Employing a Timed Colored Petri Net to Accomplish an Accurate Model for Network-on-Chip Performance Evaluation

Jarbas Silveira[1], Paulo Cesar Cortez[1], Giovani Cordeiro Barroso[2], César Marcon[3]

[1]LESC-DETI / Federal University of Ceará - Fortaleza, Ceará, Brazil - 90455-970
[2]Department of Physics / Federal University of Ceará - Fortaleza, Ceará, Brazil - 90455-970
[3]PPGCC / PUCRS - Porto Alegre, Brazil – 90619-900
[1]E-mail: jarbas@lesc.ufc.br

Abstract

Network-on-Chip (NoC) is the most promising communication architecture for modern System-on-Chip (SoC). A system level analysis with a sound NoC model may provide an efficient NoC implementation. In this paper, we propose an accurate NoC model for performance evaluation based on Timed and Colored Petri Net (TCPN). The TCPN provides a detailed modeling of discrete event systems, enabling further evaluation of logical and temporal aspects with great precision. Experimental results with a 5×5 mesh NoC under synthetic and real traffic situations demonstrate the TCPN model efficiency in the latency predictability with low errors when compared with VHDL/SystemC simulation. Additionally, this work shows the ability of the model to allow fast building of different models and changes upon NoC architectural features such as routing algorithm and buffer length.

Keywords

Petri Net Modeling, NoC, latency evaluation

1. INTRODUCTION

Several designers propose Network-on-Chip (NoC) as a scalable solution for on-chip communication for modern System-on-Chip (SoC) [1]. Due to the tight time-to-market constraints, the success of a SoC design flow relies on its ability to perform fast validation and requirements' evaluation of the system. The NoC performance is highly dependent on the dynamics of the traffic patterns, packet injection rate, routing protocols and amount of buffering resources. Consequently, accurate NoC and traffic modeling is of crucial importance for performance analysis and platform optimization.

Among NoC models, we can highlight three ruling features: implementation effort, accuracy and simulation time. In addition, we can classify models for the performance analysis of NoCs in two groups: (i) the models that are clock-cycle accurate, normally implemented into event-oriented languages, such as VHDL, imply high accuracy, but great implementation effort and long simulation time; (ii) the analytic models, in general originated from system's functionality mapping or mathematical analysis. The rewards of using these models are mainly the low implementation effort and the low analysis time, since results are achieved only by the numerical resolution of equations.

A Timed and Colored Petri Net (TCPN) is an analytic formalism that enables powerful modeling of Discrete Event (DE) systems. TCPN have been used for systems modeling with low effort [2], since it uses the structure of Petri Nets associated to a high-level programming language. In addition, the possibility of adding time restrictions to the events allows the analysis of temporal aspects.

The paper is organized as follows. Section 2 provides a brief overview of TCPN modeling, and Section 3 details how TCPN models a NoC. Section 4 summarizes some related work. Section 5 explains how we generate and analyze data traffic. Section 6 describes the NoC model validation. Section 7 explores the proposed NoC model under synthetic traffic situations, whereas Section 8 explores a real traffic situation. Section 0 shows the influence of router service time in the packets latency, and finally Section 10 concludes this work.

2. TIMED AND COLORED PETRI NET MODELING

A Colored Petri Net (CPN) is a graphical language for model's construction and DE systems analysis, which mixes the capabilities of both Petri Net and a high-level programming language. CPNs are aimed at practical use, because they allow constructing compact and parametric models to describe complex systems at high-level of abstraction [3].

The concurrent systems operation depends crucially on the time taken by certain activities, and different design decisions may have a significant impact on the performance of a system. Timed CPN (TCPN) enables to capture the time taken by events enabling the designer to use TCPN to investigate metrics such as delays, throughput, and queue lengths, and for modeling and validation of real-time systems. Moreover, TCPN is suitable for NoC system modeling as it provides important features such as modularity, maintainability and expandability enabling to add new functions or new processes on the system. A hierarchical structuring mechanism underlies the TCPN modules concept, allowing a module to have sub-modules.

The structure of a TCPN is composed of two disjoint sets: *places* (represented by ellipses) and *transitions* (represented by rectangles), with direct arcs connecting places and transitions. Each place can be marked with one or more tokens, and each token has a data value (i.e. color). The number of tokens and their colors on the individual places represent the system state. A transition represents a DE, and its occurrence removes tokens from the places that have an arc leading to the transition and adds tokens to the places that have an arc coming from the transition. These tokens are determined by means of the arc expressions, which are the labels situated next to the individual arcs. A transition may have a *guard*, which is a Boolean expression that controls the transition binding.

The TCPN model encompasses a global clock representing time and a token can carry a timestamp together with its color, which specifies the time at which the token is ready to be used or removed by an occurring transition. Even in a hierarchical TCPN model, there is a single global clock.

3. TCPN Applied to NoC Modeling

We propose the TCPN applied to Network-on-Chip model (CPNoC), which is a NoC model based on *hierarchical TCPN*. The hierarchy on CPNoC is implemented with *substitution transition* that enables to connect levels of abstraction. CPNoC is flit accurate enabling to evaluate packet latency per flow and average packet latency.

The modelling presented can be used in a wide range of topologies. However, we modeled a mesh NoC with circular FIFO buffers at the input channels, decentralized arbitration with round-robin priority, XY routing algorithm and wormhole switching. Figure 1 shows an example of a 2×2 NoC modeled in CPNoC using CPN Tools [9], which enables to use graphical representation of CPN data types and complex data manipulation for state space and performance analyses.

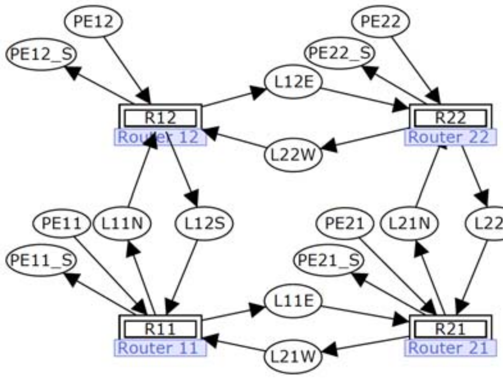

Figure 1. Top module of the CPNoC for a 2×2 mesh NoC.

Each substitution transition models a router associated to a subnet containing its internal model (e.g., the substitution transition R11 is associated with the subnet ROUTER 11). We modeled each router based on its physical structure, containing buffers, the Routing and Arbitration module (R&A) and the priority control.

The routers' interconnections are entirely modeled by input and output places of the substitution transitions (e.g., the place L11E represents the output east channel of the router R11 and the input west channel of router R21). Places model Processing Elements (PEs), e.g., the place PE11 models the source of flits of PE 11, and PE11_S represents the destination flits of PE 11.

TABLE I. Description of the n-tuple parameters.

Parameter	Description	Parameter	Description
nPac	Packet order	Or	Place origin of flit
nF	Flit order	tC	Creation time of the packet
t	Type of flit	tD	Arrival time of the packet
payload	Packet data	oL	Offered load of the packet
Rot	Routing parameter	ox, oy	X and Y origin coordinates
aux1, aux2	Auxiliary variables	dx, dy	X and Y target coordinates
		lx, ly	X and Y local coordinates

Tokens represent the flits. Several information, such as origin and destination coordinates constitute an n-tuple *Flit*. Table I summarizes the variables carried out by each flit.

Each router has an input buffer for each one of the following channels: Local, North, South, East and West. The flits in each buffer are carried out by a guard function associated with the input transition of the buffer's place. The Figure 2 models a simplified R&A module containing two input channels (i.e. South and North) and two output channels (i.e. West and East).

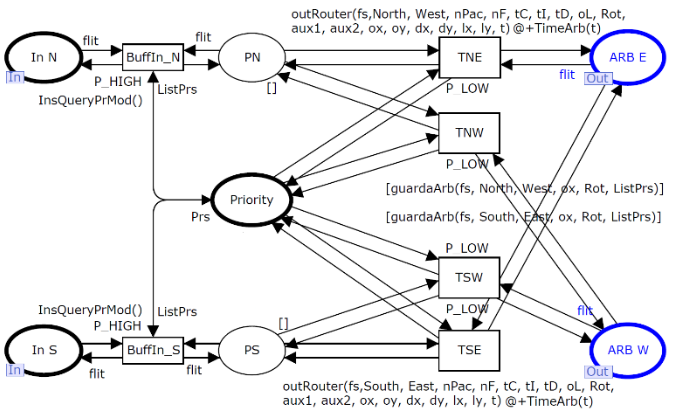
outRouter(fs,North, West, nPac, nF, tC, tI, tD, oL, Rot, aux1, aux2, ox, oy, dx, dy, lx, ly, t) @+TimeArb(t)
In N
In
flit
BuffIn_N
flit
PN
TNE
ARB E
Out
flit
P_HIGH
ListPrs
[]
P_LOW
InsQueryPrMod()
TNW
P_LOW
Priority
Prs
[guardaArb(fs, North, West, ox, Rot, ListPrs)]
[guardaArb(fs, South, East, ox, Rot, ListPrs)]
P_LOW
TSW
InsQueryPrMod()
ListPrs
P_LOW
flit
P_HIGH
[]
In S
BuffIn_S
PS
TSE
ARB W
In
flit
flit
Out
outRouter(fs,South, East, nPac, nF, tC, tI, tD, oL, Rot, aux1, aux2, ox, oy, dx, dy, lx, ly, t) @+TimeArb(t)

Figure 2. Simplified CPNoC of the routing and arbitration (R&A) of the flits (input channels North and South and output channels East and West).

In Figure 2, the arc that connects the *TNW* (i.e. Transmission from North to West) to the place *ARB W* (i.e. ARBitration on West) is associated with the function *outRouter*(...) *@+TimeArb*(*t*) that performs the routing algorithm. This function defines where each flit will be directed, obeying the route defined by the routing algorithm. In our model, we implement the XY routing, but we can explore several other algorithm by rewriting the *outRouter* function. The function *TimeArb*(*t*) temporizes the flit according to the type of flit (i.e. a header or payload). Additionally, the guard function *guardaArb*(...) verifies if the target channel is free.

The tokens in the place *Priority* implements an arbiter with round-robin priority. The flit is put in the place *ARB E* or *ARB W*, which represent the input buffer of the adjacent router. The subnets of adjacent routers use the hierarchical resources of the CPN Tools to shares places by the *Port Type* function.

In order to guarantee the temporal characteristics of the NoC modeling, the time of events associated to arcs functions and tokens were adjusted in accordance with VHDL simulation. Each flit takes 1 clock cycle to enter into the buffer. The header flit takes 4 clock cycles for routing and arbitration, while the remaining flits of the packet spends 1 clock cycle. These values are defined at the beginning of the simulation, and can be easily changed, enabling to model routers with varied service times.

We use the monitoring resource of CPN Tools to obtain the end-to-end latency of the packet and other measurements of the model. A monitor checks the time when the terminator flit arrives at its destination. Each flit terminator has its stored creation time inserted in the n-tuple variable *tC* (see Table I). The latency of the packet is calculated by the subtraction of the destination time from the creation time of the packet, thus obtaining the total end-to-end latency of the packet.

4. Related Work and Main Contributions

The analytical modeling theory underlies several works on NoC. The main objective of all these works is to achieve fast results with low precision loss if compared with clock-cycle accurate simulators. Table II summarizes the main features of some of these works relating them with our paper.

We proposed a NoC model to enable latency analysis with accuracy and low implementation effort. CPNoC allows fully visualizations of all states of the NoC and systematic execution of the model. Furthermore, the hierarchical model building enables better abstraction level, which allows easily exploring several variations of the model, like new routing algorithm, buffers depth and topologies. In addition, CPNoC accepts workload as trace files making the use of real traffic traces easy. To the best of our knowledge, CPNoC is the first model based on TCPN that presents analysis of real traffic and average packet latency on network and average packet latency per flow.

Table II. Related work summary.

Ref	Model	Analysis	error	Remarks
[2]	TCPN	latency	< 5%	Model is accurate only with less than 20% of traffic injection rate
[4]	Markov chain	latency	< 15%	Avoid state-space explosion using one Markov chain for each message flow. It is difficult to reuse the same model for other NoCs modeling
[5]	Markov chain	latency	< 5%	Avoid state-space explosion using one Markov chain for each PE of the message flow and recursively use the local mean latencies to obtain the mean latency of the complete path
[6]	QN	throughput and latency	VR	Model is only sound for low data traffic evaluation (i.e. less than 0.2 packets/cycle)
[7]	QN	latency and occupancy	< 3%	Performance analysis based on the channel service time and the channel waiting time, as well as channel occupation and contention
Our	CPNoC	latency	< 5%	Avoid state-space explosion using hierarchy

Legend:

QN - specific analytical models based on the M/G/1 queuing model [8].

VR - < 5% for low injection rate (i.e. < 0.2 packets/cycle) else high errors

5. Data Traffic Generation and Analysis

In our model, the place representing a PE receives tokens through an input text file. These tokens are grouped into packets that can contain different sizes and destinations. A special module, connected to the top hierarchy throughout a fusion place, controls the packet injection rate, which varies from 10% to 100%, in incremental steps of 10%.

The traffic analysis is performed through measurements and statistics of the packets delay. The CPNoC carries out this analysis using monitor resources, which is an inherent tool of CPN Tools. This analysis enables to identify the origin of each packet that arrives at a given target PE, allowing the analysis of individual packet flows. In addition, the CPNoC enables the use of monitors in any place or transitions, allowing internal evaluations of several NoC parameters to collect data statistics such as confidence intervals, number of observations, standard deviation, average, minimum and maximum values, and variance.

6. Validation/Calibration of CPNoC

We calibrated and validated CPNoC comparing with a VHDL NoC implementation, whose traffic generation and analysis was performed with SystemC simulation. The first set of experiments explores non-concurrent communications performed with non-blocking flows, crossing the NoC between pairs of PEs connected through varied quantity of hops. The second set of experiments, exploring concurrent flows, adopted three approaches: external concurrence, internal concurrence and simultaneous internal and external concurrence. All of these experiments contain 100 packets of 30 flits with load injection rate varying from 10% to 100%. The latencies were calculated as the average time from the packet creation until the last flit reaches the target PE.

Figure 3 illustrates the average latencies, where the CPNoC presents a negligible error (i.e. less than 1%) when compared with the VHDL/SystemC simulation.

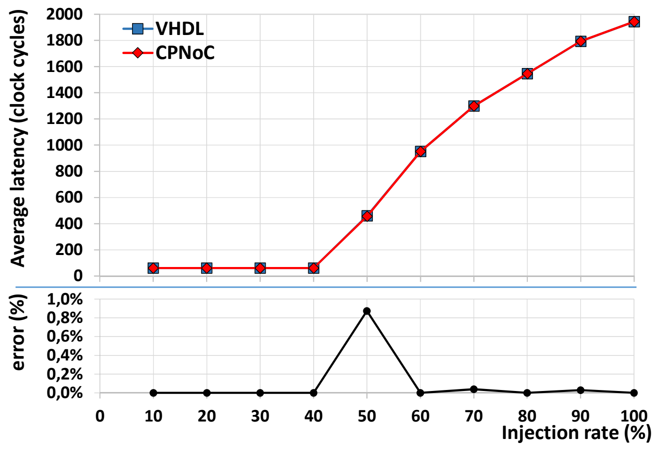

Figure 3. Average latencies and error with CPNoC and VHDL simulations.

7. Model Analysis under Synthetic Traffic

We applied uniform and hotspot traffics to analyze the average latency of a 5×5 mesh NoC. In the hotspot workload, all the packets of all PEs target the central PE. PEs send 100 packets of 20 flits, resulting 48,000 flits.

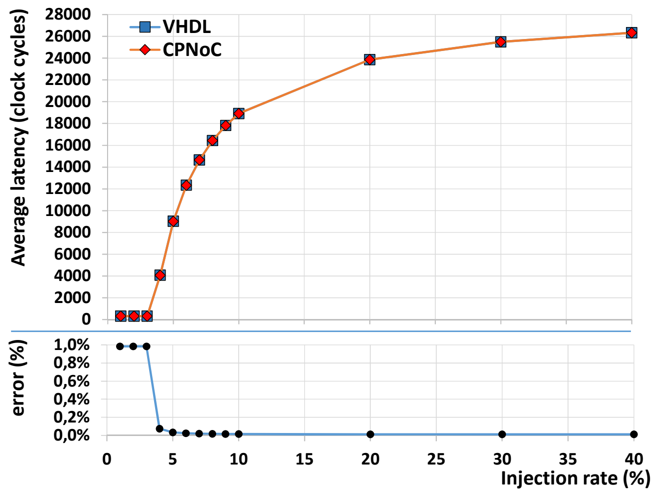

Figure 4. End-to-end average latency for Hotspot traffic.

Figure 4 illustrates the experimental results where the traffic saturation point occurred before 4% of injection rate, which is an expected behavior for hotspot traffic, because all communication channels converge to the central PE, consequently the NoC is rapidly congested.

For the uniform random traffic, the source PEs send 100 packets of 20 flits, resulting 50,000 flits. The destinations of the packets have equal probability of distribution between PEs, obeying the uniform random standard (all-to-all).

Figure 5 presents the end-to-end average latencies per packet, in which the traffic saturation point is between 24% and 26% of the traffic injection rate.

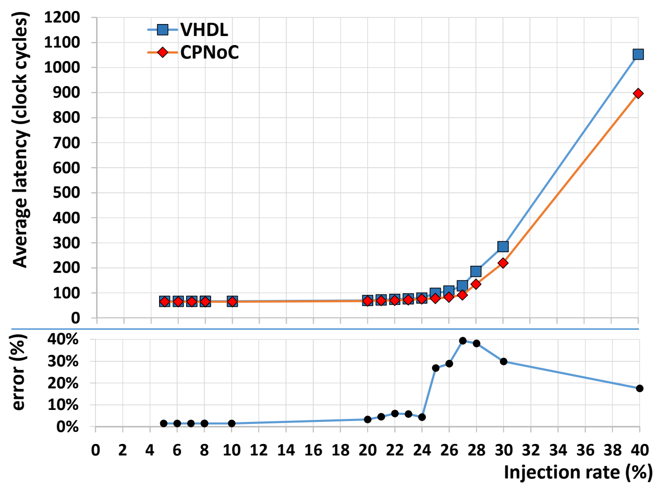

Figure 5. End-to-end average latency for all-to-all traffic distribution.

8. Model Analysis under Real Traffic Conditions

Figure 6 shows the multimedia application described in [10] mapped into a 4×4 NoC in order to evidence the ability of the model in capture the NoC behavior under real traffic conditions. The numbers in the arcs represent the volume of communication in multiples of 10 Kbits.

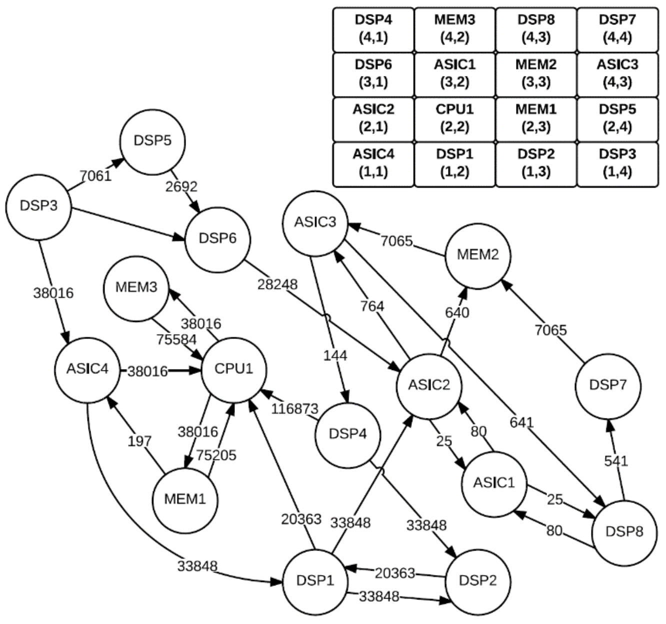

Figure 6. Multimedia application mapped in a 4×4 NoC.

Figure 7 presents the average latencies according to the traffic injection rates for CPNoC and VHDL/SystemC simulations.

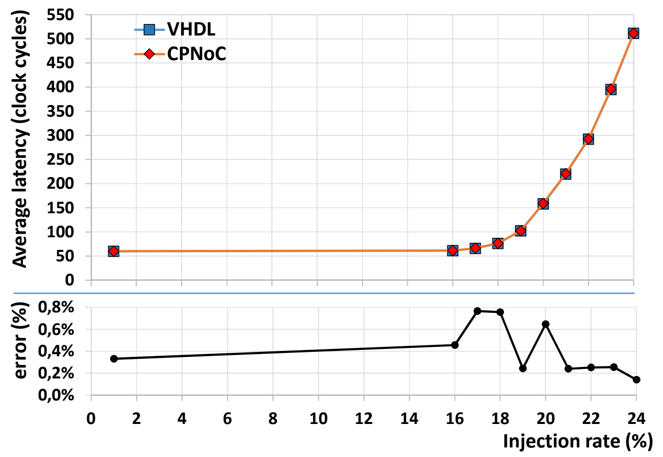

Figure 7. Average latency for a multimedia application mapped in a 4×4 NoC.

CPNoC produces estimation latencies of high precision in all simulated situations showing the model soundness, even in critical workload conditions, which is the case of the traffic saturation point. In [6], the authors analyze the same multimedia application through an analytical model, whose latency estimations follow the simulation results closely, but for packet injection rates above 20% (i.e. the traffic saturation point), their latency estimations produce significant errors, revealing that our model is much more appropriate to this kind of application's analysis under critical traffic load.

9. Analysis of Router Service Time

The service time of a router is the time that the router takes to arbitrate the header flit of a packet in the absence of congestion [6]. This time is frequently greater than the time of remaining packet flits, once the router is required to establish a route and to arbitrate the packet to the respective output channel. It is common for NoC designers to optimize the state machines of the router in order to minimize this service time, once this is able to improve the performance of the NoC [11]. In the application example, we explore 3, 5 and 7 clock cycles for the Router Service Time (RST).

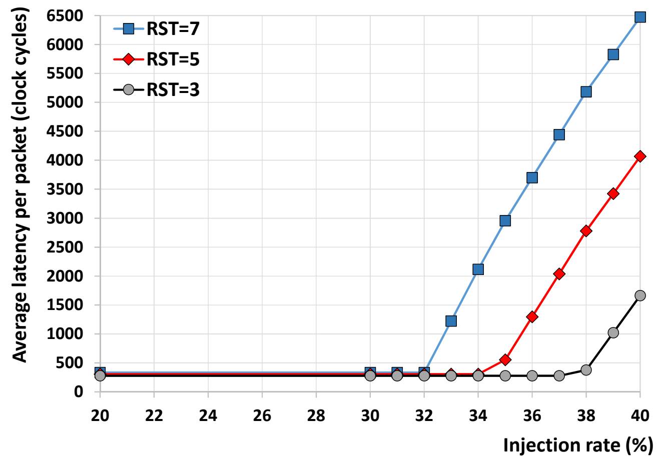

Figure 8. Latencies for 3, 5 and 7 clocks of Router Service Time (RST).

Figure 8 shows that less RST produces later saturation point, ensuring less average time of the packets. This can represent, for example, greater data flow on the NoC or an application with better performance. This information can help the designer to decide whether to utilize a router with less service time, once these routers commonly represent a greater area and consumption and great efforts implementation.

This application example shows the versatility of CPNoC, whereby a router with less service time would be of great effort in a VHDL/SystemC description. In this case, it would be necessary to design a state machine with minimum routing time and arbitration, which is an arduous task to be carried out with a hardware description language.

10. CONCLUSION

In this paper, we propose the CPNoC, which is a nearly clock-cycle accurate model of NoC based on Timed Colored Petri Net (TCPN). Experimental results of average packet latency for some traffic loads enabled to compare CPNoC with a VHDL/SystemC implementation. According to the results, CPNoC simulation presented excellent precision with deviations in order of no more than 1%. We provide performance evaluation for synthetic and real workloads. In the real traffic workload, the model presents the same precision before and after the saturation point of the network, revealing the power of model analysis for intense traffic situations.

In addition, CPNoC enables rapid performance analysis of a NoC with different router service times. In the VHDL/SystemC description, the same analysis would be of a great implementation effort. These results provide system level insights that can help designers to design NoCs efficiently.

11. REFERENCES

[1] R. Marculescu et al. “Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives,”, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 28, no. 1, pp. 3–21, Jan. 2009.

[2] H. Blume, T. von Sydow, J. Schleifer, and T. G. Noll, Petri Net Based Modelling of Communication in Systems on Chip, Petri Net, Theory and Applications, V. Kordic, Ed. InTech, 2008.

[3] K. Jensen and L. Kristensen, Coloured Petri Nets: Modelling and Validation of Concurrent Systems. Springer Berlin Heidelberg, 2009.

[4] E. Krimer, I. Keslassy, A. Kolodny, I. Walter, and M. Erez, “Static timing analysis for modeling QoS in networks-on-chip,” *J. Parallel Distrib. Comput.*, vol. 71, no. 5, pp. 687–699, May 2011.

[5] S. Foroutan et al. “A Markov Chain Based Method for NoC End-to-End Latency Evaluation,” *Symposium on* in *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pp. 1–8, Apr. 2010.

[6] U. Ogras, P. Bogdan, and R. Marculescu, “An Analytical Approach for Network-on-Chip Performance Analysis,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 12, pp. 2001–2013, Dec. 2010.

[7] S. Gajin, Z. Jovanovic. “An Accurate Performance Model for Network-on-Chip and Multicomputer Interconnection Networks”. *J. Parallel Distrib. Comput.*, vol. 72, no. 10, pp. 1280–1294, Oct. 2012.

[8] L. Kleinrock, “Queuing Systems: Theory”, Wiley, vol. 1, 1975.

[9] “CPN Tools” http://cpntools.org, 2013.

[10] J. Hu and R. Marculescu, “Energy- and performance-aware mapping for regular noc architectures,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 4, pp. 551–562, Apr. 2005.

[11] Y. Chen, Z. Lu, L. Xie, J. Li, and M. Zhang, “A single-cycle output buffered router with layered switching for networks-on-chips,” *Computers and Electrical Engineering*, 2012.