J Sign Process Syst (2015) 81:129–133
DOI 10.1007/s11265-015-0977-5

LETTER TO THE EDITOR

Comments on "Minimizing Buffer Requirements Under Rate-Optimal Schedule in Regular Dataflow Networks"

Comments on Paper from R. Govindarajan, Guang R. Gao and Palash Desai Published in *Journal of VLSI Signal Processing*, Volume n. 31, pp 207–229, 2002

José-Inácio Rocha · Octávio Páscoa Dias · Luís Gomes

Received: 3 June 2014 / Accepted: 29 January 2015 / Published online: 21 March 2015
© Springer Science+Business Media New York 2015

1 Introduction

In this paper we propose a new approach that allows one to argue and discuss the results achieved so far by [7], which investigate the minimizing buffer requirements under rate-optimal schedule in Regular Dataflow Networks. Our methodology is devoted to the construction of static schedules for Digital Signal Processing Systems presented by Synchronous Dataflows (SDF) with different rates (known as Multi Rate Dataflows) and to foreseen the total amount of buffer memory of SDFs along with per arc space allocation. In [6], Govindarajan et al. call this type of dataflows Regular Stream Flow Graphs to highlight the nature of the token flow in the arcs. The mapping proposed in [11] can be seen as a natural bridge between the two domains (Petri nets and Dataflows) in investigating the buffer memory requirements, moreover one can also measure the activity duration's on the Petri net side. The limited resource of a system is a natural issue and the claim for their availability during system's activity must be foreseen to preclude exhaustion of resources. Therefore this occurrence inhibits system evolution and blocks it.

J.-I. Rocha (✉)
Escola Superior de Tecnologia de Setúbal, Setúbal, Portugal
e-mail: jose.rocha@estsetubal.ips.pt

O. P. Dias
Universidade Lusíada - Lisboa, Lisboa, Portugal
e-mail: octavio.pdias@gmail.com

L. Gomes
Universidade Nova de Lisboa - FCT, Lisboa, Portugal
e-mail: lugo@fct.unl.pt

The paper is divided in four topics. After a summarized overview of the supporting formalisms being investigated by the authors (dataflows and Petri nets), Section 2.2. addresses the main topics behind the mapping bridge outlined between dataflows and Petri nets, followed by a summarized overview of the tools and computational environments used to develop and simulate the resulting Petri net models. In Section 3., the discussion and results achieved so far are point out and used to identify misjudgment concerning the total buffer requirement for a schedule in Synchronous Dataflows. And finally in Section 4. conclusions are highlighted to emphasize the added contribution about the foreseen memory requirements for SDFs using the proposed approach by Rocha et al. [11].

2 Petri Nets, Dataflows and the Proposed Translation Mapping

In order to explain our mapping strategy gathering Synchronous Dataflows and Petri nets, at first a summarized background of both issues is presented. Afterwards, in Section 2.2 a short presentation about the proposed mapping between Synchronous Dataflows and Petri nets is depicted to illustrate our reasoning about the mistaken and erroneous results reached so far by [7].

2.1 Petri Nets and Dataflows

Petri nets (PN) are a modeling formalism that capture the behavior of concurrent and/or distributed systems, allowing synchronization of processes [3, 9]. Nowadays system practitioners can develop the modeling, analysis and simulation

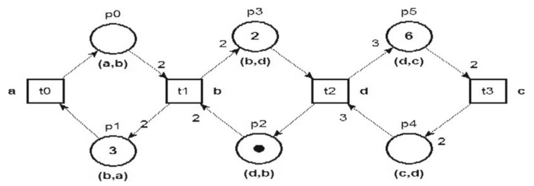


Figure 1 Petri net place-transition equivalent model of example RSFG, from [7], page 210.

of the systems not only supported by the well established mathematical representation of PN but also under graphical environments [2]. In a graphical representation, a Petri net encompasses places, transitions, directed arcs and tokens. Petri nets are bipartite graphs since it is only possible to establish a directed connection (an arc) between a transition and a place or a place and a transition. Places are usually represented by circles or ellipses and model conditions, states, resources or objects. Transitions models events, actions or activities in a Petri net and are denoted by rectangles or boxes. Directed arcs may have attached a natural number, accounting the amount of tokens carried by the arc. An example of Petri net is sketched in Fig. 1. In a dataflow representation [4] a program is composed by two components, vertices known as actors accounting for computations and edges expressing First In First Out (FIFO) queues to hold data values which are embodied in objects known as tokens. Afterwards Lee et al. [8] in 1987 presented Synchronous Dataflows (SDF), a particular set of dataflow where at compile time the number of tokens consumed/produced by an actor on each edge (arc) are known. This way a further improvement about runtime overhead can be achieved. Graphically actors in a dataflow are denoted by circles (squares or rectangles) whereas edges by directed arcs along with nonnegative integer weights posted at its input and output actor ports. These weights express the firing rates (production/consumption) of each actor. Each actor has a fixed number of inputs arcs and a fixed number of output arcs. Figure 2 shows a dataflow network with six actors (**a**, **b**, **c**, **d**, **e** and **f**).

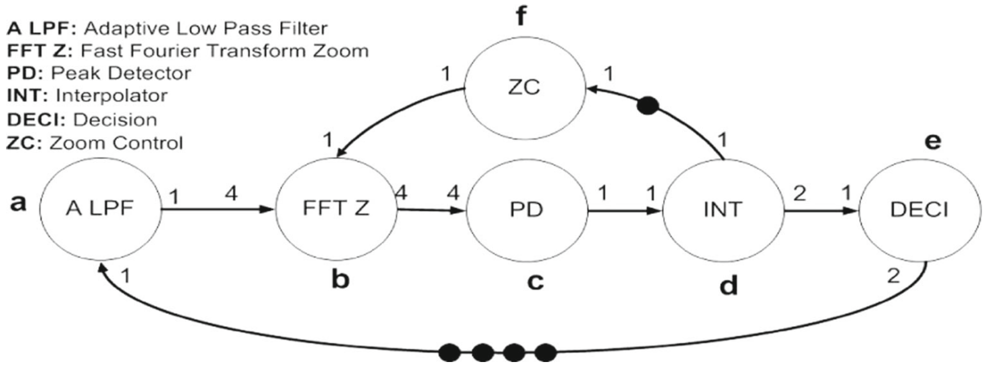
2.2 The Proposed Translation Mapping

In both modeling environments (Petri net and Dataflow) one can find elements with similar nature; elements which are passive and those that embody activity. On the side of Petri nets (Dataflows) one can find places (arcs) and transitions (actors), respectively. Arcs carry tokens representing any kind or type of values. Both environments may have weights and initial markings which have a direct match under our methodolgy, as one can see in Fig. 3. These translation rules were applied by Rocha et al. [13] in multirate systems to foresee the amount of allocated resources as well as to expose the effective and maximum potential number of tokens per arc under a cyclic and continuous flow of data. In this context, after the conversion of a dataflow into a Place/Transition (P/T) Petri net (please refer to [5]), a structural analysis is performed to attain the invariants, commonly known as P(lace)-invariants and T(ransition)-Invariants. With P-Invariants, the required amount of memory can be foreseen, since it is known from [10] that **k**-bounded places in P/T Petri nets can be associated with ***k***-length buffers in Dataflow domain. In a similar way, with T-Invariants one can preview the initial markings, M_0 for each place in P/T domain (arc in Dataflow domain). For a further reading about this mapping please refer to [10]. The development and simulation of the resulting Petri nets equivalent models was performed by two non commercial tools that are available freely in the web: (1) Signal/Net system analyzer SESA [1], a non graphical Petri net editor; (2) Still TINA - TIme Petri Net Analyzer [2], a graphical editor with a simulator to perform a Petri net step by step animation. The invariant analysis (used in Section 3) was performed in both environments.

3 Results and Discussion

The results presented by Govindarajan et al. shows several discrepancies about storage requirements. In page 221, the Fig. 2 represent a Synchronous Dataflow the algorithm

Figure 2 Synchronous dataflow of spectrum analyser.



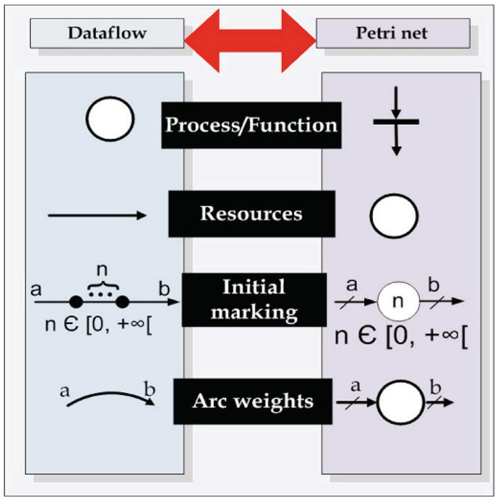


Figure 3 Translation mapping diagram (adapted from [12]).

concerning a Spectrum Analyzer. Starting on page 211, Section 2.3., second line is stated that "*Next we compute the buffer requirement for each arc of the RSFG using an operational model. In this method we compute ...*", but one does not know effectively how these results are achieved, since the meaning of "operational model" is too broad and vague. One may disagree with what authors say in page 212, line 4 about "*Since the prologue is executed only once, separate buffer allocation can be done for it.*" This strategy is time consuming as well as buffer inefficient, since a well behaved dataflow can be planned if one can know in advance at the prologue the optimal initial number of tokens on the arcs which can also accommodate the repetitive pattern of the schedule. Using our methodology on Fig. 2. we obtain a new model on the Petri net side domain depicted on Fig. 4. Likewise for Fig. 2. in [7], page 210, proceeding the same way as before, we can also reach a Petri net model of the corresponding RSFG example (refer to Fig. 1). Since each Petri net is composed by a set of places and transitions one can define a matrix, which is usually called incidence, C, of dimension $m \times n$, with m places and n transitions. The entries of this matrix expresses the token balance associated to the firing of each transition. Once in this domain our focus is on the structural analysis, mainly the properties that remain constant during the execution of the PN. In this context, P-Invariants identifies a group of places where the weighted token sum remains constant for any new marking, thereby establishing region(s) where the allocation of resources does not grow endlessly, insofar with T-Invariants a group of transitions may fire that it does not affect the new marking on the Petri net. This way T-Invariants show their adequacy in the identification of cyclical properties of a Petri net. Using these P-Invariants, v, in the fundamental equation (also known as state equation, since it is possible to predict a new state resulting from a firing sequence vector f),

$$M(p_i) = M_0(p_i) + C \cdot f \tag{1}$$

with C being the incidence matrix, $M(p_i)$ the actual marking after a firing sequence, and $M_0(p_i)$ the initial marking associated to each place p_i, where we pre-multiply it by v^T,

$$v^T \cdot M(p_i) = v^T \cdot M_0(p_i) + v^T \cdot C \cdot f \tag{2}$$

If there exists a $v \in Z^n$ such that $v \neq 0$, then v is a P-Invariant of Eq. 2 if and only if the right most part is null,

$$v^T \cdot C = 0 \tag{3}$$

Considering Petri net model in Fig. 1, performing a P-Invariant analysis using the software tools SESA [1] and TINA [2] yields the following P-Invariant vectors:

$$v = \{(1\ 1\ 0\ 0\ 0\ 0)^T, (0\ 0\ 1\ 1\ 0\ 0)^T, (0\ 0\ 0\ 0\ 1\ 1)^T\} \tag{4}$$

Based on Eq. 3, Eq. 2 reduces to

$$v^T \cdot M(p_i) = v^T \cdot M_0(p_i) \tag{5}$$

Figure 4 Petri net place-transition equivalent model of spectrum analyzer.

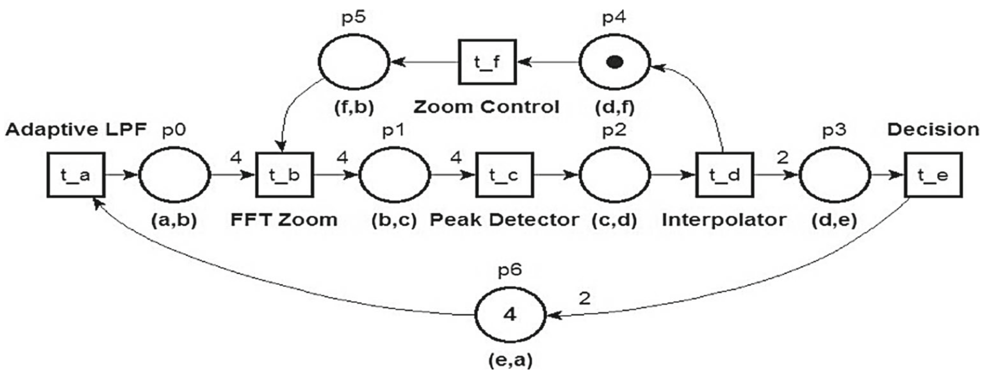


Table 1 Buffer requirements for a schedule of RSFG example in [7], pag. 210.

Buffer Requirement on Arcs						Σ
(a,b)	(b,a)	(c,d)	(d,c)	(d,b)	(b,d)	
2	3	6	6	3	2	22
$M(p_0)$	$M(p_1)$	$M(p_4)$	$M(p_5)$	$M(p_2)$	$M(p_3)$	
3	3	6	6	3	3	24

with $i = \{0, 1, ..., 5\}$, hence the corresponding P-Invariant equations are,

$$M(p_0) + M(p_1) = 3 \tag{6}$$
$$M(p_2) + M(p_3) = 3 \tag{7}$$
$$M(p_4) + M(p_5) = 6 \tag{8}$$

From the previous equations one can achieve the following maximum buffer requirements for each place in the PN model (and arc in the associated dataflow model),

$$M(p_0) = M(p_1) = 3 \tag{9}$$
$$M(p_2) = M(p_3) = 3 \tag{10}$$
$$M(p_4) = M(p_5) = 6 \tag{11}$$

Table 1 presents a summarized view of the memory buffer requirements per arc (place) reached by Govindarajan (first two rows) and (third and fourth row) using the proposed approach. So on page 212, second paragraph, line 3, the total maximum buffer size foreseen in that example of a RSFG is not correct as well as some values associated to a few arcs (refer to Table 1). Therefore instead of "*... A buffer size of 2 is required for the arcs* ***(a,b)*** *and* ***(b,d)****, and a buffer size of 3 is required for* ***(b,a)*** *and* ***(d,b)****. Thus the total buffer requirement for the schedule of Table 1 is 22.*" one should read "*... A buffer size of 3 is required for the arcs* ***(a,b)****,* ***(b,a)****,* ***(d,b)*** *and* ***(b,d)****. Thus the total buffer requirement for the schedule of Table 1 is 24.*".

It is not clear how the buffer requirements for the schedule depicted in Fig. 2 and other signal processing models present by Govindarajan et al. [7] are achieved. We claim that with our methodology (as explained previously) applied to spectrum analyzer shown in Fig. 5 (c) of [7], page 221 by the aforementioned authors also present some incongruence at certain parts. The authors state that arc **(e,a)** in spectrum analyzer requires *three (****3****) units*, but using our approach one claims that this arc demands *four (****4****) units*, therefore the total buffer requirement for a schedule of spectrum analyzer will be *seventeen (****17****) units* instead of *sixteen (****16****) units*. Due to this misjudgment, one should also read on page 222, second column, line 8, "*lastly arc* ***(e,a)*** *requires a buffer of size 4. Thus the total memory requirement for the given schedule, under buffer sharing, is only 9. Thus buffer sharing significantly reduces the buffer requirements from 17 to 9.*" in place of "*lastly arc* ***(e,a)*** *requires a buffer of size 3. Thus the total memory requirement for the given schedule, under buffer sharing, is only 8. Thus buffer sharing significantly reduces the buffer requirements from 16 to 8.''*

4 Conclusions

In this paper we point out misjudgments on the evaluation of buffer requirements based on a new framework supported by mathematical representation of Petri nets. This framework allows one to establish an inter-relation between a Dataflow and a Petri net with the help of a mapping between these two domains. This strategy has proved its reliability in several workout examples, and some signal processing applications during previous years (refer to [12, 13]).

Acknowledgements

This work was partially financed by Instituto Politécnico de Setúbal (IPS) in the framework of projects of "3° Concurso de Projectos do IPS" reference 3CP-IPS-8-2009 and Portuguese Agency "FCT - Fundação para a Ciência e a Tecnologia" in the framework of projects PEst-OE/EEI/UI0066/2011 and PTDC/EEI-AUT/2641/2012.

References

1. SESA - Signal/Net system analyzer, Humboldt - Universita zu Berlin, http://www.vyatkin.org/tools/modelchekers.html. [On-line; accessed June-2014].
2. TINA-TIme Petri Net Analyzer, Laboratoire d'Analyse et d'Architecture des Systèmes. http://homepages.laas.fr/bernard/tina/. [On-line; accessed June-2014].
3. Giraud, C., & Valk, R. (2003). *Petri nets for systems engineering. A guide to modeling, verification, and applications*. Berlin: Springer.
4. Dennis, J. (1974). First version of a data flow procedure language. In Robinet, B. (Ed.) *Programming symposium, Lecture Notes in Computer Science*, (Vol. 19 pp. 362–376).
5. Desel, J., & Reisig, W. (1998). Place/transition petri nets. In Reisig, W., & Rozenberg, G. (Eds.) *Lectures on petri nets I: basic models, lecture notes in computer science*, (Vol. 1491 pp. 122–173). Berlin: Springer.

6. Govindarajan, R., Gao, G.R., & Desai, P. (1994). Minimizing memory requirements in rate-optimal schedules. In *Application specific array processors, 1994. International Conference on Proceedings* (pp. 75–86).
7. Govindarajan, R., Gao, G.R., & Desai, P. (2002). Minimizing buffer requirements under rate-optimal schedule in regular dataflow networks. *The Journal of VLSI Signal Processing*, *31*(3), 207–229.
8. Lee, E.A., & Messerschmitt, D.G. (1987). Synchronous data flow. *Proceedings of the IEEE*, *75*(9), 1235–1245.
9. Murata, T. (1989). Petri nets: properties, analysis and applications. *Proceedings of the IEEE*, *77*(4), 541–580.
10. Rocha, J.I., Gomes, L., & Dias, O. (2011). Dataflow model property verification using Petri net translation techniques. In *Industrial informatics (INDIN), 2011 9th International Conference on IEEE* (pp. 783–788).
11. Rocha, J.I., Gomes, L., & Dias, O. (2011). Petri net verification techniques on synchronous dataflow models. In *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society* (pp. 3792–3797).
12. Rocha, J.I., Gomes, L., & Dias, O. (2012). Analysing storage resources on synchronous dataflows using Petri net verification techniques. In *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, (pp. 4676–4681).
13. Rocha, J.I., Páscoa Dias, O., & Gomes, L. (2013). Exploiting dataflows and Petri nets mappings. In *Industrial Informatics (INDIN), 2013 11th International Conference on IEEE*, (pp. 590–595).

José-Inácio Rocha obtained the degree of bachelor in Electronics and Telecommunications Engineering in 1987 and the degree of Licensed and Master of Electrical and Computer Engineering at Instituto Superior Técnico in 1993 and 2005, respectively. Since 1998, he is professor at School of Technology of Setúbal (ESTSetubal) in Electrical Engineering Department. He is currently working toward the PhD Degree at the Faculty of Science and Technology, Universidade Nova de Lisboa. He is author of scientific papers at international conferences in robotics, renewable energy,co-design of hardware and software, Petri nets and Dataflows. Among the various research areas stands out Embedded Systems, Genetic Algorithms, Signal Processing, Petri nets and Dataflows. In addition, he is with Centro de Tecnologias e Sistemas from Universidade Nova de Lisboa/FCT.

Octávio Páscoa Dias received the graduation and the Master degrees in Electrical and Computer Engineering from the Technical University of Lisbon (1987 and 1995). Received PhD degree in Electrical and Computer Engineering from the Technical University of Lisbon (2000). Presently is Professor at the Lusiada University of Lisbon. Previously he worked as a Coordinator Professor in the Department of Electrical Engineering at the Polytechnic Institute of Setubal (1992-2012). His research topics include Systems Architecture Modelling, Bioelectronics, E-learning, Signal Processing and Control Systems. He has published several papers on these topics (Journals and Intl. Conf. Proc.).

Luís Gomes received his Electrotech. Eng. degree from Universidade Técnica de Lisboa, Lisbon, Portugal, in 1981, and a PhD degree in Digital Systems from Universidade Nova de Lisboa, in 1997. He is a professor at the Electrical Engineering Department, Faculty of Sciences and Technology of Universidade Nova de Lisboa, Portugal and a researcher at UNINOVA Institute, Portugal. From 1984 to 1987, he was with EID, a Portuguese medium enterprise. His main interests include the usage of Petri nets and other concurrency models, applied to reconfigurable and embedded systems co-design. Dr. Gomes is author/co-author of more than 200 papers published in journals, books and conference proceedings. Dr. Gomes has been serving in different roles for the organization of conferences, namely as General Co-Chair or Program Co-Chair for more than 17 IEEE Conferences. He has been serving as Associate Editor for IEEE Transactions on Industrial Electronics (2009-) and for IEEE Transactions on Industrial Informatics (2005–08, 2011-), among other editorial boards.