{\text{SRC}}} + 5000 \\
& t_{v_{\text{DAC}}} \geq t_{v_{\text{DAC}}} - \lambda + 5000 \\
& t_{v_{\text{SRC}}} \geq t_{v_{\text{SRC}}} - 70560\lambda + 1320974 \\
& t_{v_{\text{MP3}}} \geq t_{v_{\text{MP3}}} - 1152\lambda + 1603621 \\
& \lambda = \frac{125}{4},
\end{aligned}$$

and the "pessimistic" linear program is equal to the optimistic linear program (and is therefore not listed), except that d_1 is substituted by $(d_1 - 480 + 1)$, and d_2 is substituted by $(d_2 - 441 + 1)$.

The solutions to these two linear programs give the following bounds on the buffer sizes: $636.65 \leq d_1 \leq 2266.65$ and $265.19 \leq d_2 \leq 705.19$. Rounding these bounds gives the integer values as depicted in Figure 1(b).

The bounds on buffer sizes, as obtained by this approach, may lie far apart. As a result, choosing the (pessimistic) upper bound computed from the pessimistic approximation may result in a severe over-dimensioning of the system. In order to guide a designer in further reducing the buffer sizes while meeting the throughput constraint, the normalisation vector indicates which buffer sizes are the cheapest to reduce (namely those with the smallest entries in said vector).

IX. Conclusions and Future Work

In this paper, we have revisited the transformation of a multi-rate synchronous dataflow (MRSDF) graph into a homogeneous synchronous dataflow (HSDF) graph. We have shown that this transformation may be done either exactly, or approximately. We have given a mathematical foundation for these transformations: we have shown how they arise from the linear, shift-variant max-plus systems that describe the behaviour of the MRSDF graph.

While currently, to our best knowledge, only a single transformation is known in the SDF literature, we have presented a total of six transformations. These transformations are grouped into two different perspectives on an MRSDF graph, each of which is related to one of the two structural invariants of the graph. One of these structural invariants, which we have named *normalisation vector*, is unknown in the SDF literature, and quantifies the impact of the size of a specific buffer on the graph's throughput.

Four of the transformations presented in this paper are approximate. The approximation may either be optimistic or pessimistic in terms of throughput. We have shown how these optimistic and pessimistic approximations may be used to compute bounds on the buffer sizes required to reach a minimum throughput.

The two different perspectives give approximating HSDF graphs that differ in their structure: the actor perspective leads to compact, approximating HSDF graphs, whereas the channel perspective gives a slightly larger graph that is composable under (throughput) analysis and thus serves an incremental design of complex systems. Although the optimistic and pessimistic systems differ in size, we have shown that the throughput bounds derived from them are, in fact, equivalent.

Because the approximating systems may be represented as HSDF graphs, existing techniques may be applied to find bounds on the maximum cycle ratio of the approximating HSDF graphs, which gives bounds on the throughput of the MRSDF graph. Furthermore, the presence of cycles with a negative token count in the pessimistically approximating HSDF graph is a necessary condition for deadlock in the original MRSDF graph; absence of such a cycle is therefore sufficient for liveness of the MRSDF graph.

Finally, we remark that the presented approach is not limited to MRSDF graphs, but may be applied to similar graphs, such as weighted timed event graphs. We furthermore plan to generalise the presented approach to more complex dataflow models, such as cyclo-static dataflow [27], and to characterise the error made by the approximations.

REFERENCES

[1] E. Lee and D. Messerschmitt, "Synchronous data flow," *Proceedings of the IEEE*, vol. 75, no. 9, pp. 1235–1245, 1987.

[2] S. Sriram and S. S. Bhattacharyya, *Embedded multiprocessors: Scheduling and synchronization*. CRC press, 2009.

[3] A. H. Ghamarian, M. C. W. Geilen, S. Stuijk, T. Basten, A. J. M. Moonen, M. Bekooij, B. Theelen, and M. Mousavi, "Throughput analysis of synchronous data flow graphs," in *Proceedings of the 6th International Conference on Application of Concurrency to System Design (ACSD)*. IEEE Computer Society Press, Los Alamitos, CA, USA, Jun. 2006, pp. 25–36.

[4] 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 tenth ACM international conference on Embedded software - EMSOFT '12*. New York, New York, USA: ACM Press, Oct. 2012, p. 185.

[5] M. H. Wiggers, "Aperiodic Multiprocessor Scheduling for Real-Time Stream Processing Applications," Ph.D. dissertation, University of Twente, Enschede, The Netherlands, Jun. 2009.

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

[7] O. Marchetti and A. Munier-Kordon, "Complexity results for Weighted Timed Event Graphs," *Discrete Optimization*, vol. 7, no. 3, pp. 166–180, Aug. 2010.

[8] G. Cohen, S. Gaubert, and J. Quadrat, "Timed-event graphs with multipliers and homogeneous min-plus systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 9, pp. 1296–1302, 1998.

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

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

[11] M. Geilen, "Reduction techniques for synchronous dataflow graphs," *Annual ACM IEEE Design Automation Conference*, pp. 911–916, 2009.

[12] R. de Groote, J. Kuper, H. Broersma, and G. J. Smit, "Max-Plus Algebraic Throughput Analysis of Synchronous Dataflow Graphs," in *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*. IEEE, Sep. 2012, pp. 29–38.

[13] K. Ito and K. K. Parhi, "Determining the minimum iteration period of an algorithm," *Journal of VLSI Signal Processing*, vol. 11, no. 3, pp. 229–244, Dec. 1995.

[14] M. Nakamura and M. Silva, "Cycle time computation in deterministically timed weighted marked graphs," in *1999 7th IEEE International Conference on Emerging Technologies and Factory Automation. Proceedings ETFA '99 (Cat. No.99TH8467)*, vol. 2. IEEE, pp. 1037–1046.

[15] A. Dasdan, "Experimental analysis of the fastest optimum cycle ratio and mean algorithms," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 9, no. 4, p. 385, 2004.

[16] J. Cochet-terrasson, G. Cohen, S. Gaubert, M. M. Gettrick, and J.-P. Quadrat, "Numerical Computation of Spectral Elements in Max-Plus Algebra," in *Proceedings of the IFAC Conference on System Structure and Control*, Jul. 1998.

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

[18] N. Young, R. Tarjan, and J. Orlin, "Faster Parametric Shortest Path and Minimum Balance Algorithms," *ArXiv Computer Science e-prints*, May 2002.

[19] R. Karp, "Parametric shortest path algorithms with an application to cyclic staffing," *Discrete Applied Mathematics*, vol. 3, no. 1, pp. 37–45, Feb. 1981.

[20] M. Geilen, "Synchronous dataflow scenarios," *ACM Transactions on Embedded Computing Systems*, vol. 10, no. 2, pp. 1–31, Dec. 2010.

[21] M. Geilen, S. Tripakis, and M. Wiggers, "The earlier the better," in *Proceedings of the 14th international conference on Hybrid systems: computation and control - HSCC '11*. New York, New York, USA: ACM Press, Apr. 2011, p. 23.

[22] A. Ghamarian, M. Geilen, T. Basten, B. Theelen, M. Mousavi, and S. Stuijk, "Liveness and boundedness of synchronous data flow graphs," *FMCAD06*, no. August, pp. 68–75, 2006.

[23] G. Cohen, S. Gaubert, and J.-P. Quadrat, "Max-plus algebra and system theory: Where we are and where to go now," *Annual Reviews in Control*, vol. 23, pp. 207–219, Jan. 1999.

[24] J. Pino, S. Bhattacharyya, and E. Lee, "A hierarchical multiprocessor scheduling system for DSP applications," in *Conference Record of The Twenty-Ninth Asilomar Conference on Signals, Systems and Computers*, vol. 1. IEEE Comput. Soc. Press, pp. 122–126.

[25] S. Tripakis, D. Bui, B. Rodiers, and E. A. Lee, "Compositionality in Synchronous Data Flow," in *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems - ICCPS '10*. New York, New York, USA: ACM Press, Apr. 2010, p. 199.

[26] S. Stuijk, M. Geilen, and T. Basten, "Exploring trade-offs in buffer requirements and throughput constraints for synchronous dataflow graphs," in *Proceedings of the 43rd annual conference on Design automation - DAC '06*. New York, New York, USA: ACM Press, Jul. 2006, p. 899.

[27] G. Bilsen, M. Engels, R. Lauwereins, and J. Peperstraete, "Cyclo-static data flow," in *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, vol. 5, 1995, pp. 3255–3258 vol.5.