Dataflow Model Property Verification Using Petri net Translation Techniques

José-Inácio Rocha*†‖§, Luís Gomes†‖‡ and Octávio Páscoa Dias*¶
*Escola Superior de Tecnologia de Setúbal
Rua Vale de Chaves, Estefanilha, Setúbal, Portugal
†Universidade Nova de Lisboa
Faculdade de Ciências e Tecnologia, Campus da FCT/UNL, Portugal
‖UNINOVA-Centro de Tecnologia e Sistemas, Portugal
‡Email: lugo@uninova.pt
§Email: jose.rocha@estsetubal.ips.pt
¶Email: octavio.dias@estsetubal.ips.pt

***Abstract*—Dataflow process networks lead to different theoretical model approaches and have demonstrated their adequacy in data-dominated intensive systems, namely Synchronous Dataflows. Since their appearance, dataflow models became too focused and specialized in their target applications. The paper presents a set of translating mechanisms allowing the mapping from dataflow models into Petri nets. This mapping allows taking advantage of Petri nets well-known properties verification capabilities and enriching dataflow models concerning scheduler information and resource allocation. This allows one to find out some hidden embedded features (model semantics and syntax) not normally addressed in dataflow analysis tools, which is briefly characterized. Dataflow model translation into Petri net domain give support to attain the required resource allocation under dataflow static scheduling list. This scheme allows one to make conclusion in Petri net domain to be applied in dataflow models to foresee the necessary amount of storage resources for each arc. An application example is used to illustrate the concept and effectiveness of the outlined approach.**

I. Motivation and Introduction

Since data flows arise along with synchronous dataflow [1], many extensions emerged to model digital signal processing applications, including cyclo-static dataflow [2], scalable synchronous dataflow [3] and boolean dataflow [4] to increase its power expressivity, and more recently meta-modeling techniques, such as Parameterized Synchronous Data flows (PSDF) and Homogeneous Parameterized Dataflow (HPDF). They allow the specification of a system, serve for simulation and may generate automatic code for real time synthesis.

Data flows graphs [5], [6] and meta-modeling techniques [7], [8], [9] that can be applied to different dataflow base models, as well as Petri Nets [10], [11] have demonstrate their adequacy in modeling systems, namely data-dominated systems as in digital signal processing intensive applications. Data flows can be viewed as a natural representation of target digital signal processing (DSP) because their representation exposes the parallel and concurrent processing for DSP systems and the constraints involved in their order evaluation are minimal, devised by a scheduler.

In distributed processing system, resources and information are shared among several processors. This sharing must be controlled or synchronized to insure the correct operation of the overall system. Petri net have been used to model a variety of synchronization mechanisms, including the mutual exclusion, readers-writers and producers-consumers problems.

Due to the lack of or reduced verification mechanisms in dataflow graphs, a major issue of our research is to devise techniques to increase the knowledge about intrinsic features of these dataflow graphs. We propose to translate dataflows into a different domain, where it is possible to reasoning about specific properties of the model, which will be mapped back into dataflow properties. For that we proposed to use Petri nets, where a set of verification techniques are available namely the invariant analysis.

With a similar goal in mind, the modeling of biological systems using Petri nets was introduced by Reddy et al. [12], and the research is nowadays govern into the domain of understanding the process in a living cell [13]. Petri nets were introduced as a modeling mechanism due to their similarity with biological systems, since the systems are bipartite (species/substrates and interactions), concurrent, stochastic and non-deterministic. Petri nets provide a mathematically representation of the inherently complex structures of biochemical pathways. Besides, with Petri net analysis techniques, biological system models may be validated qualitatively. In [13] behavior properties of the Petri nets are employed to check the model consistency and correctness using T-invariants. T-invariants are a set of events returning to a given state, and any system behavior may be decomposed into a linear combination of these basic behaviors reached by T-invariant analysis. To explore the validation of Petri net apoptosis [14] models Ian Wee Jin Low et al. [15] introduce P-invariant analysis. With this approach they validate and increase the confidence on transduction pathways [16] models' level, as well as based on definition of P-invariant, guarantees that certain parts of the net always maintain a fixed amount of tokens which can be understood as a conservation of the biological signal inside the modeled system.

The paper is organized as follows: In section II a summarized introduction of Petri nets is given. Section III presents a

978-1-4577-0434-5/11/$26.00 ©2011 IEEE

brief overview of dataflows. The translation between dataflows and Petri nets is revealed and unmasked in section IV. A working example is presented in section V to exemplify the translation of a Synchronous Dataflows into a Petri net graph. Results are discussed based on P-Invariants and T-Invariants analysis in section VI. The final section gives a summary of the results reached so far and addresses an outlook on future research directions concerning the duality of dataflows and Petri nets, namely the translation analysis of macro structures.

II. Introduction to Petri nets

In order to introduce Petri nets a summarized view is provided, for a more complete and formal description of Petri nets see for example [10], [17] or [18]. In this work we use Place-Transitions nets as our reference class of Petri nets.

Petri nets benefit from having a mathematical representation (which allows tool support and formal analysis) and graphical representation (which support easy production of documentation). Coming into the graphical representation, in Petri nets we can distinguish two basic elements, one that embodies the passive nature of the structure and the active part of the system to be modeled. Passive elements are represented by circles or ellipses which model normally conditions, states, resources or objects, while the active elements known as transitions, denoted by rectangles or boxes, models events, actions or activities that changes the values of the supposed conditions or objects. Petri nets also comprise tokens, which represent the value of a condition or object, drawn by black dots inside places. Arcs are used to describe interactions between the active and passive elements. It is only possible to connect nodes (transition/place) of different type, the so called bipartite graphs. An arc may has attached a natural number written near the corresponding arc, the arc weight, which specifies that a transition is enabled only when the input places has at least as many tokens as given by the arc weight. When an enabled transition fires, an equal number of tokens given by the arc weight in the input place is destroyed. Firing a transition will also create an amount of tokens specified by the associated arc weight in the output places.

The token distribution, called Petri net marking changes with the firing of enabled transitions in a net. A firing sequence will determine a sequence of new marking distribution. If a Petri net has *m* places, their marking is represented by a (*m*x1) vector M, where each element correspond to the number of the tokens in each corresponding place.

To describe the initial state of a model, a set of tokens is associated to the places, known as initial marking M_0. A Petri net can be defined formally by a 5 tuple structure as PN=(P, T, I, O, M_0) [17], where:

- P=$\{p_1, p_2, ..., p_m\}$ is a finite set of places;
- T=$\{t_1, t_2, ..., t_n\}$ is a finite set of transitions, with $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$;
- $I : (P \times T) \longmapsto N$ is an input function that defines directed arcs from P to T, where N is a set of natural numbers;
- $O : (T \times P) \longmapsto N$ is an output function that defines directed arcs from T to P;
- $M_0 : P \longmapsto N_0$ is the initial marking.

In the structural analysis, our attention is focused on two properties:

- **Place Invariant**. The existence of a place invariant means that the weighted token sum on a group of places remains invariant, i.e., constant, for any new marking. This property is useful to detect possible regions where the number of tokens grow indefinitely, requiring the allocation of infinite resources. In engineering domain [19] a place invariant reflects a conservation property of the protocol;
- **Transition Invariant**. In this case, a group of transitions may fire that it does not affect the new marking on the Petri net.

A. P-Invariant Analysis

Every Petri net with *m* places and *n* transitions can be transformed into a matrix called the incidence matrix $C = [c_{ij}]$ with dimension $m \times n$. The entries c_{ij} of the incidence matrix are integers obtained as

$$c_{ij} = c_{ij}^{+} - c_{ij}^{-} \tag{1}$$

where c_{ij}^{+} defines the arc weight from transition *j* to place *i*, and c_{ij}^{-} the arc weight from input place *i* to transition *j*, therefore equation (1) represent the tokens balance associated to the firing of transition *j*. The entries of the incidence matrix are expected to be positive values if there is a token gain, negative if exists a loss and zero if there is no change concerning the transition *j* and place *i*.

In invariant analysis, our concern is to design systems without inconsistencies that cope with properties which remain constant during execution of the model.

Using these P-Invariants in fundamental equation $M = M_0 + C \cdot f$, with firing sequence *f*, where we multiply it by v^T,

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

If there exist a $v \in Z^n$ such that $v \neq 0$, then v is a P-Invariant if only if in the equation (2), the right most part $v^T \cdot C \cdot f$ is null, i.e.,

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

B. T-Invariant Analysis

T-invariant analysis is another important property related to the incidence matrix. The nonzero entries in the T-invariant represent the number of firings of the corresponding transitions that belongs to a firing sequence allowing the Petri net to start with a marking M_n and goes back to M_n.

The integer solution *w* of the following equation,

$$C \cdot w = 0 \tag{4}$$

is called a T-invariant. With the T-invariant it is only possible to know the number of times each transition fires till it ends the cycle. In this sequence of firings the order of firings is not predictable. However with the reachability tree one can find the exact firing sequence, if exists.

III. Introduction to Dataflows

Gilles Kahn introduced in 1974 the principles of a language for parallel programming. The paper [6] presented the semantics, syntax and the graphical representation of simple processes to explore the concepts of the proposed outlined formulation approach. The basic idea was to represent programs using nodes and arcs. Nodes are associated to processes and arcs connect nodes to represent the flow of information between processes. These arcs are first-in first-out (FIFO) communication queue channels, where read operations are blocking and write operations non-blocking. Kahn Process Networks (KPN) are monotonic and continuous. Monotonicity implies that if nodes process more input information then also generate more output stream information. This is an important property since it allow parallel computation. The continuity property establish that a increasing continuous chain of information is mapped (processed) into another increasing chain.

Synchronous Dataflow (SDF) is an extension of dataflow where the number of data samples (tokens) are known in advance at compile time. In [20], Lee and Messerchmitt, point out conditions for correctness of SDF for homogeneous parallel processors sharing memory. Under this paradigm, algorithms are described by nodes and arcs. Nodes represent computations and the arcs are associated to data paths. Nodes may have no incoming arcs. In this case, the program execution may start at any time. If a node has data available on its incoming or outgoing arcs, it may fire (tokens or data samples are consumed or produced) whenever desires. SDF dataflow uses two types of nodes: (1) Synchronous and (2) Asynchronous. Asynchronous nodes are useful to control conditional execution of a subgraph, in similar way to a structured construct programming statement if-then-else. SDF is suitable for systems where all sample rates are rational multiples of other sample rates. In synchronous dataflow the number of tokens produced and consumed are fixed, which allows to perform a static schedule.

Since SDFs are limited in their representation of large parts of a program, a more general model is required to represent data-dependent and conditional execution using the balance equations. The addition of two actors (switch and select) allowed conditional statements like if-then-else and do-while loops of token consumption and production, as introduced in [4]. A switch actor has two inputs (input and control) and two output (one for the true, T condition and another for false, F condition). The Boolean value of the control token determined which output (T or F) receives a copy of a token from data input. Selector actor has three inputs (T, F and control) and one output. When a control token is received a token copy of the appropriate input (T or F) is transferred to the output, based on the Boolean value of the control token.

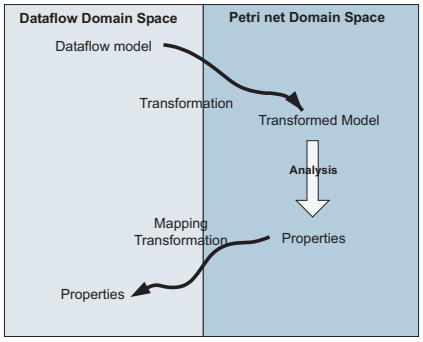


Fig. 1: Dataflow and Petri net domains

IV. Translating Synchronous Dataflow into Petri net

In signal processing domain, signals are commonly represented in the time domain, but most signal analysis techniques are done in the frequency domain. Signals can indeed be represented in both forms since they are equivalent ways of describing the same signal. Analyzing signals in time domain precludes and hide important signal features, so it is necessary to make a ”bridge” between the two domains (time and frequency). And sometimes problems are more easily solved in the time domain, while others are better settled in the frequency domain.

The time and frequency domains are linked by considering complex exponencial signals and impedances. Solving a differential equation relating the input and output of a circuit is easier if one provides the transformation to frequency domain. One method to switch from one domain to another is known as Fourier transform. With the Fourier transform (under certain signal conditions) its possible to make a bound to the frequency domain and achieve a continuous spectrum from a time signal, as well as it is also possible to perform the inverse operation.

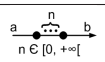
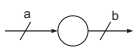
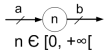
Likewise by analogy with the signal processing domain, our idea is to perform a switching method between two domains: (1) Dataflow (DF) domain space; (2) And Petri net (PN) domain space. Figure 1 depicts the whole picture of the proposed mapping approach. In dataflow domain space it is common to obtain a static scheduling using one balance equation per arc. And sometimes this type of analysis leads to a dead end since, some dataflow do not hold a static schedule. Due to this drawback and the lack of a formal analysis well established a mapping is outlined to take advantage of Petri nets well-known properties verification capabilities and at the same time enriching dataflow models concerning scheduler information and resource allocation performing an inverse transformation.

A. Translation rules

In table I one can find the proposed mapping in between dataflow domain space and Petri net domain space. A process

or function in dataflow domain is represented as a circle, whereas a resource, e.g. a First In First Out (FIFO) queue, is represented by an edge (arc) marked with an arrowhead to show the direction of the information flow. Arcs can also include weights to denote the amount of information that goes in and out of a resource. Taking into account the concepts of passivity and activity it is easy to make a connection between dataflow and Petri net components (nodes). Where one can find activity in dataflow domain (DD) the associated node in Petri net domain (PND) will be a transition, while if the component shows passivity in DD it will be mapped into a place in PND. Arc weights in DD will be mapped directly into arc weights in PND. As what concerns the initial marking there is a unequivocal direct mapping between dataflow domain and Petri net domain.

TABLE I: Translation Table from Synchronous Dataflows (DF) to Petri nets (PN)

	Proc./Function	Resource	Arc weight	Initial Marking
DF				
PN				

V. Working Example

Our analysis starts with a simple dataflow model depicted in figure 2. In signal processing systems it is important to know if there exists a static scheduling where a process can be repeated forever. This cycle (static scheduling list) can be determined establishing one balance equation per arc. This equation states that on every arc the amount of tokens produced by an origin node is equal to the amount of tokens consumed by a destination node. For the model considered later on and showed in figure 2, a set of balance equations (in our case five, one per arc) is obtained:

$$\begin{aligned} 2 \cdot a - 4 \cdot b &= 0 \\ b - 2 \cdot c &= 0 \\ 2 \cdot c - d &= 0 \\ 2 \cdot b - 2 \cdot d &= 0 \\ 2 \cdot d - a &= 0 \end{aligned}$$

Solving this trivial linear set of equation, the achieved solutions are:

$$a = 4,\ b = 2,\ c = 1,\ d = 2,$$

which states that in every cycle *node a* acts four times, *nodes b and d* act twice, and *node c* act just once, which correspondes to the static scheduling. Although it is important to note that there are situations where this set of equation does not have a root, which means that it is not possible to achieve a static schedule; This is not the case for our example. Applying the translation rules presented in table I we obtain Petri net model of figure 3 which allows us to gain a better insight of the functionality of the modeled system to be analyzed in the next section.

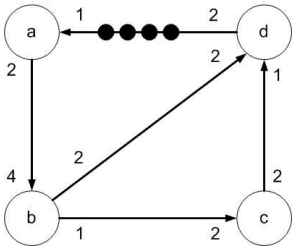


Fig. 2: Dataflow model. (From [21]).

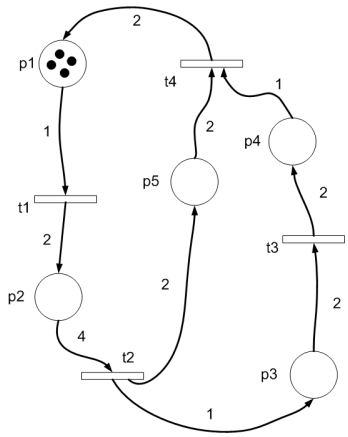


Fig. 3: Petri model obtained using the associated translation table I.

VI. Analysis

Two different software tools were used to produce analysis. INA (Integrated Net Analyzer) [22] is a software that allows modeling and analysis of Place/Transition nets and time/untimed nets using a text editor or the alphanumeric editor in a DOS window. With this alphanumeric editor the user can manage all functions necessary to enter, merge and edit nets. In fact the lack of a graphic user interface constrains the user to know the proposed semantic, to model nets. And TINA (TIme Petri Net Analyser) [23] is a software environment that cope with editing and analyzing of Petri nets and Time Petri nets.

Using INA to analyze the transformed dataflow into a Petri net in figure 3, two elementary properties are displayed, to mention:

- The net is structurally bounded, i.e., the net is bounded for every initial marking;
- The net is bounded, meaning that no place has never an unlimited number of tokens.

These two properties states that in the Petri net domain the former dataflow needs a limited amount of resources to perform their functionalities or activities. On other end the structurally bounded property reinforce that the behavior

of dataflow always require a limited amount of resources independently of the initial marking. This evaluation states that we must be aware of aspects related to problems with overflow since in communication processes places are associated to information storage areas, besides information must be stored without any kind of corruption or problem in the buffer areas.

In invariant analysis the incidence matrix (hereafter denoted by C) plays a center role. The components of each invariant vector denotes weights. For P-invariant, the associated vector reflects a constant weighted amount of tokens in a set of places independently of the firing sequence. Whereas for T-invariants, their vector components guarantees that after firing all transitions (as many times as indicated by each component value) the model reaches the same marking as before. The incidence matrix (where places are committed to rows and transition to columns) for this example is as follows:

$$C = \begin{pmatrix} -1 & 0 & 0 & 2 \\ 2 & -4 & 0 & 0 \\ 0 & 1 & -2 & 0 \\ 0 & 0 & 2 & -1 \\ 0 & 2 & 0 & -2 \end{pmatrix} \tag{5}$$

Performing a P-invariant analysis with INA/TINA, i.e, solving the system of linear equations $v^T \cdot C = 0$ yields the following P-Invariants vectors of the invariant basis space:

$$v = \{(2 \;\; 1 \;\; 4 \;\; 4 \;\; 0)^T, (2 \;\; 1 \;\; 0 \;\; 0 \;\; 2)^T\} \tag{6}$$

Therefore (2) reduces to $v^T \cdot M = v^T \cdot M_0$, and with the obtained P-Invariants we get,

$$\begin{aligned} 2 \cdot M(p_1) + M(p_2) + 4 \cdot M(p_3) + 4 \cdot M(p_4) = \\ 2 \cdot M_0(p_1) + M_0(p_2) + 4 \cdot M_0(p_3) \\ +4 \cdot M_0(p_4) \quad = \quad 8 \end{aligned} \tag{7}$$

$$\begin{aligned} 2 \cdot M(p_1) + M(p_2) + 2 \cdot M(p_5) = 2 \cdot M_0(p_1) \\ +M_0(p_2) + 2 \cdot M_0(p_5) \quad = \quad 8 \end{aligned} \tag{8}$$

where $M(p_i)$ denotes the marking in place p_i and $M_0(p_i)$ represents the initial marking in p_i.

To describe the P-invariants in (7) and (8) we can represent both expression as follow:

$$\sum_i k_{n_i} \cdot M(p_i) = K_n \tag{9}$$

where k_{n_i} represents the individual components of *n* P-invariant and K_n the result of the *n* components in $v^T \cdot M_0$. In this example both are $K_n = 8$.

It is important to note that based on equations (9) we can conclude about the maximum potencial number of tokens in each places belonging to P-invariant using,

$$M(p_i)_{max} = \frac{K_n}{k_{n_i}} \tag{10}$$

Analyzing the reachability tree we can see that the effective maximum number of tokens are exactly the same as the potencial maximum number of tokens for this situation. In general in every Petri net the maximum potencial number of tokens is less or equal then the maximum effective number of tokens.

On the other end with T-invariant analysis performed by INA/TINA, the Petri net shows just one vector in the basis space:

$$w = \{(4 \;\; 2 \;\; 1 \;\; 2)\} \tag{11}$$

As one can see in (11), this result was also achieved using a linear system formed by the balance equation of each node in dataflow domain. Furthermore this was expected in T-invariant analysis due to the performed translation between the two domains (dataflow and Petri net). On the other hand it is also important to outline that this T-invariant correspond to the static scheduling of the corresponding dataflow.

Another important issue in designing a signal processing system is whether a system exhibits a particular functional behavior, or the system requires a limited amount of resources. To find out if the system behaves in a certain specific way, one must search a sequence of firing transitions which enables the system to go along from a marking M_0 to M_n, where M_n is the specific state, and the sequence of firing in between defines the particular functional behavior.

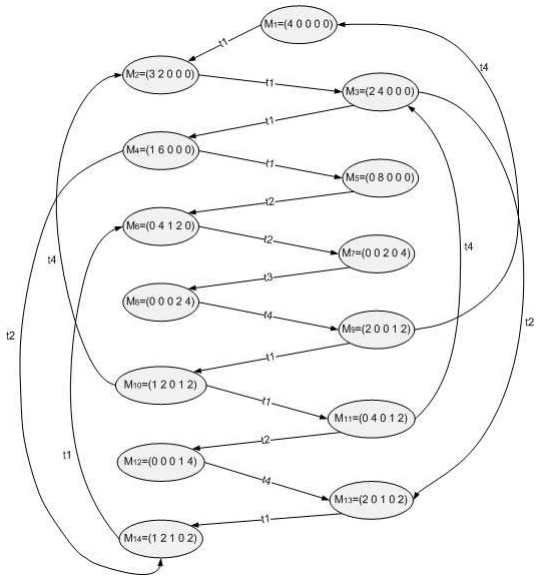


Fig. 4: Reachability tree of the Petri net shown in Fig. 3.

The coverability tree shown in figure 4 allows the study of these two former stated problems. Since resulting Petri net is bounded, the coverability tree may be called reachability tree. Observing figure 4 one finds out fourteen distinct states (marked in gray background elipses). Table II presents all reachable states obtained from the coverability tree.

Analyzing this table, without forgetting that each place represents a particular resource (for example a FIFO queue),

TABLE II: States of the Petri net shown in figure 3.

States	p_1	p_2	p_3	p_4	p_5
M_1	4	0	0	0	0
M_2	3	2	0	0	0
M_3	2	4	0	0	0
M_4	1	6	0	0	0
M_5	0	8	0	0	0
M_6	0	4	1	0	2
M_7	0	0	2	0	4
M_8	0	0	0	2	4
M_9	2	0	0	1	2
M_{10}	1	2	0	1	2
M_{11}	0	4	0	1	2
M_{12}	0	0	0	1	4
M_{13}	2	0	1	0	2
M_{14}	1	2	1	0	2

one can foresee in each queue the necessary amount of space to perform the static scheduling. The states in table II that comprise the static scheduling includes state M_0 to state M_9. Therefore for resource p_1 it is required a maximum buffer size of 4, for p_2 a maximum of 8 positions, for $\{p_3, p_4\}$ a maximum of 2 and finally for p_5 a maximum buffer size of 4.

VII. Conclusion and Results

Table III summarizes the relationship outlined and analyzed in section VI. To conlcude **k**-bounded places in Petri nets can be associated to buffers with length **k** in the dataflow domain. Regarding T-invariants there is a direct relation with the static scheduling of the dataflow, as focused in the former section. Places involved in P-invariants at Petri nets matches the potencial maximum buffer size at the dataflow models.

TABLE III: Properties mapping table

Petri net Properties	**Dataflow Properties**
k-bounded place	buffer with length **k**
T-invariant	static scheduling
place involved in P-invariant	Potencial maximum buffer size

To exploit and gain a better insight between dataflow domain and Petri net domain further refinement, improvement or extensions will be investigated later on.

Study on invariants focused on misbehaving dataflow, in which there is no static scheduling will be explored in future work to find out more hidden embedded features not normally addressed in dataflow analysis tools.

We intend to expand the analysis to complex and huge systems, meaning that a single node/place or arc/transition may represent a macro object; In this case we are saying that a single entity may contain inside a (sub)net or (sub)dataflow expanding the concept in order to introduce hierarchical structuring.

Also the integration of the proposed translation mechanisms within an embedded systems development framework based in Petri nets will be addressed in a near future.

References

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

[2] R. L. G.Bilsen, M. Engels and J. Peperstraete, "Cyclo-static dataflow," *IEEE Trans. Signal Processing*, vol. 44, pp. 397–408, February 1996.

[3] S. Ritz, M. Pankert, V. Zivojnovic, and H. Meyr, "Optimum Vectorisation of Scalable Synchronous Dataflow Graphs Scalable synchronous dataflow," *Methods*, pp. 285–296, 1993.

[4] J. Buck and E. Lee, "Scheduling Dynamic Dataflow Graphs with Bounded Memory using the Token Flow Model," in *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, vol. 1, Apr. 1993, pp. 429 –432 vol.1.

[5] E. Lee and T. Parks, "Dataflow process networks," *Proceedings of the IEEE*, vol. 83, no. 5, pp. 773–801, May 1995. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=381846

[6] G. Kahn, "The Semantic of a Simple Language for Parallel Programming," in *in Proc. of the IFIP Congress 74*. North-Holland Publishing Co, 1974.

[7] M. Sen, I. Corretjer, F. Haim, S. Saha, J. Schlessman, T. Lv, S. S. Bhattacharyya, and W. Wolf, "Dataflow-Based Mapping of Computer Vision Algorithms onto FPGAs," *EURASIP Journal on Embedded Systems*, vol. 2007, pp. 1–13, 2007. [Online]. Available: http://www.hindawi.com/journals/es/2007/049236.abs.html

[8] M. Sen, S. S. Bhattacharyya, T. Lv, and W. Wolf, "Modeling image processing systems with homogeneous parameterized dataflow graphs," *Computer*, no. March, pp. 133–136, 2005.

[9] B. Bhattacharya, S. S. Bhattacharyya, and S. Member, "Parameterized Dataflow Modeling for DSP Systems," *October*, vol. 49, no. 10, pp. 2408–2421, 2001.

[10] C.Giraud and R.Valk, *Petri Nets for Systems Engineering. A Guide to Modeling, Verification, and Applications*. Berlin - Heidelberg - New York: Springer, 2003.

[11] A. Yakovlev, L. Gomes, and L. Lavagno, Eds., *Hardware Design and Petri Nets*. Kluwer Academic Publishers, 2000, "ISBN 0-7923-7791-5, 331 pgs".

[12] V. N. Reddy, M. L. Mavrovouniotis, and M. N. Liebman, "Petri net representations in metabolic pathways," in *Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, 1993, pp. 328–336. [Online]. Available: http://portal.acm.org/citation.cfm?id=645630.662699

[13] M. Heiner, I. Koch, and J. Will, "Model validation of biological pathways using Petri nets demonstrated for apoptosis," *Biosystems*, vol. 75, pp. 15–28, 2004.

[14] M. Heiner and I. Koch, "Petri net based model validation in systems biology," in *Applications and Theory of Petri Nets 2004*, ser. Lecture Notes in Computer Science, J. Cortadella and W. Reisig, Eds. Springer Berlin / Heidelberg, 2004, vol. 3099, pp. 216–237. [Online]. Available: http://dx.doi.org/10.1007/

[15] I. Low, Y. Yang, and H. Lin, "Validation of Petri net apoptosis models using p-invariant analysis," in *Control and Automation, 2009. ICCA 2009. IEEE International Conference on*, 2009, pp. 416 –421.

[16] A. Sackmann, M. Heiner, and I. Koch, "Application of Petri net based analysis techniques to signal transduction pathways," *BMC Bioinformatics*, vol. 7, no. 1, p. 482, 2006. [Online]. Available: http://www.biomedcentral.com/1471-2105/7/482

[17] T. Murata, "Petri Nets : Properties , Analysis and Applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[18] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1981.

[19] To-Yat Cheung, "Petri Nets for Protocol Engineering," *Computer Communications*, vol. 19, no. 14, pp. 1250 – 1257, 1996, Protocol Engineering. [Online]. Available: http://www.sciencedirect.com/science/article/B6TYP-3XDS917-C/2/20ad02e2dad7df361fc5553cae15cb32

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

[21] L. Gomes and J. P. Barros, "Models of computation for embedded systems," in *The Industrial Information Technology Handbook, Section VI - Real Time and Embedded Systems*, R. Zurwaski, Ed. Boca Raton, FL: CRC Press, 2005.

[22] INA - Integrated Net Analyser, Humboldt - Universita zu Berlin. [Online]. Available: http://www2.informatik.hu-berlin.de/ starke/ina.html

[23] TINA-TIme Petri Net Analyzer, Laboratoire d'Analyse et d'Architecture des Systèmes. [Online]. Available: http://homepages.laas.fr/bernard/tina/