Performance Analysis of Weakly-Consistent Scenario-Aware Dataflow Graphs

Marc Geilen[1], Joachim Falk[2], Christian Haubelt[3], Twan Basten[1,4], Bart Theelen[4] and Sander Stuijk[1]

[1] Electrical Engineering
Eindhoven Univ. of Technology
Eindhoven, The Netherlands

[2] Computer Science
Univ. of Erlangen-Nuremberg
Erlangen, Germany

[3] Comp. Science and Electr. Eng.
Univ. of Rostock
Rostock, Germany

[4] Embedded Systems Innovation
by TNO
Eindhoven, The Netherlands

***Abstract*—The timed dataflow model of computation is a useful performance analysis tool for Electronic System Level Design automation and embedded software synthesis. Its determinism gives it strong analysability properties. It is expressive enough to cover a large class of applications and platforms. The trend however, in both embedded applications and their platforms is to become more dynamic, reaching the limits of what the model can express and analyse with tight performance guarantees. Scenario-aware dataflow (SADF) allows more dynamism to be expressed, introducing a controlled amount of non-determinism into the model to represent different scenarios of behaviour. We investigate so-called weakly consistent graphs in which the scenario changes are not tightly coupled with periods of repetitive behaviour of the static dataflow behaviour in scenarios as in previous methods. We define the semantics of such graphs in terms of $(\max, +)$-algebra and we introduce a method to analyse throughput using a generalisation of $(\max, +)$-automata. An extended version of this paper can be found in [15].**

***Index Terms*—performance analysis, synchronous dataflow, $(\max, +)$-algebra**

I. Introduction

To develop concurrent embedded software applications and the platforms on which they execute, it is important to be able to efficiently assess whether or not performance requirements will be met. The parallel tasks and resource arbitrations create synchronization dependencies between tasks and time delays for processing arbitration. Such behavior is captured well by performance models that build upon the $(\max, +)$-semi-ring [1], such as Network Calculus [5], Real-Time Calculus [27], timed Petri-nets [8], [22], max-plus automata [13], and the timed dataflow models. An important feature of timed dataflow models for performance analysis is their determinacy. In the most restricted dataflow models, such as timed Synchronous Dataflow (SDF) [19], [23], dependencies must be independent of input data, while some more dynamic models (for instance Dynamic Dataflow [7]) allow such dependencies. The growing challenge is that the static structures of data dependencies and regular execution times with limited variation are becoming more and more exceptional as both applications and platforms are becoming more dynamic. Applications are becoming more dynamic, for instance, because of complex data reduction schemes, which introduce strong data-content dependencies. Hand-held devices need to support a wide range and diversity of communication protocols. More and more is handled in software by software-defined radio implementations. For MP3 compression, different parts of the audio, called frames, may be encoded using different methods. These methods cannot be accurately captured in a static dataflow model. Besides the application, also the platforms are becoming more dynamic. They need to dynamically handle various use-case scenarios of applications and use dynamic QoS management to match available resources with applications.

To deal with the increasing amounts of dynamic behaviour in applications and platforms, there is a growing need for performance models that can deal with more dynamic behaviour and can still provide tight performance guarantees. The Scenario-Aware Dataflow (SADF) timed dataflow model [14], [16], [24], [26] tries to maintain as much as possible of the determinacy of dataflow behaviour, while introducing the possibility for non-deterministic variations in the form of *scenarios*. In MP3 decoding for instance, there are five individual coding schemes for audio frames. Each of these schemes can be represented accurately by a static dataflow graph, while the types of frames may occur non-deterministically in arbitrary orders. The SADF model and analysis techniques exploit the determinacy in behaviour within a single scenario, while allowing for non-deterministic selection of the scenarios that occur. A crucial aspect is the concurrency among scenarios. Concurrent implementations of streaming applications are often pipelined. For the MP3 decoder this means that different frames in different scenarios may simultaneously be decoded. Yet, the analysis of scenario behaviours can be separately and sequentially handled, despite their overlap in time.

Analysis methods for the Synchronous Dataflow model, based on spectral analysis techniques in the linear algebra on the $(\max, +)$-semi-ring have been introduced [14], [16] to the analysis of a particular subset of SADF models that are called *strongly consistent*, which means that every individual scenario behaviour corresponds to a complete iteration of an SDF graph. For the analysis of the combination of non-deterministic sequences of scenarios which are modelled as SDF behaviour, the theory of $(\max, +)$-automata [13] has been used [16].

The contribution of this paper is to generalise the performance analysis approach of [16] to the case of [26], where scenarios may occur at a finer granularity than complete SDF iterations. This class is called *weakly consistent* SADFs, as opposed to the *strongly consistent* case, in which every

This work was supported in part by projects CA505 BENEFIC

scenario corresponds to a full iteration. For weakly consistent graphs, this is not necessarily the case, although in the long run, they need still be consistent to guarantee boundedness and deadlock freedom. This generalization is important because it allows us to use non-determinism to model dataflow graphs that are not globally synchronous. This is observed for instance in the MP3 example in this paper where the file reading front-end operates asynchronously from the sound decoding back-end. We introduce methods to determine the worst-case throughput of a weakly consistent SADF and a compact state-space from which latency type of properties can be determined. The generalization also makes the FSM based SADF model a proper generalisation of the CSDF [4] model.

II. Related Work

Dataflow models of computation range from very static through more dynamic and (partially analysable) models, to very dynamic, but also very hard to analyse models [24]. The static models include (homogeneous, cyclo-static) synchronous dataflow [4], [20], [23] and Computation Graphs [18]. Heterochronous Dataflow (HDF) [17] introduces dynamism by combining a finite state automaton with synchronous dataflow graphs in the individual states. The model is restricted to executing complete iterations per state transition of the automaton and the model does not have a timed version for performance analysis. Parameterised Synchronous Dataflow (PSDF) [2] considers a static structure of a dataflow graph, where one or more of the port rates are parameters. It is possible to find parameterized schedules and appropriate buffer sizes, but the possibilities for expressing dynamism are limited. In the variable rate dataflow model (VRDF) [29] communication rates may vary arbitrarily and are not necessarily constant over a complete iteration. Analysis methods for this model are restricted to (conservative) buffer sizing under throughput constraints. More dynamic variations on dataflow models have been defined, but they introduce serious difficulties in the analysis. Examples include Dynamic dataflow (DDF) and Boolean Dataflow (BDF) [6], which are models with data dependent firing rules. Their buffer sizing and throughput analysis problems are undecidable.

An appropriate semantic domain for timed synchronous dataflow behaviour is $(\max, +)$-linear algebra [1]. Spectral analysis in this linear algebra is intimately related to throughput and latency analysis. $(\max, +)$-automata [13] combine $(\max, +)$-linear behaviour with non-deterministic choice. We use this combination to model scenario non-determinism.

We show how our analysis problem is ultimately mapped on a Maximum Cycle Ratio (MCR) problem on a directed multigraph, derived from a $(\max, +)$-automaton. A generalisation of cycle ratio analysis is provided by spectral analysis of $(\max, +)$-linear systems [9], [11]. Spectral analysis gives not only the cycle mean (eigenvalue), but also an eigenvector, which relates to the relative firing times of actors, or latency. A good overview and comparison of cycle mean, cycle ratio and spectral analysis methods can be found in [10].

In this paper we use synchronous dataflow graphs in the individual scenarios. However, we do not consider only complete iterations [16], [17], but allow partial repetition vectors. A special case of grouping firings results from clustering of SDF actors [3], [12], [21]. The result can be a quasi-statically scheduled system, in which the clustered actors can be modelled as scenarios of a weakly-consistent SADF. Hence, the proposed analysis can also be applied to such systems. Another work exploring this aspect is [28], which considers a modular implementation of SDF, where firings of an SDF iteration are grouped together. These may be individually scheduled depending on the presence of input data.

III. Preliminaries

We assume the reader is familiar with SDF and its $(\max, +)$-algebra semantics. A more detailed introduction to the preliminaries can be found in [15].

A. Scenario-Aware Dataflow Graphs

Scenario Aware Dataflow graphs [26] are a variant of dataflow models that try to occupy a sweet spot in the trade-off between analysability and expressiveness [24], in particular to express more dynamic behaviour. It combines Synchronous Dataflow behaviour with finite state-automata. It allows the FSM transitions to occur not only at the borders of complete iterations of the SDF behaviours, but also at intermediary stages. An important element of the timed model is that even though the FSM transitions occur in-between pieces of deterministic dataflow behavior, this does *not* mean that such pieces cannot overlap in time. They can be pipelined.

An important strength of the (timed) synchronous dataflow model is its determinism. An important goal of the SADF model is to avoid loss of the benefits of the deterministic behaviour within scenarios for efficient analysis.

The semantics of SADF can be captured by a combination of classical FSM semantics and $(\max, +)$-based semantics of the scenarios of determinate synchronous dataflow behaviour. The combination of state machines and $(\max, +)$-matrix multiplication is called a $(\max, +)$-automaton and is briefly introduced in the next subsection.

B. $(\max, +)$-automata

A $(\max, +)$-automaton [13], is a tuple $\mathcal{A} = (\Sigma, \mathbf{M}, \mathcal{M})$, of a finite set Σ of scenarios, a mapping $\mathbf{M}$, which assigns to every scenario $\sigma \in \Sigma$ a $(\max, +)$-matrix $\mathbf{M}(\sigma)$ and a morphism $\mathcal{M}$ on finite sequences of scenarios, mapping such sequences to a $(\max, +)$-matrix such that

$$\mathcal{M}(\sigma_1 \dots \sigma_k) = \mathbf{M}(\sigma_k) \dots \mathbf{M}(\sigma_1).$$

For a given sequence of scenarios, the automaton defines the completion time as follows:

$$\mathcal{A}(\sigma_1 \dots \sigma_k) = ||\mathcal{M}(\sigma_1 \dots \sigma_k)\mathbf{0}|| = ||\mathbf{M}(\sigma_k) \dots \mathbf{M}(\sigma_1)\mathbf{0}||.$$

Then, $\mathcal{M}(\bar{\sigma})\mathbf{0}$ captures the production times of the tokens of the SADF after the sequence $\bar{\sigma}$ of scenarios. The time when the final token is produced is captured by taking the $(\max, +)$-norm (maximum entry) of the resulting vector. We are often interested in the worst-case throughput for any possible sequence of scenarios. Gaubert shows [13] how this maximum growth rate (minimum throughput) can be computed as the maximum cycle mean of the equivalent timed event

graph [1] of the matrix $\mathbf{M} = \max_{\sigma\in\Sigma} \mathbf{M}(\sigma)$. It also shows how, given an infinite regular sub language of Σ^*, the set of all finite scenario sequences, the maximum growth rate can be determined using a classical product automaton construction.

For this paper, we will need to generalize this concept. In particular, instead of studying the average growth rate per *step* of the automaton, we will study the ratio of the growth rate relative to another quantity expressed as the sum of a certain benefit or *reward* per scenario. This amounts to application of the *generalised spectral problem* [9] to $(\max, +)$-automata. In this case, the worst-case throughput can be determined as an MCR of the automaton where edges have two labellings, delays and rewards. We also associate non-square matrices $\mathbf{M}(\sigma)$ with scenarios. We assume that we use a specification of legal scenario sequences that is consistent with the matrix sizes, i.e., such that the morphism $\mathcal{M}$ is well-defined.

IV. A semantic model of weakly consistent SADF

In strongly consistent SADF, every transition of the FSM corresponds to a *full iteration* of the SDF graph for the particular scenario. It is therefore a piece of behaviour that can be repeated forever. Moreover, switches between scenarios are possible in such states, because the initial states are identical. *State* in this case refers to the tokens present in the graph, as the graph's actors and channels may be different in different scenarios, but the initial tokens are found in all scenarios. Specifically 'state' of those tokens refers to time stamps indicating the time of their availability to be used (consumed).

We generalize the model to allow for edges of the FSM to correspond to arbitrary (fixed) collections of firings. In contrast with the strongly consistent case, the starting and ending state of a scenario graph are not necessarily the same. Such weakly consistent graphs can still be fully and precisely characterised by a $(\max, +)$-matrix, which is not necessarily square.

V. Example

We establish the $(\max, +)$-automaton model of the running example graph shown in Fig. 1. The initial state is k with an edge to itself labelled with the scenario α, which has one firing of P and one firing of Q in 'mode' a. This combination of firings has no net effect on the distribution of the (three) tokens. The starting state is defined by the two tokens 1 and 2 in the figure, with time stamps t_1 and t_2. This is captured in a $(\max, +)$-vector $[t_1 \;\; t_2]^T$. P needs to fire before Q_a (a firing of actor Q in mode a, in which it takes 2 time units to execute and produces 0 tokens on the edge from Q to R) and consumes both tokens. Hence, its earliest starting time is $\max(t_1, t_2)$. The firing takes 2 time units and completes at time $\max(t_1, t_2) + 2$, which in $(\max, +)$-sum-of-product form is equal to $\max(t_1+2, t_2+2)$, or in vector inner-product notation: $[2 \;\; 2] \cdot [t_1 \;\; t_2]^T$. This is the time stamp of the new token produced at position 1. Next, Q_a fires and consumes the token just produced by P on the edge from P to Q. Its firing takes also 2 units of time and completes at $\max(t_1+4, t_2+4)$, or: $[4 \;\; 4] \cdot [t_1 \;\; t_2]^T$. At this time the token at position 2 is reproduced. Combining the two symbolic states into a matrix-vector equation, we get the following relation between the starting state vector and the end state vector.

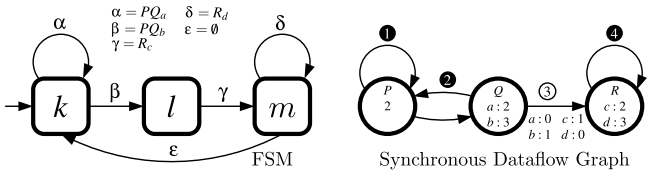

Fig. 1. An example weakly-consistent SADF.

$$\begin{bmatrix} t'_1 \\ t'_2 \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 4 & 4 \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

Another edge, from state k to state l, is labeled with scenario β consisting of the firings $\{P, Q_b\}$. This produces an additional token on position 3 on the edge from Q to R. We obtain the following matrix vector equation.

$$\begin{bmatrix} t'_1 \\ t'_2 \\ t'_3 \\ t'_4 \end{bmatrix} = \begin{bmatrix} 2 & 2 & -\infty \\ 5 & 5 & -\infty \\ 5 & 5 & -\infty \\ -\infty & -\infty & 0 \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \\ t_4 \end{bmatrix}$$

After this, a firing R_c will take place in scenario γ, moving from state l to m. This consumes tokens 3 and 4 and produces token 4, according to $t'_4 = [2 \; 2] \begin{bmatrix} t_3 \\ t_4 \end{bmatrix}$. Token 3 disappears in this process and tokens 1 and 2 remain untouched.

From state m, an arbitrary number of scenarios 'δ' are possible, a single firing of R_d, each of which involves only token 4, according to $t'_4 = t_4 + 3$.

At some point[1], a transition ϵ is taken, back to state k. It is labelled with an empty set of firings and therefore leaves all tokens at rest. The matrix representation is an identity matrix.

VI. Model and Semantics

A weakly consistent SADF graph is defined by a tuple $(\Sigma, G, \boldsymbol{\rho}, i, f, \pi, \mathcal{A})$. It has a finite set Σ of scenarios and every scenario $\sigma \in \Sigma$ has an associated SDF graph $G(\sigma)$ and a partial repetition vector $\boldsymbol{\rho}(\sigma)$, which maps every actor of $G(\sigma)$ to a non-negative number specifying how often the actor fires in the scenario. The graph $G(\sigma)$ has a collection of $i(\sigma) \in \mathbb{N}$ initial tokens, which are indexed $0 \leq n < i(\sigma)$. After execution of the partial repetition vector, the graph $G(\sigma)$ has a collection of $f(\sigma) \in \mathbb{N}$ 'final' tokens, which are indexed $0 \leq n < f(\sigma)$. We use the $(\max, +)$-semantics of the SDF graphs [1] to associate with every graph $G(\sigma)$, a $(\max, +)$-matrix $\mathbf{M}(G(\sigma)) \in (\mathbb{R}^{-\infty})^{f(\sigma)\times i(\sigma)}$, or in short $\mathbf{M}(\sigma)$. The FSM $\mathcal{A}$ is a tuple (Q, q_0, δ) with a set Q of states, an initial state q_0 and a labelled transition relation $\delta \subseteq Q \times \Sigma \times Q$. The scenario labels in the edges must be consistent in the sense that for any state $q \in Q$, any incoming edge labelled with scenario σ_1 and outgoing edge labelled with scenario σ_2, $f(\sigma_1) = i(\sigma_2)$. We denote this number of tokens for a state q: $n(q)$. $\mathcal{A}$ accepts the sequence $\bar{\sigma}$ of scenarios if and only

[1]Note that we could add Büchi acceptance conditions to the automaton to enforce that progress is made eventually. However, such requirements typically have no impact on the worst-case performance.

if $\bar{\sigma}$ is in the language $\mathcal{L}(\mathcal{A})$ of the FSM, i.e., there exists a sequence $\bar{q}$ of states such that $\bar{q}(0) = q_0$ and for every $n \geq 0$, there exists an edge $(\bar{q}(n), \bar{\sigma}(n), \bar{q}(n+1)) \in \delta$.

With sequence $\bar{\sigma}$, we associate the timing behaviour, a sequence of $(\max, +)$-vectors, such that $\boldsymbol{t}_0 = \mathbf{0}$ and for all $n \geq 0$, $\boldsymbol{t}_{n+1} = \mathbf{M}(\bar{\sigma}(n))\boldsymbol{t}_n$. We can now clearly recognise the structure of a $(\max, +)$-automaton.

For synchronous dataflow analysis, it is common to quantify throughput by measuring the number of iterations per time unit. In our case, it depends on the model how much actual, 'real-world' progress is made per scenario. We therefore assume that we explicitly quantify the amount of progress per scenario. For instance, for the example graph, we may be primarily interested in the number of firings of actor R. In this case the progress is 1 for scenarios γ and δ and 0 for any other scenario. In general, we define a *reward* function $\pi : \Sigma \to \mathbb{R}^{\geq 0}$, which quantifies the amount of progress per scenario σ as $\pi(\sigma)$. The throughput obtained from a scenario sequence $\bar{\sigma}$ can hence be defined as follows.

$$\tau(\bar{\sigma}) = \lim_{k \to \infty} \sup \frac{\sum_{n=0}^{k-1} \pi(\bar{\sigma}(n))}{||\boldsymbol{t}_k||}$$

I.e., throughput is defined as the average amount of progress made per unit of time. The primary analysis question we answer in this paper is to determine the worst-case throughput of an SADF graph:

$$\tau = \inf_{\bar{\sigma} \in \mathcal{L}(\mathcal{A})} \tau(\bar{\sigma}).$$

We can define an explicit state space semantics of the model. The states of the state-space consist of pairs $(q, \boldsymbol{t})$ consisting of a state $q \in Q$ of the FSM and a *normalized* vector $\boldsymbol{t}$. The initial state is $(q_0, \mathbf{0})$. The transitions of the state-space are constructed as follows. For a state $(q, \boldsymbol{t})$, consider every outgoing edge (q, σ, q') of q in the FSM. Then the state-space has a labelled transition

$$((q, \boldsymbol{t}), ||\boldsymbol{u}|| - ||\boldsymbol{t}||, \pi(\sigma), (q', \boldsymbol{u}^{norm}))\,,$$

where $\boldsymbol{u} = \mathbf{M}(\sigma)\boldsymbol{t}$. The transitions are decorated with two labels, the amount of time progress and the progress reward.

VII. Analysis

To determine the infimum of the throughput values for all possible scenario sequences on the graph, we need to find the worst-case scenario sequence. Progress of time is measured as the $(\max, +)$-norm of the state vectors $\boldsymbol{t}_k$, the maximum element of the vector. Since $\boldsymbol{t}_{k+1} = \mathbf{M}(\bar{\sigma}(k))\boldsymbol{t}_k$, every element of $\boldsymbol{t}_{k+1}$ is determined by some element of $\boldsymbol{t}_k$ and offset by the corresponding dependency in the matrix. This element in $\boldsymbol{t}_k$ can in turn be traced back to a single element in $\boldsymbol{t}_{k-1}$ and so forth back to $\boldsymbol{t}_0$. In other words, to study the relation between time progress and scenario sequences, we need not look at complete vectors, but we can concentrate on individual elements (initial / final tokens) and their individual dependencies as expressed by the entries in the matrices.

Fig. 2 shows a structure which encodes these dependencies for the example of Fig. 1. The nodes in this graph represent the initial/final tokens (horizontally) in each of the states of the FSM (vertically). For every edge of the FSM, we take the matrix $\mathbf{M}(\sigma)$, with σ the label of the edge, and for every finite (non $-\infty$) element in the matrix we draw an edge between the corresponding initial/final tokens and label it with the value of that element and with the reward $\pi(\sigma)$ of σ. For clarity we have labeled it with the scenario σ itself in the figure. The precise definition of the $(\max, +)$-automaton corresponding to the SADF graph is as follows.

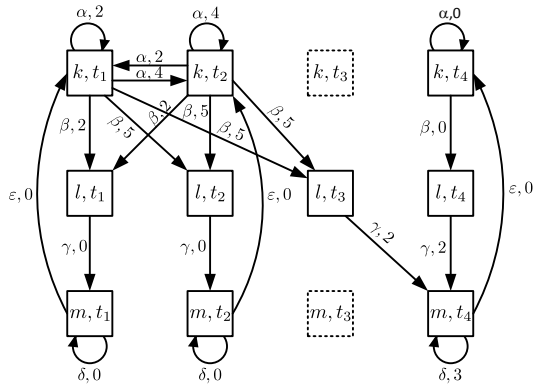

Fig. 2. $(\max, +)$-automaton of the example graph.

Definition 1: For a given SADFG graph $(\Sigma, G, \boldsymbol{\rho}, i, f, \pi, \mathcal{A})$, the analysis $(\max, +)$-automaton is defined in the form of a graph (R, ϵ) with vertices R and edges ϵ, as follows.

- $R = \{(q, i) \mid q \in Q, 1 \leq i \leq n(q)\}$
- $\epsilon = \{((q_1, i), \mathbf{M}(\sigma)_{i,j}, \pi(\sigma), (q_2, j)) \mid (q_1, \sigma, q_2) \in \delta, 1 \leq i \leq n(q_1), 1 \leq j \leq n(q_2)\}$

The worst case throughput of the graph can be determined from a *maximum cycle ratio* analysis of the corresponding $(\max, +)$-automaton, i.e., find the cycle in the graph with the worst-case ratio of total reward over time progress.

Theorem 1: Let $\mathcal{G} = (\Sigma, G, \boldsymbol{\rho}, i, f, \pi, \mathcal{A})$ be an SADF graph and (R, ϵ) be the corresponding $(\max, +)$-automaton graph, then $\inf_{\bar{\sigma} \in \mathcal{L}(\mathcal{A})} \tau(\bar{\sigma}) = \mathrm{MCR}(R, \epsilon)$ is the worst-case throughput of $\mathcal{G}$.

Note that the graph (Fig. 2) has cycles of zero reward (α self-loops) and hence the worst-case throughput is zero. Indeed, actor Q may never produce any output to R in which case, R will never fire. In a refined model we may limit the number of firings of Q in mode a to two, before it must fire in mode b. And similarly we bound the number of firings of actor R in mode d to three. This can be modeled by introducing extra states in the FSM that count the number of firings. The analysis of the refined model is shown in [15]. The worst-case throughput is shown to be $\frac{1}{13}$. Note that the behavior of this example is characteristic for the asynchronous file reader front-end of the MP3 example. After a predictable number of reads it must have data available for decoding.

VIII. Experimental Evaluation

We have implemented the worst-case throughput analysis method in the SDF^3 tool set [25] as an extension to the available scenario-aware dataflow analysis. To get the performance analysis results, a maximum cycle ratio analysis is performed

on the $(\max, +)$-automaton using the algorithm of Young et al. [30]. We have used the tool to analyse an MP3 decoder model with a file reader frond-end [15]. Weak consistency is needed for this model to be able to express the asynchronous operation of the file reader and the decompression of the MP3 decoding. The specification consists of seven dataflow graphs for the five coding scheme scenarios and two additional file processing scenarios. The graphs of the frame decoding scenarios are fairly large (up to 25 actors). The specified FSM has 65 states. The $(\max, +)$-matrices extracted from the scenario dataflow graphs are 3 by 3 matrices, where the rows/columns represent one of the three processors on which the decoder is presumably mapped. The determinate behaviour of the large scenario dataflow graphs, with many firings, can thus be very compactly represented. The $(\max, +)$-automaton that is constructed from the FSM and the matrices has 195 nodes (one for every combination of the three initial tokens and one of the 65 FSM states) and it has 2745 edges. The computation time on a standard PC is around 45ms. As a result we also get a critical scenario sequence from a critical cycle of the MCR analysis.

IX. Conclusion

We introduced an exact analysis method for a class of dynamic dataflow graphs, called weakly consistent scenario-aware dataflow in which the behaviour may non-deterministically vary according to scenarios of behaviour, yet within these scenarios behaviour is deterministic and follows the synchronous dataflow paradigm which provides us with powerful analysis techniques. We have generalized the $(\max, +)$semantics of SADF to allow non-consistent scenario behavior to (a generalisation of) $(\max, +)$-automata and exploit existing spectral analysis techniques in $(\max, +)$-algebra for performance analysis. We have implemented the techniques in a tool for performance analysis of dataflow models and we see that it can effectively analyse a model of an MP3 decoder.

References

[1] F. Baccelli, G. Cohen, G. Olsder, and J.P.Quadrat, *Synchronization and Linearity*. John Wiley & Sons, 1992.

[2] B. Bhattacharya and S. Bhattacharyya, "Parameterized dataflow modeling for DSP systems," *IEEE Trans. Signal Processing*, vol. 49, no. 10, pp. 2408–2421, October 2001.

[3] S. S. Bhattacharyya, P. Murthy, and E. Lee, "APGAN and RPMC: Complementary Heuristics for Translating DSP Block Diagrams into Efficient Software Implementations," *Journal of Design Automation for Embedded Systems*, Jan. 1997.

[4] G. Bilsen, M. Engels, R. Lauwereins, and J. Peperstraete, "Cyclo-static dataflow," *IEEE Trans. Signal Processing*, vol. 44, no. 2, pp. 397–408, February 1996.

[5] J. L. Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, ser. Lecture Notes in Computer Science. Springer, 2003, vol. 2050.

[6] J. Buck and E. A. Lee, *Advanced Topics in Dataflow Computing and Multithreading*. IEEE Computer Society Press, 1994, ch. The Token Flow Model.

[7] J. T. Buck, "A dynamic dataflow model suitable for efficient mixed hardware and software implementations of dsp applications," in *Proc. of the 3rd international workshop on Hardware/software co-design*, 1994, pp. 165–172.

[8] J. Campos, G. Chiola, J. Colom, and M. Silva, "Properties and performance bounds for timed marked graphs," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Trans.*, vol. 39, no. 5, pp. 386 –401, May 1992.

[9] J. Cochet-Terrasson, G. Cohen, S. Gaubert, M. Gettrick, and J.-P. Quadrat, "Numerical computation of spectral elements in max-plus algebra," in *Proc. of the IFAC Conference on System Structure and Control*, Nantes, July 1998.

[10] A. Dasdan, "Experimental analysis of the fastest optimum cycle ratio and mean algorithms," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 9, pp. 385–418, October 2004.

[11] V. Dhingra and S. Gaubert, "How to solve large scale deterministic games with mean payoff by policy iteration," in *Proc. of the 1st international conference on Performance evaluation methodologies and tools*, ser. valuetools '06. New York, NY, USA: ACM, 2006.

[12] J. Falk, J. Keinert, C. Haubelt, J. Teich, and S. Bhattacharyya, "A Generalized Static Data Flow Clustering Algorithm for MPSoC Scheduling of Multimedia Applications," in *EMSOFT'08: Proc. of the 8th ACM international conference on Embedded software*, Oct. 2008.

[13] S. Gaubert, "Performance evaluation of (max, +) automata," *IEEE Trans. Automatic Control*, vol. 40, no. 12, pp. 2014–2025, 1995.

[14] M. Geilen, "Synchronous data flow scenarios," *Trans. Embedded Computing Systems*, vol. 10, no. 2, pp. 16:1–16:31, January 2011.

[15] M. Geilen, J. Falk, C. Haubelt, T. Basten, B. Theelen, and S.Stuijk, "Performance analysis of weakly-consistent scenario-aware dataflow graphs," Eindhoven University of Technology, Tech. Rep. ESR-2011-03, December 2011.

[16] M. Geilen and S. Stuijk, "Worst-case performance analysis of synchronous dataflow scenarios," in *International Conference on Hardware-Software Codesign and System Synthesis, CODES+ISSS 10, Proc., Scottsdale, Az, USA, 24-29 October, 2010*, 2010, pp. 125–134.

[17] A. Girault, B. Lee, and E. Lee, "Hierarchical finite state machines with multiple concurrency models," *IEEE Trans. Computer-aided Design of Integrated Circuits and Systems*, vol. 18, no. 6, pp. 742–760, June 1999.

[18] R. M. Karp and R. E. Miller, "Properties of a model for parallel computations: Determinancy, termination, queueing," *SIAM Jounal of Applied Mathematics*, vol. 14, no. 6, pp. 1390–1411, Nov. 1966.

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

[20] E. A. Lee and E. Matsikoudis, *The Semantics of Dataflow with Firing*. Cambridge University Press, 2007, chapter from "From Semantics to Computer Science: Essays in memory of Gilles Kahn".

[21] S. Meijer, H. Nikolov, and T. Stefanov, "Throughput modeling to evaluate process merging transformations in polyhedral process networks," in *DATE*, 2010, pp. 747–752.

[22] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[23] S. Sriram and S. S. Bhattacharyya, *Embedded Multiprocessors: Scheduling and Synchronization*. New York, NY, USA: Marcel Dekker, Inc., 2009.

[24] S. Stuijk, M. Geilen, B. Theelen, and T. Basten, "Scenario-aware dataflow: Modeling, analysis and implementation of dynamic applications," in *Proc. of International Conference on Embedded Computer Systems (SAMOS), 2011*, 2011, pp. 404–411.

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

[26] B. D. Theelen, M. Geilen, T. Basten, J. Voeten, S. V. Gheorghita, and S. Stuijk, "A scenario-aware data flow model for combined long-run average and worst-case performance analysis," in *MEMOCODE*, 2006, pp. 185–194.

[27] L. Thiele, S. Chakraborty, and M. Naedele, "Real-time calculus for scheduling hard real-time systems," in *International Symposium on Circuits and Systems ISCAS 2000*, vol. 4, Geneva, Switzerland, Mar. 2000, pp. 101–104.

[28] S. Tripakis, D. Bui, M. Geilen, B. Rodiers, and E. A. Lee, "Compositionality in synchronous data flow: Modular code generation from hierarchical sdf graphs," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-52, May 2010.

[29] M. H. Wiggers, M. J. Bekooij, and G. J. Smit, "Buffer capacity computation for throughput constrained streaming applications with data-dependent inter-task communication," in *Proc. the 14th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS'08*, April 2008, pp. 183–194.

[30] N. E. Young, R. Tarjan, and J. Orlin, "Faster parametric shortest path and minimum balance algorithms," *Networks*, vol. 21, no. 2, pp. 205–221, 1991.