Analytical computation of packet latency in a 2D-mesh NoC

Sahar Foroutan[1,2], Yvain Thonnart[2], Richard Hersemeule[1], Ahmed Jerraya[2]
[1] ST-Microelectronics
[2] CEA-Leti
{sahar.foroutan, yvain.thonnart, ahmed.jerraya}@cea.fr
richard.hersemeule@st.com

Abstract

In order to avoid time-consuming iterations in the design flow of system-on-chip communication, it is required to make early decisions based on the performance estimations of the system. To fulfill such requirements, analytical techniques are fast and reliable alternatives for long and non-exhaustive traditional simulation-based methods. This paper presents a case-study of using an analytical method, based on Markov chain stochastic processes, for latency evaluation of a Network-on-Chip arranged in a two dimensional Mesh topology, with an x-first routing algorithm and an uniform traffic pattern. Results obtained by the analytical method are compared with the results of a SystemC simulation platform.

1 Introduction

Networks-on-Chips (NoCs) are new interconnecting solutions between different modules of nowadays complex System-on-Chips (SoCs). While the first published NoC was SPIN[1] designers have been proposing numerous architectures since then, like Aethereal[2], Mango[3], ANoC[4], Spidergon NoC[5] and so on. A vast number of parameters, such as topology, routing algorithm, switching strategy and flow control mechanism, make NoC architectures different from each other and influence the NoC performance. In addition to architectural parameters the traffic pattern (application) influences the performance of NoCs.

Due to growing Network-on-Chip complexity and communication parallelism and in the context of the increasing importance of a tight time-to-market, formal and analytical methods are finding their place in system-on-chip design flow. As elegant, effective and rapid alternatives for traditional simulation techniques, such methods are being used more and more for both functional verification and performance evaluation of complex communicating systems on chip. While formal techniques like theorem-proving, model-checking and equivalence-checking are used for verification purposes, mathematical techniques like queuing theory and stochastic processes, particularly Markov chains are widely used in performance evaluation domains. Also elegant formal methods and tools have been developed to integrate both aspects of functional verification and performance evaluation into a single model. Stochastic Petri Nets (SPN) [6] and Interactive Markov Chains (IMC) [7] are two examples of such methods. In the later a unique system model, expressed by LOTOS specification language [8], is convertible to either a pure state machine suitable for model-checking and equivalence-checking, or to a Continuous Time Markov Chain suited for performance evaluation purposes.

This paper presents a Markov-chain based method for determining the mean latency of the end-to-end communications via a NoC. Separating two kinds of delay; *link-transfer-delay* and *link-acquisition-delay*, our method approximates the latency of crossing each node (router) of the path between a given source and destination. We target the 2D-mesh NoCs in this paper. However our method is generic and completely adaptable with any NoC architecture.

The remainder of this paper is organized as follows. Section 2 presents the underlying theory of our method. Section 3 contains the implementation of the method as well as the experimental results. Conclusion and future work appear in section 4.

2 Analytical method for latency analysis

We have implemented an analytical method that returns the mean latency of the path between a given couple of source and destination via a NoC. Latency of the path is interpreted as the time needed by a packet for crossing the path. The NoC architecture targeted by our analytical method is arranged in a two dimensional mesh (2D-mesh) topology with an x-first routing algorithm which route packets first on the x direction and then on the y direction. The x-first algorithm is deterministic, deadlock-free and it guarantees the in-order delivery property of the network [9] [10]. The NoC is loaded under a uniform random traffic pattern which according to [11] is the most common traffic pattern used for NoC performance evaluations. For a *m*n* 2D-mesh NoC the uniform traffic is such that each source core sends the same fraction of a common offered load to any of the other *m*n-1* destination cores.

In order to find the mean latency of the path the method determines first the mean latency of crossing each node (and the output link connected to it) of the path and then the mean latency of the entire path is the sum of the nodes mean latencies. Nodes are abstract routers with five input-output ports. I/O ports are identified by their direction. So

there are south, north, east, west and core I/O ports per router. Nodes are identified with their (x,y) coordinates starting from upper-right corner with (0,0). Nodes are indicated with N_{xy} and cores with C_{xy}. The "latency of node N_{xy}" means the latency between the moment a packet header is at input port "*i*" of node N_{xy} and the moment it comes up to the input port of the next node (N_{next}). The latency of node N_{xy} is abbreviated with "*lat-i2o-*N_{xy}" and comprises the time needed for granting output "*o*" of N_{xy} (acquisition time) and crossing link N_{xy}➔N_{next} (transfer time). Index *i* and *o* can get values of n, s, w, e and c for north, south, west, east and core directions respectively.

2.1 Mean latency of a path

Fig.1 shows a vertical and a horizontal-vertical path in a 6*6 2D-mesh NoC, both beginning from the source $C_{2,1}$ and end to destinations $C_{2,4}$ and $C_{5,0}$ respectively.

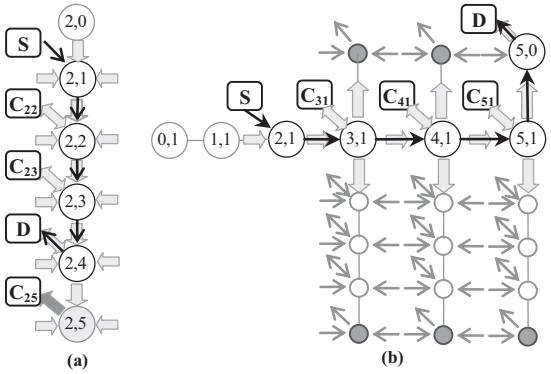


Fig.1 A vertical (a) and a horizontal (b) path in a 6*6 2D-mesh, x-first algorithm NoC

Narrow arrows depict the source-to-destination paths while wide arrows depict packets of other flows disrupting the latency the considered paths. Such packets are called "disrupting packets". The disrupting packets affecting a path are themselves disturbed by other ones. For example in fig.1.b disrupting packets forking toward south and north direction of the horizontal path are affected by flows showing by narrow gray arrows. According to the analytical method for obtaining the mean latency of the vertical path the mean latencies of traversing nodes N_{21} from core to south (*lat-c2s-*N_{21}), N_{22} and N_{23} from north to south (*lat-n2s-*N_{22} and *lat-n2s-*N_{23}) and finally N_{24} from north to core (*lat-n2c-*N_{24}) must be first determined. As well the mean latency of the horizontal-vertical path is obtained from the mean latencies of its nodes: *lat-c2e-*N_{21}, *lat-w2e-*N_{31}, *lat-w2e-*N_{41}, *lat-w2n-*N_{51} and *lat-s2c-*N_{50}. For determining each of these latencies we propose a Markov chain model as follows.

2.2 Markov chain model for node latency

Fig.2 illustrates the Markov chain model we have proposed for determining the mean latency between "*i*" input and "*o*" output of N_{xy} i.e. *lat-i2o-*N_{xy}. This is a generic model that is specialized for each pair of I/O ports and according to the position of the node an also the number of disrupting-packet inputs. Section 3 shows how distribution and mean latency are obtained from the Markov chain.

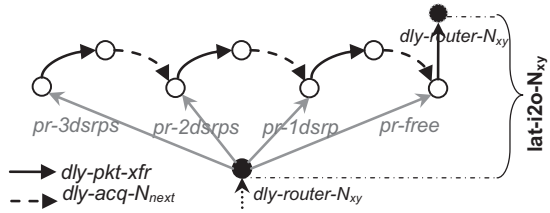


Fig.2: Markov chain underlying the latency distribution corresponds to traversing N_{42} from north input to south output link.

At input "*i*" of N_{xy} a packet addressing output "*o*" is probably stalled by one, two or three disrupting packets coming from other three input ports and addressing the same output. Each disrupting packet takes an amount of time for crossing the node. In order to express the probabilities of existence of disrupting packets and also the delays caused by them, the Markov Model is made up both probabilistic and delay transitions.

Probabilistic transitions are labeled with *pr-free*, *pr-1dsrp*, *pr-2dsrps* and *pr-3dsrps* representing probabilities in which output "*o*" is free, taken by one disrupting packet, two and three disrupting packets which are routed to output "*o*" before the packet coming from "*i*". According to the position of node in the NoC some of the probabilities may be zero. For example at north input of N_{25} a packet addressing C_{25} can be stalled with at most two other disrupting packets from east and west so the most left-hand probabilistic transition in the associated Markov chain is equal to 0. Probabilities are determined from routing algorithm and traffic matrix.

Delay transitions are of two kinds. The first kind, solid arrows depict the "*transfer delay*" which means the delay needed for the transfer of the disrupting packet through link N_{xy}➔N_{next}. It is labeled with *dly-pkt-xfr* and assumed to have a mean value equal to the packet length. The second kind, dashed arrows labeled with "*dly-acq-*N_{next}" represent the "*acquisition delay*" indicating the time needed by the header of the disrupting packet for acquisition of its output at N_{next}. In fact when a disrupting packet is transferred via link N_{xy}➔N_{next} (takes *dly-pkt-xfr*) it doesn't free N_{xy} until its header at N_{next} acquires the output (takes *dly-acq-*N_{next}). The approximation of acquisition-delay is complex and depends on latencies of a sequence of nodes. For example while a packet coming from the core input-port of N_{21} is waiting to grant the south output-link, the later may be occupied by a disrupting packet stalled itself at the north input of N_{22} by another one stalled at the north of N_{23} and so on. Such dependency between following nodes performs a sequence that is called *latency-*

dependency sequence (or dependency graph) and depends of the topology, routing algorithm, and traffic patterns.

When the output port *o* is equal to the core, the acquisition delay doesn't exist in the Markov chain because there is no next node. Cores are considered as sinks absorbing all traffics targeting them (with the pace of one flit per cycle) without any dependency on upper layers. So the latency between any input *i* and the core ($dly\text{-}i2c\text{-}N_{xy}$) can be obtained from a Markov chain similar to fig.2 but without delay transitions $dly\text{-}acq\text{-}N_{next}$. Furthermore for a deadlock-free routing algorithm, there is no cycle in the dependency graph. So starting from the final latencies of cores, the whole dependency graph can be reversely covered, and all dependent latencies are computed.

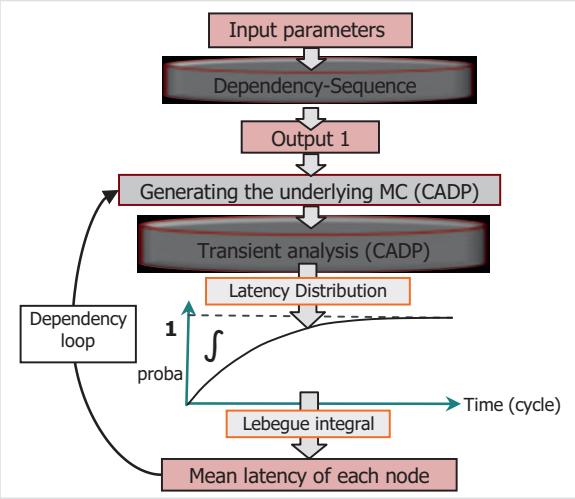


Fig.3 The implementation flow

3 Implementation of the analytical method

As we have described our analytical method aims to obtain the mean latencies of crossing each node of a given path, and by adding them achieves the mean latency of the entire path. For this we have implemented a tool taking as input parameters NoC dimensions, a source/destination pair, an offered-load and the packet length. Fig.3 illustrates the flow of our method. Beginning by determining the path between source and destination the tool constructs the dependency-sequence involved with the first node of the path and writes it in output 1. For each node of the dependency sequence probabilities of appearance of disrupting packets are calculated and written in the file.

3.1 Dependency loop

After that the dependency graph is written in output 1 through a loop starting from the end of the graph and according to the reverse order of the dependency, the tool builds Markov chains in order to determine mean latencies of dependent nodes. Generation and analysis of Markov chains are done by using CADP toolbox [12][13].The mean latency of each node is calculated (described in next section) and used in the construction of the Markov model of it backward node. In fact the dependency graph is covered in reverse order. Fig.4 shows the dependency graph involving with $lat\text{-}c2s\text{-}N_{21}$ of the vertical path (fig.1.a) ending at N_{25} ($Lat\text{-}n2c\text{-}N_{25}$) which is the starting step of the tool.

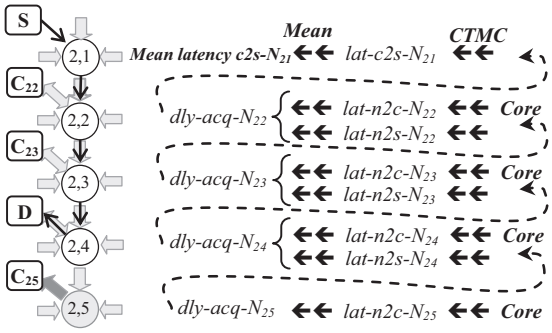


Fig.4 Analytical method for determining the mean latency between core i-port and south o-port of N_{21} ($Lat\text{-}c2s\text{-}N_{21}$)

Note that the latency of a node may be dependent on latencies of reaching different outputs of its following node. For example $lat\text{-}c2s\text{-}N_{21}$ in fig.4 depends on $dly\text{-}acq\text{-}N_{22}$ which is obtained from $lat\text{-}n2c\text{-}N_{22}$ and also $lat\text{-}n2s\text{-}N_{22}$. This is because a disrupting packet occupying link $N_{21} \rightarrow N_{22}$ can be routed to either the core or the south output of N_{22}. So our tool determines the latencies of reaching each of the possible outputs at the next node and takes the average as the delay needed for the acquisition of next node (here $dly\text{-}acq\text{-}N_{22}$). The average is obtained according to the probabilities in which each output direction is selected. Once the latency between the core and the south ports of N_{21} is obtained the latencies of all other nodes, required for obtaining the latency of entire path (i.e. $lat\text{-}n2s\text{-}N_{22}$, $lat\text{-}n2s\text{-}N_{23}$ and $lat\text{-}n2c\text{-}N_{24}$) are also obtained.

3.2 Mean latency between I/O ports of a node

There are tools that give the probability of being in a specific state of a Markov chain for a given time instant. This kind of probability is called *transient-state probability*. An example of such tools that we use in our implementation is the BCG_Transient of the CADP toolbox [12] [13]. Obtaining the transient probability of being in a particular state of a Markov chain for several time instants t_0, t_1, ... t_n performs the probability distribution (more exactly the Cumulative Distribution Function) of the time needed for reaching that state. In the same way the transient probability of being in the last state of the Markov chain of fig.2, for several time instants, gives the distribution of the latency between input "*i*" and output "*o*" of node N_{xy}, such as the curve of fig.3. By implying the Lebesgue integral on the curve of latency distribution, (hatched area in fig.3) we

are able to calculate the mean (expected value) of the latency (time duration of reaching the last state of the Markov chain).

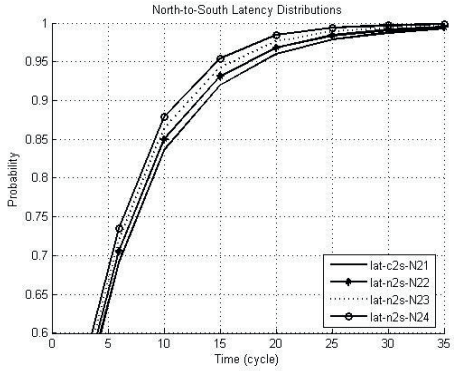
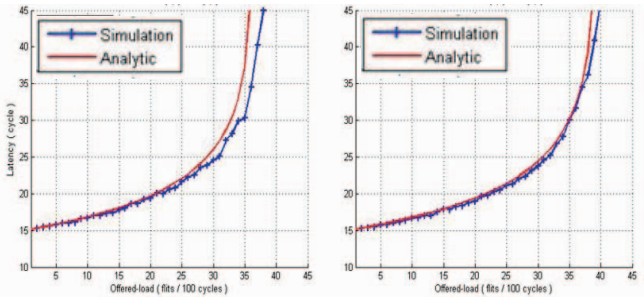


Fig.5 Latency distributions between the north input and the south output port of N_{21}, N_{22}, N_{23} and N_{24}

According to fig.4, 8 Markov models are constructed in order to achieve *lat-c2s-N_{21}*. Transient analysis of them gives latency distributions of core-2-south of N_{21}, north-to-south of N_{22}, N_{23}, N_{23} and north-to-core of N_{22}, N_{23}, N_{24} and N_{25}, the first four ones, obtained for 10-flit packet length and offered-load of 20% are illustrated in fig.5.

3.3 Latency-load curves

Repeating the analytical method for several amounts of offered-load, for a given path, provides the latency-load curve of the path.



(a) (b)

Fig.6 Analytical method vs. simulation results related to a) the vertical and b) the horizontal path of fig.1

Fig.6 (a) and (b) display the latency-load curves of the vertical and the horizontal-vertical paths of fig.1 respectively. The mean latency of both paths was examined for several offered-loads. Results obtained by our analytical method are compared with results of a corresponding SystemC CABA simulation platform. Mean packet length injected by each cores is equal to 10 flits. Offered-load is expressed with the average number of flits per 100 cycles. In the simulations for each path and for each offered-load we have taken the average latency of 1000 received packets. Modeling an infinite source buffer between each traffic generator and the network, indispensable during latency-load measurements [9, 11], enable us to observe the NoC saturation threshold. Without such buffers, generator stops the data injection every now and then and the real offered-load does not match the expected [11]. Once the NoC gets saturated, the source queues and consequently the average latency start growing exponentially. As can be observed, the inaccuracy of our approach (compared with the simulation results) is less than 5% for offered-loads less than 40%. After the offered-load 35% the NoC begins to be saturated and the latency tends to infinity. Our method predicts a saturation threshold of 37% for this NoC architecture.

4 Conclusion and future work

This paper has proposed a novel analytical method based on the Markov chain approach for obtaining the mean latency between two nodes of 2D-mesh NoCs in which packets are routed according to the x-first algorithm. For any dimension of NoC our method totally avoids the state-space explosion which is a very common problem when working with Markov chains. Unlike previously presented analytical methods for NoC latency, our method is rather simple and provides quite accurate results.

References

[1] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Proceedings of the conference on Design, automation and test in Europe* Paris, France: ACM Press, 2000.

[2] K. Goossens, J. Dielissen, and A. Radulescu, "AEthereal Network on Chip: Concepts, Architectures, and Implementations," *IEEE Des. Test,* vol. 22, pp. 414-421, 2005.

[3] T. Bjerregaard, "The MANGO Clockless Network-on-Chip: Concepts and Implementation." vol. PhD thesis: Technical University of Denmark, 2005.

[4] E. Beigne, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin, "An Asynchronous NOC Architecture Providing Low Latency Service and Its Multi-Level Design Framework," in *Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems*: IEEE Computer Society, 2005.

[5] M. Coppola, R. Locatelli, G. Maruccia, L. Pieralisi, and A. Scandurra, "Spidergon: a novel on-chip communication network," 2004.

[6] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, "Modelling with Generalized Stochastic Petri Nets," *ACM SIGMETRICS Performance Evaluation Review,* vol. 26, 1998.

[7] H. Hermanns, "Interactive Markov Chains and the Quest for Quantified Quality. LNCS 2428," Springer, 2002.

[8] L. Logrippo, M. Faci, and M. Haj-Hussein, "An Introduction to LOTOS: Learning by Examples," *Computer Networks and ISDN Systems,* vol. 23, pp. 325-342, 1991.

[9] W. J. Dally, *Principles and practices of interconnection networks*: Morgan Kaufmann, 2004.

[10] A. Sheibanyrad, I. Miro Panades, and A. Greiner, "Systematic comparison between the asynchronous and the multi-synchronous implementations of a network on chip architecture," in *Proceedings of the conference on Design, automation and test in Europe* Nice, France: ACM Press, 2007.

[11] E. Salminen, A. Kulmala, and T. D. Hamalainen, "On network-on-chip comparison," 2007, pp. 503-510.

[12] H. Garavel, F. Lang, and R. Mateescu, "An overview of CADP 2001," *European Association for Software Science and Technology (EASST) Newsletter,* vol. 4, pp. 13-24, 2002.

[13] H. Garavel and H. Hermanns, "On Combining Functional Verification and Performance Evaluation Using CADP," *LECTURE NOTES IN COMPUTER SCIENCE,* pp. 410-429, 2002