Accurate Model for Network-on-Chip Performance Evaluation Based on Timed Colored Petri Net

J. Silveira[1], A. Cadore[1], G. Barroso[2], C. Marcon[3], T. Webber[3], R. Czekster[4]

[1] Federal University of Ceará, LESC, DETI
[2] Federal University of Ceará, Department of Physics
[3] Pontifícia Universidade Católica do Rio Grande do Sul, PPGCC
[4] Universidade de Santa Cruz do Sul, PPGSPI

ABSTRACT

Network-on-Chip (NoC) is a power architecture that emerged to solve communication issues present in modern Systems-on-Chip (SoCs). NoC based architectures are very scalable and offer high levels of communication parallelism, among other features. Every efficient NoC implementation requires several design steps to accomplish indices of performance. Although there are many system level models, high-level models for NoC are representative in the context of design since they provide fast and accurate analysis, with low modeling effort, for further VHDL implementations. This work proposes a NoC model based on a Timed Colored Petri Net (TCPN) that computes performance indices seamlessly. Network latency and buffer occupation are of special interest in our approach as they represent the key indices when assessing NoC performance. As results, we have validated and refined the model of a 5×5 mesh NoC comparing its indices with equivalent VHDL RTL description under synthetic and real traffic situations. The proposed model is capable of analyzing the influence of the router service time on the average latency time, enabling internal NoC evaluation to optimize buffer length. Simulation results demonstrate the model suitability for latency evaluation with time estimation errors often below 1%. Furthermore, this paper discusses the effort required to extend the model with other NoC architectural features. We conclude that the use of a TCPN model of NoC generates accurate results providing as much detailed information as their equivalent experiments using VHDL description.

Index Terms: Network-on-Chip, Colored Petri Net, Performance Evaluation, System-on-Chip

I INTRODUCTION

Network-on-Chip (NoC) is a scalable architecture for on-chip communication that fulfills several requirements of modern Systems-on-Chip (SoCs), such as high communication parallelism and efficient energy consumption [1]. Under NoC communication paradigm, the routers are interconnected throughout links, and parallel messages concur for the same resources shifting the focus to the Quality of Service (QoS), whose primary metrics are packet delay and throughput [2]. Moreover, NoC performance is highly dependent on traffic patterns, packet injection rates, routing protocols and on the amount of buffering resources available on-chip.

Due to tight time-to-market constraints, the success of SoC designs relies on the ability to perform fast and rigorous evaluation and system validation on early stages of the design. Furthermore, accurate traffic modeling and simulation are crucial for adequate performance analysis and platform optimization. In this context, the application of meaningful simulation models plays a significant role in NoC design. The choice of a well-suited formalism for system representation considers three important ruling features: implementation effort, accuracy, and simulation time [3]. Usually, models with a high-level abstraction of a system description, i.e. simplified models, reduce the evaluation time but penalize accuracy. In contrast, models that are more complex, i.e. more detailed models, increase the accuracy when assessing results at the cost of spending much more time for evaluation and validation.

There are several NoC models for performance analysis, varying from analytical models (e.g., originated from system functionality mapping through mathematic or logic analysis) to cycle and bit accurate models, commonly known as CABA (Cycle Accurate Bit Accurate) models [3]. The CABA models are implemented using event-oriented languages, such as VHDL and System-C, requiring considerable implementation effort and usually long simulation time; however, the technique provides more accurate results.

Petri nets formalism [4] provide graphical modeling primitives such as places, transitions, and tokens, mainly for representing and analyzing systems with concurrent behavior. Colored Petri Net (CPN)

combines the major features of Petri nets for describing systems with the capabilities of a high-level programming language named CPN ML [4], and allows constructing compact and parametric models.

Formally, the structure of a CPN is composed of two disjoint sets: places and transitions. Directed arcs connect places to transitions and transitions to places. Circles or ellipses graphically represent places; i.e., the state of the modeled system. Bars or rectangles represent transitions; i.e., the events that can occur in the system. Each place can be marked with one or more tokens containing a data value, which is called the token color. The number of tokens and the token's colors on individual places represent the state of the system. The tokens on a specific place constitute the marking of the CPN model. The set of markings defines the possible system states that the model allows.

The occurrence of a transition consumes tokens from input places (i.e., places with arcs leading to transitions) and adds tokens to output places (i.e., places with incoming arcs from transitions). The occurrence of a transition is determined by arc expressions, which are written in CPN ML and built using variables, constants, operators, and functions. When all variables in an expression are bounded to values of the same type, the expression can be evaluated. A transition may contain an additional Boolean expression to describe guard conditions, which adds a constraint to enable it.

Timing aspects play a significant role in a broad range of concurrent systems modeling. The correct functioning of some of these systems depends crucially on time spent on certain activities. CPN includes time concepts that allow the description of timed events of the system. In a Timed CPN (TCPN) model [5], the tokens can carry a timestamp besides the color. Thus, the marking of a place where the tokens carry timestamps is a timed marking. The timestamp specifies the time at which the token is ready to be used or removed by an occurring transition. The TCPN can model performance measures such as maximum queue length and mean waiting times, and even if the operation of real-time systems meets a required deadline.

Each transition allows the insertion of a *time delay* flag, whose occurrence adds the delay to all output tokens that carry a timestamp. If this flag is associated with an output arc of a transition, the delay is added only to the timed tokens that are placed in the output place associated with that arc.

Figure 1 illustrates the CPN modeling of a communication buffer connecting a processor element to an arbiter of the network architecture. The place PE1 represents the processor element. The tokens in this CPN model represent the communication flits. The place B1 models the buffer while the place Arb1 symbolizes the arbiter. The place Depth indicates the buffer size, which has four units in this model. Remark that place PE1 has 8 Flits as an initial marking, which are inserted as '1`"Flit 1" ++, 1`"Flit 2" ++… 1`"Flit 8"" following CPN's syntax. The box indicated by the letter (a), directly above place PE1, indicates the processor element waiting for buffer space are currently processing 'Flit 7' and 'Flit 8'. The buffer (place B1) is at this time full (i.e. with four flits in processing: 'Flit 3' to 'Flit 6', observing the box indicated by the letter (b), directly above place B1). Besides, the arbiter consumed two flits ('Flit 1' and 'Flit 2') according to the box indicated by the letter (c) above place Arb1. The fire (occurrence) of transition T1 symbolizes the flit entry in the buffer B1, while the fire of transition T2 represents the output of a flit. Likewise, an arc can contain complex functions operating over tokens. The symbol '@N+' above a transition indicate that the event may spend some time related to this transition. The expression related to this time may contain a given constant value or a function to evaluate restrictions. Transition T2 exemplifies a restriction using the function '@+(if Flit= "Flit1" then 5 else 1)' indicating that if 'Flit 1' is being processed the time spent will be 5, else it will be 1.

Much of the NoC modeling work is based on the Markov chains and Queueing theory. To represent NoCs as Markov chains, the internal state of each router, as well as the characteristics of all flows, need to be expressed as states in the chain. Commonly, this approach results in an enormous and thus an intractable

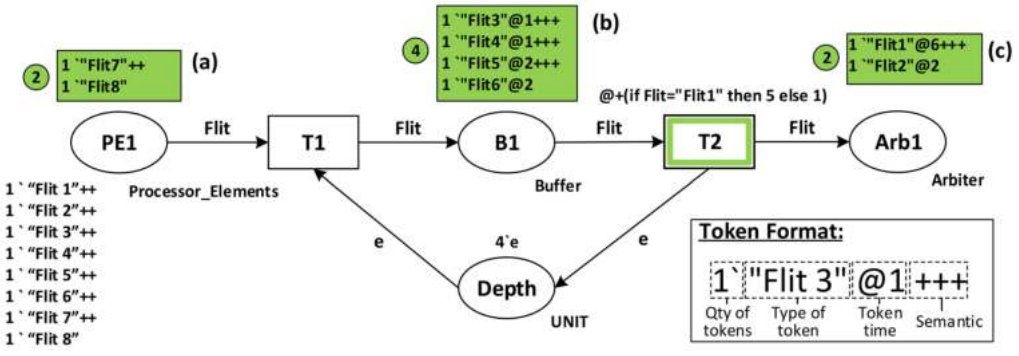

Figure 1. Example of CPN modeling of a communication buffer using CPN Tools [6].

number of states. Krimer et al. [7] generate a separate model for each isolated flow to avoid state-space explosion reducing the Markov chain to a manageable size. The model exploits three types of flow and calculates other flows from these three. However, it is not easy to apply the model for other NoC topologies, once it is necessary to find and generate the newly isolated flows. In [8], another model based on Markov chains has been developed to achieve latency time between two NoC processing elements. The authors modeled a reduced Markov chain to cope with state space explosion. Although the model presents high precision for less than 5% of error, each network flow needs to be described in details, making difficult the analysis of other flows.

One advantage of analytic models is their use for optimization purposes once the metrics are provided in a closed mathematical formula. Ogras et al. [9] proposed an analytic model to analyze performance within NoCs. In their approach, a router model based on a set of FIFO (First-in-First-Out) type buffers is connected through a switch. Their proposal provides important metrics that are typical in NoC performance analysis, such as the average packet latency per flow and maximum network flow. According to the results, the model shows good precision with an error around 5%, before the saturation point of the network. However, the model is not suitable for intense traffic; i.e., traffic generated with high injection rates (above 30%, in average) leading to NoC congestion.

The network traffic is an important aspect in NoC modeling. Most queueing approaches consider incoming and outgoing traffic as probability distributions (e.g., Poisson traffic) and allow designers to evaluate the network using statistical analysis to calculate metrics, such as average buffer occupancy and average buffer delay in an equilibrium state [10]. However, NoC applications usually possess traffic patterns that are not directly mapped to Poisson distributions [11]. More precisely, Poisson model fails to capture important network characteristics such as self-similarity or long-range dependence [12]. Some approaches, like [13] circumvent these limitations using a traffic approach based on a non-stationary and multifractal analysis. Nevertheless, one attractive feature of a model is to accept traces of real traffic, where it is possible to inject packets with different lengths, variable injection rate, and various destinations.

Blume et al. [14] modeled a 5×5 mesh NoC with an XY routing algorithm and FCFS (First Come First Served) arbitration using a Colored and Timed Petri Net (CTPN) [5]. In their approach, only the header and terminator flits were used to estimate the packet latency on the network. This approach limits the internal network analysis such as evaluation of internal channels and buffers. Moreover, the authors show only the emulation comparison for 20% of offered load since it is not possible to analyze the model performance under intense traffic workloads.

This paper proposes the CPNoC, which is a formalism for NoC modeling at high abstraction level using a hierarchical CTPN. The model is flit-accurate and presents typical results of NoC evaluation, such as packet latency per flow and average packet latency. The model includes the analysis and the injection of traffic, which simplifies the evaluation and parameters fitting. We used CPNoC to model a 5×5 mesh NoC with wormhole switching and XY routing. Furthermore, the model was applied to analyze the influence of the router service time on the average latency time of network packets. We validated the model throughout the comparison with CABA implementation in VHDL/SystemC using synthetic and real traffic situations.

Comparing against another model available in the literature, the main contribution of this paper, in contrast to related works, is a model that enables accurate performance analysis of a NoC with low implementation effort when compared to VHDL implementations. The CPNoC model also allows full visualization of NoC states and systematic execution of the model. Furthermore, the hierarchical model building enables better abstraction level, allowing easy modifications in the model such as changes in the routing algorithm, workload traffic, buffers depth and topology model. Besides, CPNoC accepts workload as trace files allowing the use of real traffic.

To the best of our knowledge, CPNoC is the first model based on CTPN that presents real traffic assessment, presenting standard metrics of NoC analysis, such as average packet latency and average packet latency per flow, providing an internal evaluation of NoCs, such as average buffers occupancy and packet latency per flow.

The remainder of this paper is structured as follows. Section 2 details a NoC model and a case study. Section 3 explains the model generation and analysis, as well as methodological aspects of the work. Section 4 exemplifies the proposed model application for synthetic traffic situations. Section 5 applies the model to a real traffic situation, whereas Section 6 shows the flexibility of the proposed model for exploring variations of the router service time and for evaluating buffers occupancy and internal flows. Finally, Section 7 presents our conclusions.

II NOC MODELING

Figure 2 shows a model with two hierarchy levels. The Level 1 (top level) represents the system more abstractly, containing a page for traffic generation (Page 1) and NoC analysis (Page 2). The Level 2 contains the instances of the subnet that represents the most refined model of the router.

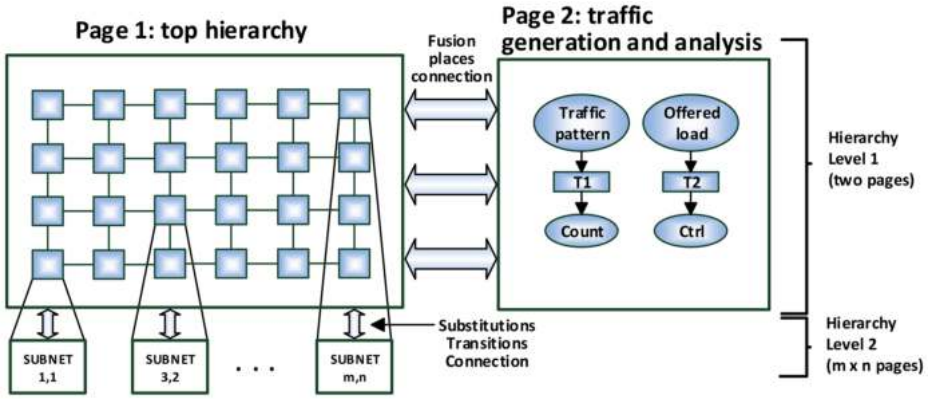

Figure 2. Hierarchy levels of the NoC model.

Figure 3 shows the top hierarchy view of the model that contains four substitution transitions modeling four NoC routers (hierarchy Level 1); e.g., the transition R11 is associated with a subnet by the name of ROUTER 11.

The interconnection between routers are entirely modeled by input and output places of the substitution transitions (e.g., the place L11E represents the east channel output of the router R11 and the west channel input of router R21). Places model the 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.

The router model uses an approach based on its physical structure encompassing the following parts: buffer, the Routing and Arbitration (R&A) module and the priority control circuit. Each router is represented by a subnet and can be instantiated throughout by a clone page within CPN Tools. We use the CPN Tools Variable Product that concatenates various types of variables performing an n-upla. The n-upla Flit, which is represented by a token, is constituted by information such as origin, actual and destination coordinates. The Table I summarizes the variables of the flit.

Each router can have up to five buffers, representing the input buffers of the following channels: local, north, south, east, and west. The size of the buffer is configurable, and the number of flits in each buffer is carried out by a guard function associated with the input transition of the place modeling the buffer.

Table I. N-upla parameters notation.

Parameter	Description
nPack	Packet order
nF	Flit order
t	Type of flit
payload	Packet data
Rout	Routing parameter
aux1, aux2	Auxiliary variables
Or	Place origin of flit
tC	Packet creation time
tD	Packet arrival time
oL	Packet offered load
ox, oy	X and Y origin coordinates
dx, dy	X and Y target coordinates
lx, ly	X and Y local coordinates

Figure 4 shows a simplified model of R&A module containing two input and two output channels.

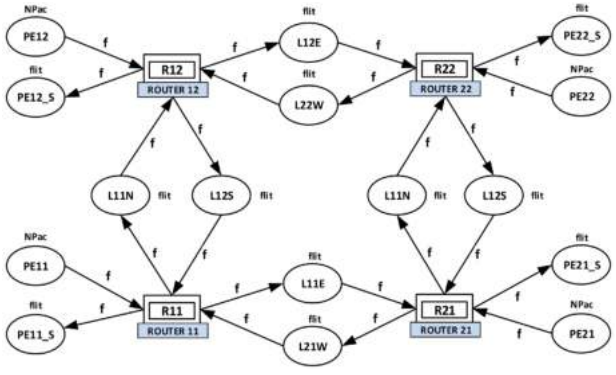

Figure 3. Top page of the model for a 2×2 mesh NoC.

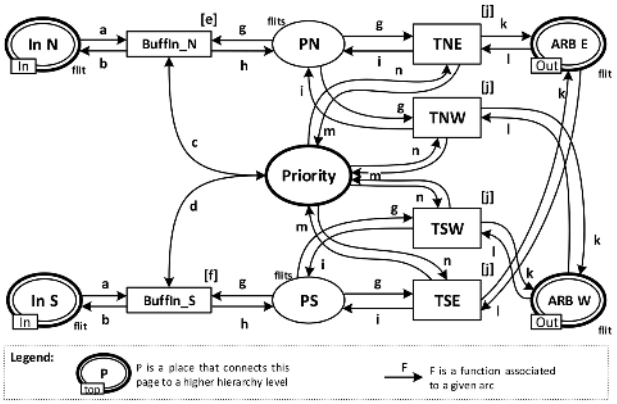

Figure 4. Simplified CPNoC model of the routing and arbitration module containing only the input channels North and South, and the output channels East and West. Table II describes the symbols associated with each transition of the model.

Table II. Symbols of Figure 4 with associated function and description.

Symbol	Function	Description
a	(nPac, nF, tC, tI, tD, oL, Rot, aux1, aux2, ox, oy, dx, dy, lx, ly, t, dado)	Gets the token representing a flit
b	1`(0,0,0,0,0,0,Rot,0,0,0,0,0,0,0,0,0,"null")	Returns a null token representing a channel in use
c, d	InsQueryPrMod(ListPrs, Local, ListPrs, retornaFlit(nf))	Control the round-robin arbiter's priority
E, F	Ox ≠ 0 and length fs < Buffer_Length	Guard functions that control the buffer size
g	fs	Returns the list of flits (tokens symbolizing a packet)
h	ins fs (nPac, nF, tC, tI, tD, oL, 0, aux1, aux2, ox, oy, dx, dy, lx, ly, t, dado)	Inserts the flit into the buffer queue
i	inRouter(fs, ox, Rot, inPort, outPort)	Returns a packet with non-forwarded flits
J	[guardArb(fs, inPort, outPort, ox, Rot, ListPrs)]	Guard function for the control of the packets flow between inPort and outPort buffers
k	outRouter(fs, inPort, outPort, nPac, nF, tC, tI, tD, oL, Rot, aux1, aux2, ox, oy, dx, dy, lx, ly, t) @+TimeArb(t)	Routing a packet from inPort to outPort
l	(nPac, nF, tC, tI, tD, oL, Rot, aux1, aux2, ox, oy, dx, dy, lx, ly, t, dado)	Returns the token which represents the flit
m	ListPrs	Gets the list of flits in buffers
n	QueryPrMod(ListPrs,inPort,outPort, ListPrs, hd fs)	Gets the priority regarding inPort and outPort

Following a flit transmission from the North input buffer to the arbiter of the West port, the arc connecting the TNW transition to the arbitration place ARB W is associated with the function 'outRouter(fs, …, t) @+TimeArb(t)' (letter 'k' in Figure 4). The function TimeArb(t) temporizes the flit according to its type. The letters inside brackets indicate the guard functions related to transitions; i.e., a Boolean condition that enables a given transition to occur. Empty brackets indicate that there is no guard function attached. Places associated to the type "flit" (see the word near the places) represent one flit at a time, as well as, model buffers containing several flits.

The priority module controls the precedence of input buffers in a Round-Robin (RR) manner. For better visualization, Figure 4 shows a simplified router with only two input channels. The TNW transition verifies through the guard function guardArb(fs, North, West, ox, Rot, ListPrs) whether the channel destination is free. The tokens in the Priority place controls the precedence turn since the arbiter implements an RR priority. The flit is put in the arbitration place (ARB E or ARB W), which represents the input buffer of the adjacent router. The subnets of adjacent routers share this place through the Port Type function, which is a hierarchical resource of the CPN Tools.

In CPNoC model, the time of events assume discrete values and are associated with arc functions and tokens. The time of events was adjusted according to the registered times in the CABA simulation to guarantee that the model meets the NoC temporal characteristics. Each flit spends one clock cycle to enter into the buffer. The header flit spends four cycles for routing and arbitration while the other flits spend one clock cycle. These time values are defined as a constant value at the beginning of the simulation, and can be easily changed, enabling to simulate routers with other service time.

Figure 4 shows that the arc function outRouter (represented in the figure by the letter 'k') performs the routing algorithm. This function defines where each flit will be directed, following the path defined by the routing algorithm. In our model, we implement the XY routing that can be easily changed by rewriting the outRouter function.

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

The modeling methodology presented in this paper is all-purpose, and it can be used in a broad range of NoC topologies with minor modifications.

III TRAFFIC GENERATION AND TRAFFIC ANALYSIS

The traffic modeling defines how the PEs must transmit data to their respective destinations. This modeling can consider parameters such as packets spatial distribution, packet injection rate and packet size [15].

The size of the packets injected into a NoC can be fixed or variable. This work uses fixed size packets, although the model enables the use of packets of varied sizes. Therefore, the control of the packet injection rate into the network is carried out in the creation time interval between packets and follows the concept of percentage channel occupation. Equation 1 shows the waiting time computation between packets.

$$\text{Time}_{\text{idle}} = \left(\frac{1}{\text{Of}_{\text{load}}} - 1\right) \times \text{Pack}_{\text{size}} \times \text{NCyclesFlit} \qquad (1)$$

where: $\text{Time}_{\text{idle}}$ is the time interval between packet creations, Of_{load} is the percentage of the offered load, $\text{Pack}_{\text{size}}$ is the packet size and NCyclesFlit is the number of cycles that each flit spends entering in the buffer.

In CPNoC model, the place that represents the output of a PE receives the tokens from a text file. These tokens are grouped in packets that can contain several $\text{Pack}_{\text{size}}$ and destination nodes. Another page (Figure 2) of the model controls the injection rate. This page is connected to the top hierarchy throughout fusion places and it controls the traffic injection and the traffic analysis. The interval time between packets is calculated for each Of_{load} (Equation 1). For the model validation phase, Of_{load} varies from 10% to 100% in incremental steps of 10%.

The traffic analysis was performed through measurement and statistics of packets delay. The CPNoC model carries out this analysis using monitor resources, a built-in mechanism of CPN Tools. This analysis enables to identify the origin of each packet that arrives at a given destination, which allows identifying individual flows in the NoC. Besides, the CPNoC model permits the use of monitor in any place or transition, allowing internal analysis of the NoC parameters. It is possible to collect data statistics for a given condition of observation previously defined on a monitor, such as average, minimum and maximum values, confidence intervals, number of observations, standard deviation, and variance. These characteristics allow the traffic analysis and generation be performed based on the model, which presents implemented monitors provided by CPNTools.

A) NoC Model Implementation

The NoC model implemented for experimental results purposes presents the following characteristics: 5×5 mesh topology, circular FIFO buffers at the channels input, decentralized arbitration with round-robin priority, XY routing algorithm, and wormhole switching. The mesh topology is one of the most regular and exploited topologies in NoCs. Its regular structure facilitates scalability, whereby each router is connected to another router using adjacent links.

The router implements decentralized arbitration that allows the connection of all five channels in the best case, decreases average service times and allows to produce reduced latencies [16]. The router contains a buffer in each one of the five input channels. The buffer size is configurable within the CPNoC model and CABA implementation.

The wormhole switching is a technique that separates the packets of a message into flits, which are the smallest units of traffic on a network. This technique is widely used in cluster applications [17], SoCs and NoCs [18], decreasing the network latency and requiring small router buffer size. During a packet transmission, its flits are distributed along the destination path on the NoC. Our implementation uses two control flits. A header flit initiates the packet and contains all routing information to establish a network path. A termination flit finalizes the packet after the remainder of the packet is transmitted throughout the established path. The size of the packet varies according to the injected traffic in the network and is fixed for a network data structure. In the proposed model, the flits have a fixed width, while the packets can have variable sizes.

B) Traffic Detailing

We validate the CPNoC model through a NoC implemented in VHDL with concurrent and non-concurrent traffic. Both, VHDL and CPNoC implementations follow the same 5×5 NoC mesh specification detailed in Section 3.1. In the VHDL implementation, the communication channel has data band and control band. The data band composes the flit that has a configurable size. The control band has 2 pins that indicate the type of current flit ("01" for the header, "10" for the payload and "11" for the terminator). The header encloses the packet destination address that is used for the choice of path to be followed. The terminator flit contains the end of packet signaling. The traffic is injected and collected using SystemC modules and the simulation stops when all the packets are received.

Initially, the traffic simulation is carried out without concurrence to validate and refine the CPNoC model, with only one flow transiting the network. We use eight destinations, varying the number of hops between the nodes of the network. For instance, in Figure 5, the flow from node 35 to node 51 has six hops of distance while the flow from node 31 to node 12 has three hops of distance applying XY algorithm.

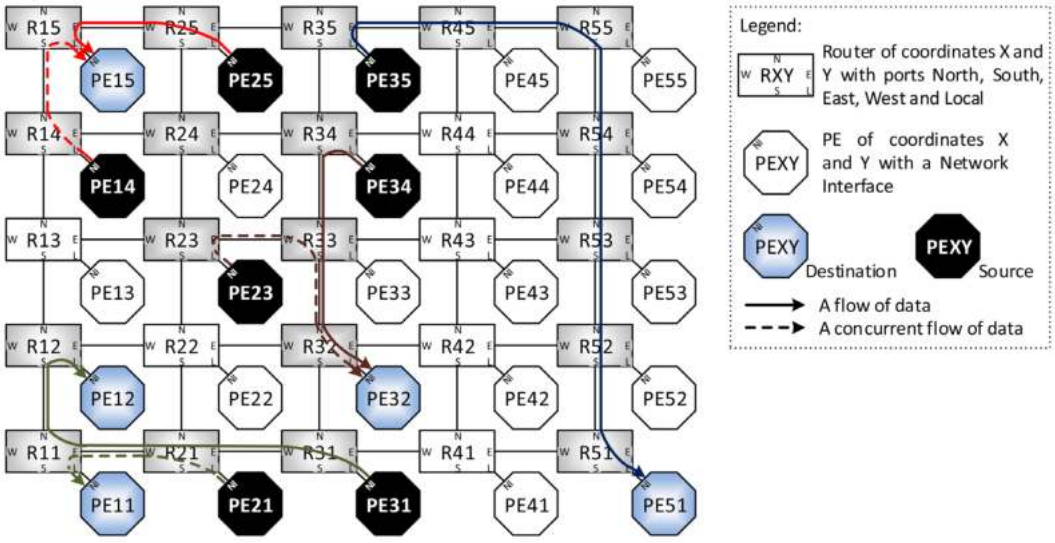

Figure 5. Examples of packet flows highlighting the XY routing and the number of hops.

We adopted three approaches for the concurrent flow tests: external concurrence, internal concurrence and simultaneous internal and external concurrence. In Figure 5, the flow from node 14 to node 15 is concurring with the flow from node 25 to 15 for the NoC output channel of router 15 (input channel of the PE), representing the competition for external resources of the NoC. The flow from node 23 to node 32 uses the South channel of node 33, concurring with the flow from node 34 to node 32. These two flows represent an internal competition for the link between nodes 33 and 32, and, in the same way, an external competition, because the local channel of node 32 is in dispute. Another situation, exemplified in Figure 5, shows that the flow from node 31 to node 11 concurs for the East channel of router 21 that transports the flow from node 21 to node 12, representing an internal competition for network resources.

In both simulations (CPNoC and VHDL models), we apply the following parameters: 100 packets of 20 flits with offered load varying from 10% to 100%. The delays applied to timed arcs of CPNoC model are the same in the VHDL; i.e., the router service time of the header flit delays five clock cycles and each one of the remaining flits delays one clock cycle. The end-to-end average latency metric is calculated as the average time since the packet creation until the last flit of this packet reaches the destination.

Figure 6 illustrates that the CPNoC model presents a minor error when compared to the CABA model. The results of the other concurrent and non-concurrent flows are not shown, but they are in the same order of precision.

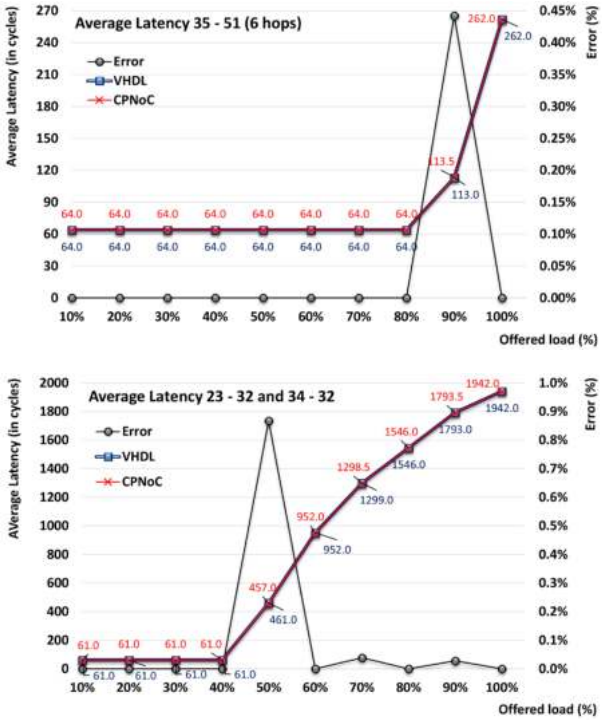

Figure 6. Adjustment of the average latency for end-to-end flows.

IV MODEL ANALYSIS UNDER SYNTHETIC TRAFFIC WORKLOADS

Frequently, designers of NoCs apply synthetic traffic workloads (e.g., uniform random, bit permutation and hotspot) to analyze performance [1]. This section uses the uniform random, and the hotspot standards to analyze the average latency time of a 5×5 mesh NoC. In the hotspot workload, all

PEs have a unique destination. The central PE of the network (i.e., node 33) was used as the destination of all the packets. The 24 remaining PEs send 100 packets of 20 flits, totaling 48,000 flits on the network for each offered load. Figure 7 shows that the saturation point of the network occurs before the 4% offered load, which is the expected behavior for hotspot traffic once the channels connect themselves to the central PE and the network congests rapidly.

In the uniform random traffic, all PEs send 100 packets of 20 flits to random destinations, totaling 50,000 flits on the NoC for each offered load. The packets destinations have an equal probability of distribution between the PEs, following the uniform random standard. Figure 8 shows the curve of the end-to-end average latency per packet, where the NoC saturation point is between 25% and 30% of the offered load.

For the traffic with uniform distribution, the model presents better behavior before and after the point of saturation, with errors of less than 5% but this error increases during saturation. The CABA model is synchronous; therefore, the flits always travel in sequence following a clock event. In the CPNoC model, one can have various transitions enabled, but it is not guaranteed that they will always be fired in the same sequence. To minimize this problem, the transitions that do not represent conflicts are modeled with less priority, ensuring that the tokens arrive at the same time in arbitration place at distinct routers. The model presents high precision with hotspot traffic, thereby producing errors of less than 2% during all the simulation. Due to the nature of the hotspot traffic, the conflicts are more concentrated, which minimizes the effect of lack of synchronization yielding estimations that are more accurate.

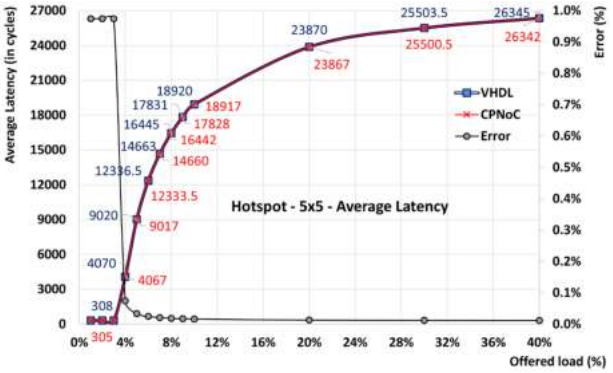

Figure 7. End-to-end average latency for hotspot traffic in a 5×5 NoC.

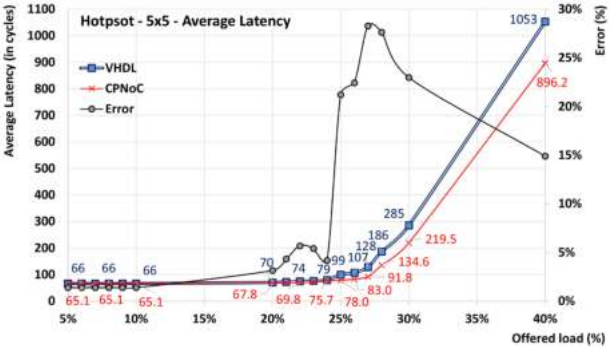

Figure 8. End-to-end average latency for traffic with random uniform distribution in a 5×5 NoC.

V ANALYSIS OF THE MODEL UNDER REAL TRAFFIC CONDITIONS

We employed the multimedia application described in [19] to highlight the model ability in the NoC analysis under real traffic conditions. Figure 9 displays a task graph that characterizes the application, which was mapped into a 4×4 NoC. The numbers in the arcs represent the volume of communication in multiples of 10 Kbits.

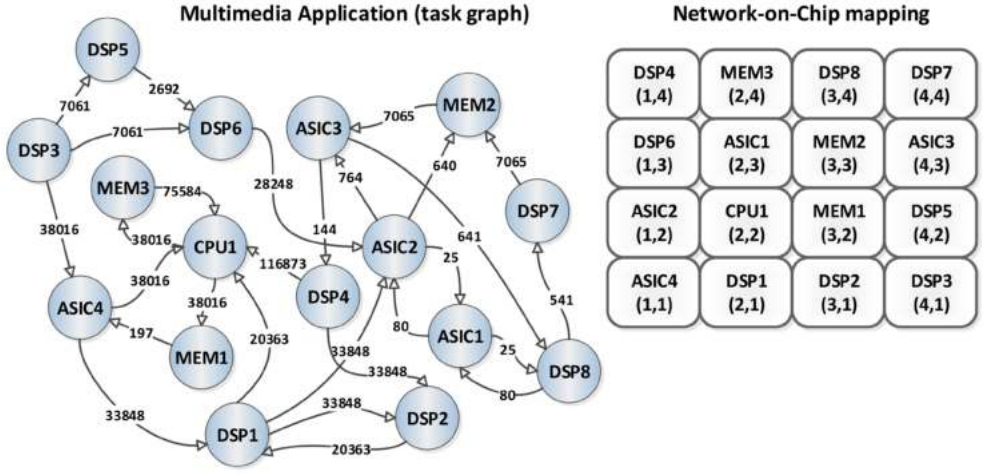

Figure 9. Multimedia application mapped into a 4×4 NoC.

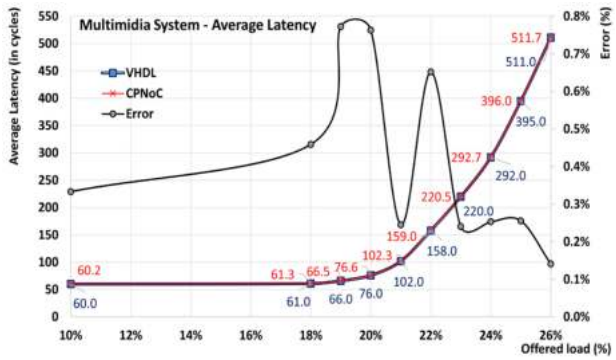

Figure 10. End to end average latency for a multimedia application mapped in a 4×4 NoC.

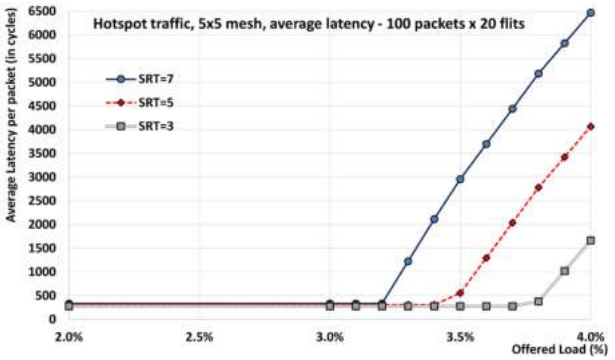

Figure 11. End-to-end average latency for hotspot traffic with the variation of router service time.

Figure 10 illustrates the values of average latena cy as a function of the offered loads for both CPNoC modeling and VHDL simulation. The CPNoC model accurately estimates the latency in all situations of offered load, even after the point of saturation, which demonstrates the model capacity in critical workload conditions.

In [9], the authors analyze the same multimedia application through an analytical model. We observe that the latency values estimated by the proposed approach follow the simulation results closely, but for packet injection rate above 20%, the latency values start increasing abruptly, since at this critical traffic load the network enters in congestion.

VI ANALYZING THE MODEL VERSATILITY

A) Exploration of the Router Service Time

Ogras et al. [9] define the router service time as the time that the router takes to arbitrate the packet header flit in the absence of congestion. This time is frequently greater than the time to arbitrate the remaining packet flits, once the router is required to establish a path and to arbitrate the packet to the respective output channel. It is common for NoC designers to optimize the state machine of the router to minimize this service time, and consequently, improve the performance of the NoC [20].

In the application example, we vary the router service time from 3 to 7 clock cycles with intervals of two cycles. Figure 11 shows that less service time implies later saturation point, enabling less average time of the packets on the network and representing, for example, a greater data flow on the NoC or an application with better performance. This information can help the designer to decide whether to use a router with less service time; once these routers commonly represent a larger area and consumption with increased implementation effort.

This application example shows the versatility of the CPNoC model, where a router implementation with less service time would be hard to achieve regarding effort in CABA models. In this case, it would be necessary to design a state machine with a minimum time of routing and arbitration, which is a hard task to be carried out in a hardware description language.

Evaluation of Average Buffer Occupancy and Isolated Flows

Internal performance evaluation of NoCs allows the designer to verify the performance at each router and each channel, buffers usage and isolated flows. This section shows how the CPNoC model can be used to measure the average usage of buffers at each router and how the length of buffers can be adjusted to optimize the packet latency and buffer usage. The application example applies the same workload shown in Figure 9.

Figure 12(a) illustrates the 4×4 NoC target architecture with the average value of buffers usage of each router. Here, one can notice that the higher average value of buffers usage is less than two. Additionally, Figure 12(b) depicts that the average usage of buffers is less than one. Consequently, the buffers of the NoC could be designed to use less depth while maintaining the same packet latency. Figure 12(c) and (d) highlight this affirmation showing the average latency for NoC and isolated flows considering buffers of depths 8 and 2, respectively. These two figures show that the average latency values are the same.

This example shows the applicability of the CPNoC model to internal NoC evaluation. Furthermore, it shows how the CPNoC model can be used to highlight valuable insights to NoC designers, demonstrating that consuming low area and power the NoC can provide the same average latency and consequently the same performance. The designers obtain these results with few implementation effort just including simple monitors in the CPNoC model. Our model is easily translated and mapped to other realities in NoC design, thus justifying the use of analytic modular approaches to complex systems.

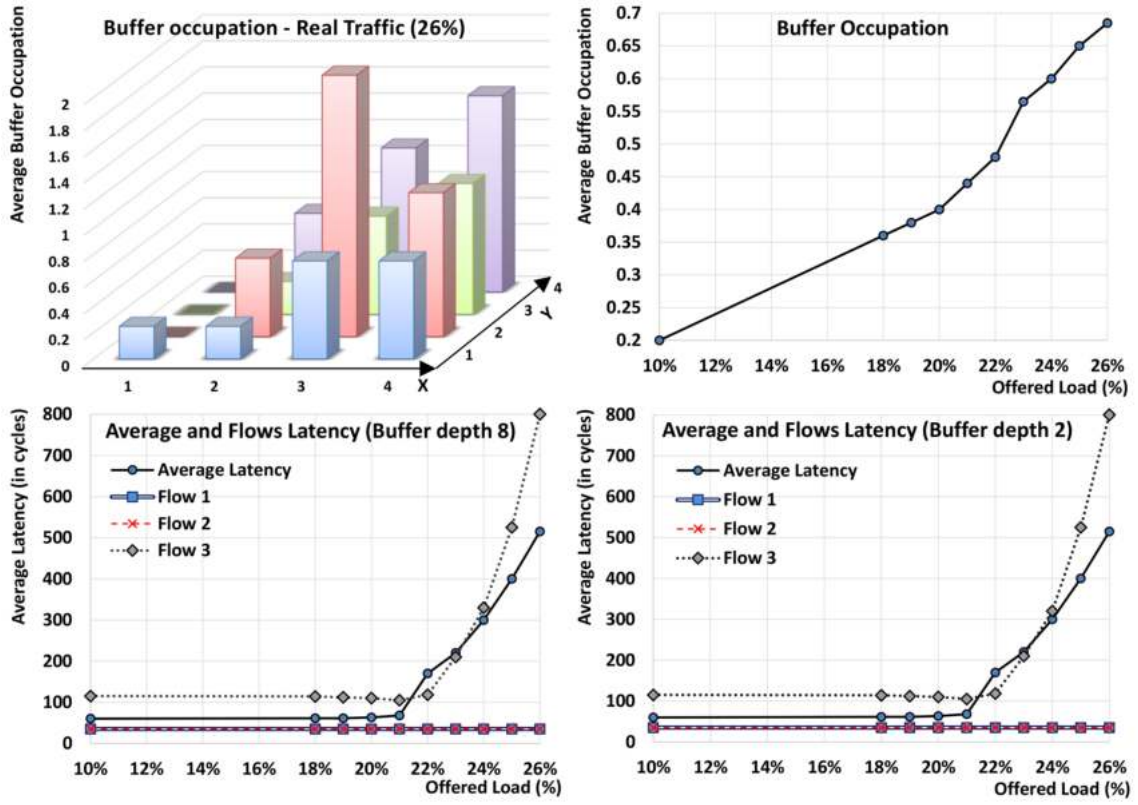

Figure 12. Internal NoC evaluation for a multimedia application mapped in a 4×4 NoC: (a) Average buffers occupation at 26% of offered load and 8-buffer depth. (b) NoC buffers occupation with 8-buffer depth, in average; (c) and (d) shows the average packet latency for NoC and the average packet latency for isolated flows for 8-buffer and 2-buffer length, respectively.

VII CONCLUSION

This paper proposes the CPNoC model, which is based on Colored and Timed Petri Nets. The model is accurate at flit transition level and presents typical NoC evaluation results, such as average packet latency for different traffic offered loads. All the analysis and offered loads are planned for the model, which aids the exploration and evaluation of NoC configurations.

We employed some experiments to compare the CPNoC model with a CABA precision model implemented in VHDL/SystemC, and according to the results, the model presents an excellent accuracy. Additionally, we provide performance evaluation for synthetic and real workloads. In the real traffic workload, the model presents the same precision before and after the network saturation point, revealing the power of model analysis for intense traffic situations.

The CPNoC model enables the performance analysis of a NoC with different router service times. These results provide system-level insights that can help engineers to design a NoC efficiently. In the CABA model, the same analysis would demand higher implementation effort. Furthermore, we provide an internal evaluation of the NoC, presenting buffer occupancy analysis. The results show the applicability of the CPNoC model to internal NoC evaluation, because the model enables to explore some network resources as buffer size and the service time of the router allowing reducing area consumption and power dissipation.

REFERENCES

[1] A. Touzene. **On All-to-All Broadcast in Dense Gaussian Network On-Chip**. *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, Issue 4, pp. 1085-1095, Apr. 2015.

[2] E. Bolotin, I. Cidon, R. Ginosar, A. Kolodny. **QNoC: QoS Architecture and Design Process for Network on Chip**. *Journal of Systems Architecture: the EUROMICRO Journal - Special issue: Networks on chip*. v. 50, Issue 2-3, pp. 105-128, Feb. 2004.

[3] S. Pasricha, N. Dutt. **On-Chip Communication Architectures, System on Chip Interconnect**. *Morgan Kaufmann*, 2008.

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

[5] J. Wang. **Timed Petri Nets Theory and Application**. *Kluwer Academic Publishers*, USA, 1998.

[6] AIS group, Eindhoven University of Technology. **CPN Simulation Tool, CPN tools 4.0.0**, *http://cpntools.org*. Last access: Dec. 05, 2016.

[7] E. Krimer, I. Keslassy, A. Kolodny, I. Walter, M. Erez, **Static Timing Analysis for Modeling QoS in Networks-on-Chip**. Journal of Parallel and Distributed Computing, vol. 71, Issue 5, pp. 687-699, May 2011.

[8] S. Foroutan, Y. Thonnart, R. Hersemeule, A. Jerraya. **A Markov Chain based Method for NoC End-to-End Latency Evaluation**. *IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pp. 1-8, 2010.

[9] U. Ogras, P. Bogdan, R. Marculescu. **An Analytical Approach for Network-on-Chip Performance Analysis**. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, Issue 12, pp. 2001-2013, Dec. 2010.

[10] M. Bakhouya, S. Suboh, J. Gaber, T. El-Ghazawi, S. Niar. **Performance Evaluation and Design Tradeoffs of On-Chip Interconnect Architectures**. *Simulation Modelling Practice and Theory*, vol. 19, Issue 6, pp. 1496-1505, Oct. 2011.

[11] R. Marculescu, J. Hu, U. Ogras. **Key Research Problems in NoC Design: a Holistic Perspective.** *IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pp. 69-74, 2005.

[12] C. Celik, C. Bazlamacci. **Effect of Application Mapping on Network-on-Chip Performance**. *Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pp. 465-472, 2012.

[13] P. Bogdan, R. Marculescu. **Workload Characterization and Its Impact on Multicore Platform Design**, *IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pp. 231-240, 2010.

[14] H. Blume, T. von Sydow, J. Schleifer, T. Noll. **Petri Net Based Modelling of Communication in Systems on Chip.** *Petri Net, Theory and Applications*, *InTech*, 2008.

[15] L. Tedesco, A. Mello, D. Garibotti, N. Calazans, F. Moraes. **Traffic Generation and Performance Evaluation for Mesh-based NoCs**. *Symposium on Integrated Circuits and System Design (SBCCI)*, pp. 184-189, 2005.

[16] E. Moreno, C. Marcon, N. Calazans, F. Moraes. **Arbitration and Routing Impact on NoC Design**. *IEEE International Symposium on Rapid System Prototyping (RSP)*, pp. 193-198, 2011.

[17] P. Mohapatra. **Wormhole Routing Techniques for Directly Connected Multicomputer Systems**. *ACM Computing Surveys, vol. 30, n. 3, pp. 374-410*, Sep. 1998.

[18] W. Dally, B. Towles. **Principles and Practices of Interconnection Networks**, *Morgan Kaufmann Publishers*, San Francisco, USA, 2004.

[19] J. Hu, R. Marculescu. **Energy- and Performance-Aware Mapping for Regular NoC Architectures**, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, n. 4, pp. 551-562, Apr. 2005.

[20] Y. Chen, Z. Lu, L. Xie, J. Li, M. Zhang. **A Single-Cycle Output Buffered Router with Layered Switching for Networks-on-Chips**, *Computers and Electrical Engineering*, vol. 38, Issue 4, pp. 906-916, Jul. 2012.