International Journal of Computer Applications (0975 - 8887)
Volume 137 - No.12, March 2016

Modeling and Performance Evaluation of 2D and 3D NoCs using Discrete Event Simulation

Nejib Mediouni, Samir Ben Abid, Oussama Kallel and Salem Hasnaoui
Communication System Laboratory SYSCOM
National Engineering School Of Tunis
University Tunis El Manar

ABSTRACT

Network on Chips are a method of interconnecting Processing Elements, such as processors and communication controllers, through a high scalability interconnect architecture. Planning and implementing NoCs is a complex task, and simulating them at the RTL level is time consuming which has motivated the implementation of a big number of cycle accurate and behavioral simulators. In this paper, we join the effort of NoC simulation platform implementation and we introduce a high level NoC simulation platform that is based on Mathworks Simulink and the SimEvents discrete event simulation engine. We, then, model a 2D and a 3D mesh NoCs using this method and we evaluate their performances. The obtained results are, then, validated using the booksim2 cycle accurate NoC simulator.

General Terms

NoC, Discrete Event Simulation, SimEvents, QoS.

Keywords

2D, 3D NoC, Latency, Throughput

1. INTRODUCTION

Smaller size transistors, and especially with the adoption of the FinFet technology for sub $22nm$ VLSI processes, did allow for the increased number of integrated cores and processors per die. In order for these latters to communicate without much latency, the master-slave interconnect method had to be replaced by a more powerful and extendable interconnect architecture that is the Network on Chip. Nowadays, Multiprocessor System on Chips (MPSoCs) rely on NoCs for information exchange between the processors and peripherals[1] instead of an interconnect such as OPB and PLB [2]. In a network on chip, the packets are transmitted through switches and routers using advanced routing algorithms and allocation schemes in contrast with the classical interconnect that act as master-slave bus. The architectural choices for NoCs have been the subject of numerous research works, which has yielded different architectural decisions based on constraints such as energy consumption, QoS metrics, and application specific needs.
Due to the complexity of the design and validation processes for NoCs, simulators are usually used during the design process in order to insure the suitability of the interconnect for the goal applications. Simulators are especially used for early stage design exploration and performance estimation.Simulators can be classified into three categories, CA (cycle accurate) simulators such as Orion [3], DARSIM [4] and [5], behavioral simulators such as OMNET++ [6] and BENoC [7] and high level stochastic simulators based on petri nets [8, 9]for example.
During the bibliographic review, we extracted the following metrics that characterize a NoC simulator.

—**Flexibility:** The simulator should not be locked into a single NoC architecture or topology. The implementation of alternative routing and scheduling algorithms should be possible.
—**User friendliness**: A usually overlooked aspect in most simulators. Modifying the source code of the simulator, which is the prevalent practice, reduces the flexibility of the simulator, and makes modifying the NoC structure and parameters extremely difficult and contrived.
—**Complexity:** Modifying the structure of the studied NoC should be easy, and shouldn't need delving into the inner-working of the simulator itself. Complexity extends, also, to simulation time.
—**Metrics:** The usual performance metrics, such as throughput, loss rate, etc... should be extractable. Used defined metrics should also be supported.
—**Performance:** Simulation time should be appropriate for the complexity of the NoC to simulate in order to accelerate the design cycle and speed-up simulation based decisions.
—**Validity:** The obtained results should mirror, to a certain degree, those obtained by the physical system.

In this work, our aim is to provide the aforementioned characteristics in the most user friendly manner which would simplify introducing modifications to the simulated NoC structure. The simulation methodology described within this paper is based on the (DES) Discrete Event Simulation. A DES tool reacts to events, than can be as simple as a rising edge or as complex as the end of a routing round of a flit and are well suited for sequential digital signal simulation. The granularity of the simulator is, thus, widely tunable, which mean that cycle accurate simulation is possible, but not always profitable in terms of implementation complexity or simulation cost, especially for DES as in has been shown in the work of Heid et al. [10], thus our choice of simulating the abstract behavior of the NoC without delving into clock cycle-accurate simulation.
As a proof of concept, 4×4 2D mesh NoC based on five-port switches and a $4 \times 4 \times 3$ 3D mesh NoC implemented using the

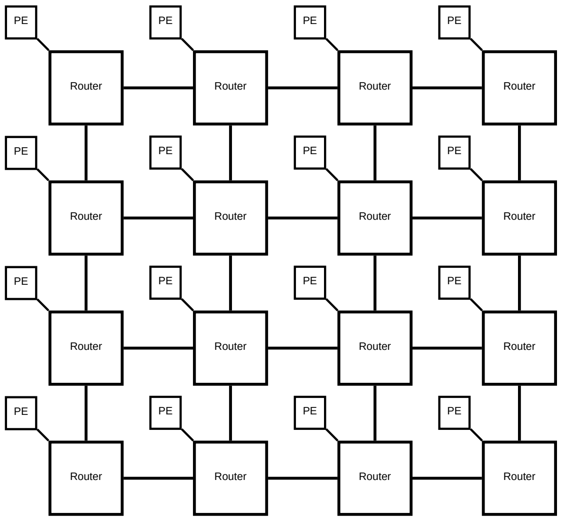


Fig. 1: NoC structure featuring five-port routers and PE blocs comprising the traffic generators and statistics collection functions.

SimEvents toolbox under Mathworks^{TM} Simulink. The routing functions are implemented using plain Matlab functions, and the performance metrics that were extracted are the packet loss rate, throughput, and average latency. A synthetic traffic generator was implemented that generates periodic, uniform, and MMPP (Markov Modulated Poisson Process) traffic patterns. The paper is structured as follows, the second section introduces the DES concept. In the third section, the implementation details are presented. The fourth section is consecrated to the simulation setting and the obtained results.

2. DISCRETE EVENT SIMULATION

In digital logic, systems are discrete by definition. A system state, which is described by the state of its registers, only change when certain events occurs. The most basic event in digital circuitry is the rising or falling edge of a clock. Between events, the system's state doesn't change. This makes digital systems well appropriate for Discrete Event Simulation [11].
The use of DES have been used in the simulation of multiprocessor systems in many works. One of the earliest examples is RSIM [12] that is aimed to simulate to study shared-memory multiprocessor architectures made of high instruction-level parallelism exploiting processors. OMNET++ [6] is a relatively recent example of a DES engine that have been re-purposed for NoC simulation [6, 13].

3. MATHWORKS SIMEVENTS

In this paper, we use the SimEvents toolbox found in the Mathworks^{TM} Simulink library. SimEvents is a discrete event that exploits the graphical programming and modeling environment Simulink. The authors chose SimEvents for two main reasons, the first being the tight integration between SimEvents and Matlab which provides a plethora of tools for results post processing and charting. The second reason is that SimEvents is well appropriate for modeling communicating systems as per its description [14] and the presence of specialized blocks such as FIFOs and switches. Furthermore, it is easier to modify a Simulink diagram than modify the source code of a C, C++ or SystemC based simulator for, virtually, the same effect. Here are some of the most important key concepts in SimEvents. An entity is the discrete item of interest for the SimEvents discrete simulation engine. It is characterized by attributes that can be scalars or vectors, and that can be changed or set in reaction to one or more events. The generation time of an entity is controllable, and can be made to obey a stochastic process. Entities can be queued and routed. Event are triggered by the change in the state of a signal, a function call or an event happening to an entity (advancement from on block to another, etc...).

4. 2D NOC ROUTER

As illustrated in Fig. 1 , the implemented NoC is comprised of 16 five-port routers arranged into a mesh topology, the used switching technique is store and forward.
Each router has a pair of connections for each ports denoted by $DataOut\_x$ and $DataIn\_x$, where x is the direction of the port (N for north, S for south, W for west and E for east). The fifth port is reserved for the NI (Network Interface), to which the synthetic traffic generator and the statistics collection blocks are connected. The PE blocks are connected to the input and output ports of the NI port. Each flit is represented by an entity, generated by a synthetic traffic generator, that must contain the *Destination* field. A user can add any additional field that may contain the payload for example. Each flit is timed from its generation to its arrival, which allows for the logging of timing based metrics (throughput, mean delay, etc...).
A router is composed of three stages, input queues, routing function and output crossbar (Fig. 2).

The input queues are modeled using a custom block called *Clocked FIFO*, that is a FIFO which output is enabled periodically. The output of the FIFOs are combined and then directed to a *Set Attribute* block that sets the field ME to the routers position within the mesh. The *Attribute Function* block reads the ME field and the Destination field in order to compute the necessary output field that is outputted as an additional field called RT. The XY routing algorithm [15] was implemented using a plain Matlab function (Alg. 1).
To simulate the routing delay, and prevent the occurrence of an algebraic loop, a *Signal Server* block, which service time is set to the said delay. Finally, the output crossbar is simulated by an *Output Switch* which output port is decided using the RT field. Due to the fact that SimEvents blocks can process parallel-time events, it is possible to output multiple flits simultaneously through the output switch similarly to a physical crossbar.
The router's block parameter dialog (Fig. 3) has five parameters which are the router's index, its input FIFO size, the routing and arbitration delay, the clock frequency at which it operates, and the standard deviation of the clock frequency which represents fabrication variability and thermal effects.

PEs are represented as the PE_i blocks, which are characterized by an address that is equal to that's of the router to which it is connected, in addition to some information about the NoC structure (list of the possible addresses, etc...). A PE block is capable of generating three types of synthetic traffic, periodic uniform and MMPP namely. It also acts as a flit sink and gathers the performance metrics.

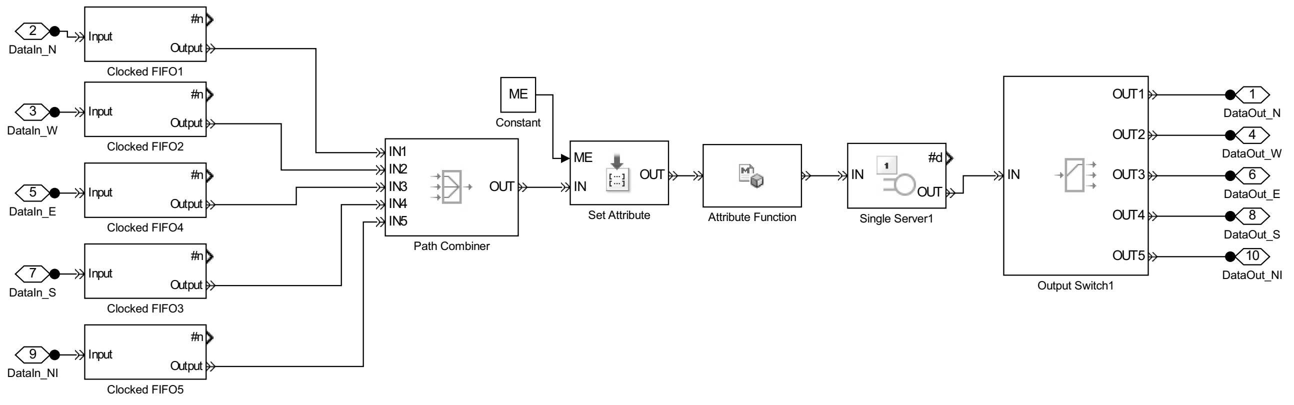
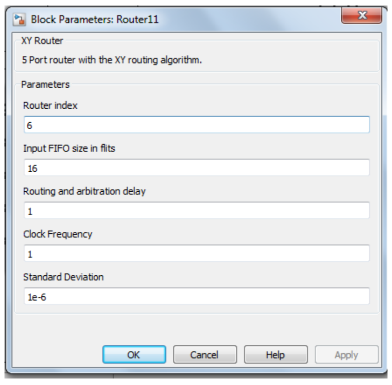


Fig. 2: SimEvents implementation of the 2D router, with ME being the router's address.


Block Parameters: Router11

XY Router

5 Port router with the XY routing algorithm.

Parameters

Router index

6

Input FIFO size in flits

16

Routing and arbitration delay

1

Clock Frequency

1

Standard Deviation

1e-6

Fig. 3: Properties dialog of the 2D router.

It is possible to simulate the presence of different jobs with different traffic patterns withing the PE. This is achieved by combining the output of a number of traffic generators using the *Path Combiner* block.
As for the link statistics, each PE block logs the instant arrival delay, mean arrival delay and loss rate using Matlab variables that are outputted to the workspace. The name of the variables are tunable using the PE block parameters window (Fig. 4).

5. 3D NOC ROUTER

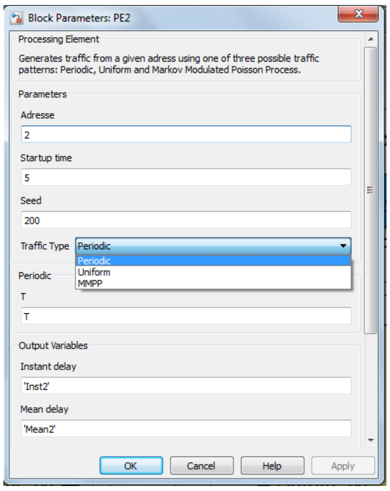
3D IC technology is already trending and is under active development, and have been proven to improve the performance of NoCs. In this case, the routers acquire two additional ports (Up and Down) that are connected using Through-Silicon Vias (TSVs) [16]. An $N \times M \times L$ 3D mesh NoC is equivalent to a $N \times (M * L)$ 2D networks but with some nodes being connected to one or two additional routers. Intuitively, 3D NoCs should perform better than 2D NoCs [17].

The structure of the 3D mesh NoC is based on seven bi-directional port routers that are arranged into a mesh configuration as illustrated in Fig. 5.The ports are labeled NI (Network Interface), North, South, East, West, Up and Down. The NoC is composed of three layers to which we would refer as the Down layer, Middle layer and Top layer.

The implementation under Simulink is illustrated in Fig. 6. Each layer of the NoC is implemented independently with only the Down

Algorithm 1 XY routing algorithm implementation within the *Set Attribute* block.

```
function out_RT  = fcn(DST,ME)

yi = floor(ME/4)+1;
xi = mod(ME,4);

if (xi==0),
    xi = 4;
    yi = yi - 1;
end

y = floor(DST/4)+1;
x = mod(DST,4);

if (x==0),
   x = 4;
    y = y - 1;
end

if x~=xi,
    if (x<xi),
        out_RT = 2;
    else
        out_RT = 3;
    end
elseif y~=yi,
    if (y<yi),
        out_RT = 1;
    else
        out_RT = 4;
    end
else
    out_RT = 5;
end
```



Block Parameters: PE2

Processing Element

Generates traffic from a given adress using one of three possible traffic patterns: Periodic, Uniform and Markov Modulated Poisson Process.

Parameters

Adresse: 2

Startup time: 5

Seed: 200

Traffic Type: Periodic

Periodic

T: T

Output Variables

Instant delay: 'Inst2'

Mean delay: 'Mean2'

Fig. 4: Properties dialog of the traffic generator.

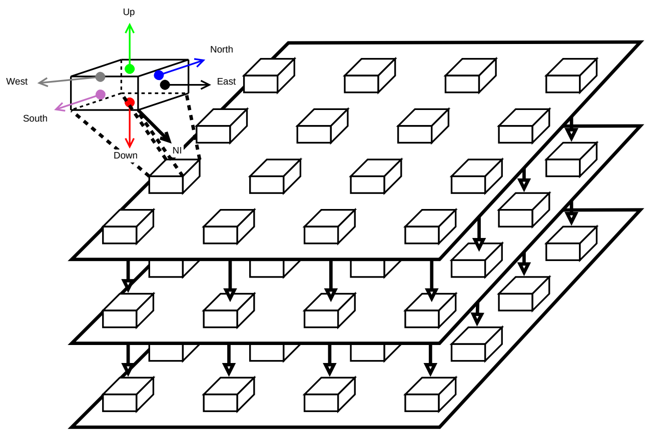


Fig. 5: 3D NoC structure featuring seven-port routers and PEs.

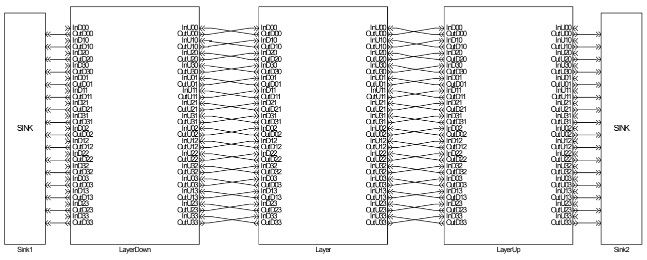


Fig. 6: Top level of the 3D NoC implementation.

and Up ports exposed. Each subsystem is configurable (clock frequency, layer index, etc. . .). The two Sink subsystems are used to detect erroneous routing to non-existing layers.
The routers can be classified into four categories, Corner, Side, Ring and Middle:

—**Corner:** three out of seven ports are connected to adjacent routers.

—**Side:** four out of seven ports are connected to adjacent routers.

—**Ring:** five of the seven ports are connected to adjacent routers.

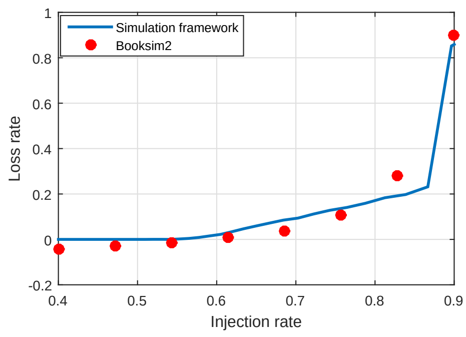


Fig. 9: Mean loss rate versus injection rate per core with periodic traffic injection.

—**Middle:** six of seven ports are connected to adjacent routers.

The routers composing the layers implementation is shown in Fig. 7. The input FIFOs contain the arriving entities representing the flits. The Route XYZ block sets a field, called RT, that indicates to which output port the flits are to be routed. The single server introduces the routing and scheduling delays, which are accounted in clock cycles. The flits are routed to the output ports using the Output Switch that relies on the RT field to determine the destination port.
In order to simulate the presence of PEs and collect the statistics, a synthetic traffic generator/traffic sink (STGTS). If the output of the traffic generator is blocked, the flit time-outs and is is accounted as a lost flit. Three traffic patterns are supported by the traffic generator:

—**Periodic:** Generates flits periodically. It takes one parameter, the intergeneration period.

—**Uniform:** Generates flits using with the intergeneration time a uniformly distributed random variable. It takes, as parameters, the minimum and maximum intergeneration period.

—**MMPP:** Markov Modulated Poisson Process. The rate of the modulating source (λ_1) and the modulated source (λ_2) are the only needed parameters.

The SimEvents implementation of the STGTS is presented in Fig. 8. If the need arises to combine different traffic patterns to simulate the presence of different jobs with different traffic patterns within a single PE, it is possible to combine multiple traffic generators using the Path Combiner Block.

6. SIMULATION RESULTS

6.1 Simulation Results of 2D NoC Router

We implemented a 4×4 mesh NoC in order to validate the platform. The implementation methodology is based on the flexibility of Simulink, so that placing the routers, ad the links is enough for the topology implementation. Each router is configured with an address that represents its location within the network, without reference to the rest of the nodes. To each router is connected a PE

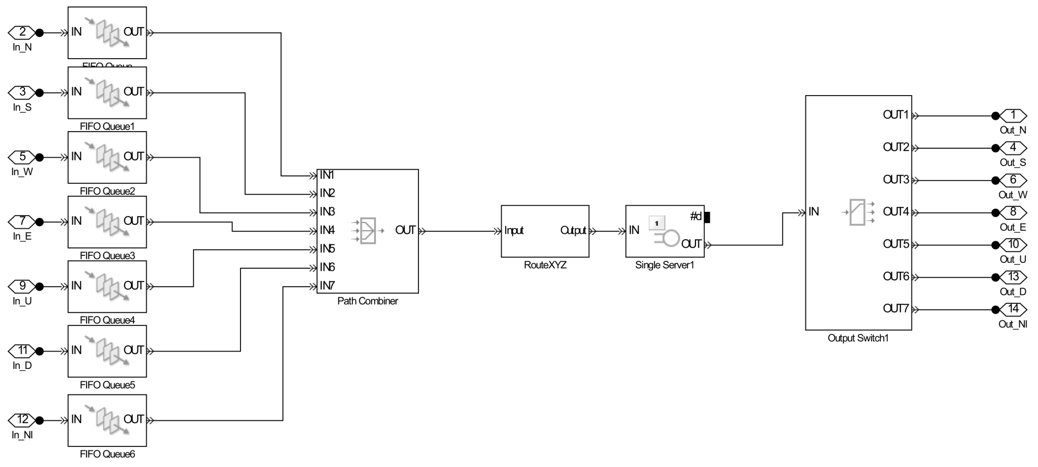


Fig. 7: SimEvents implementation of the 3D router.

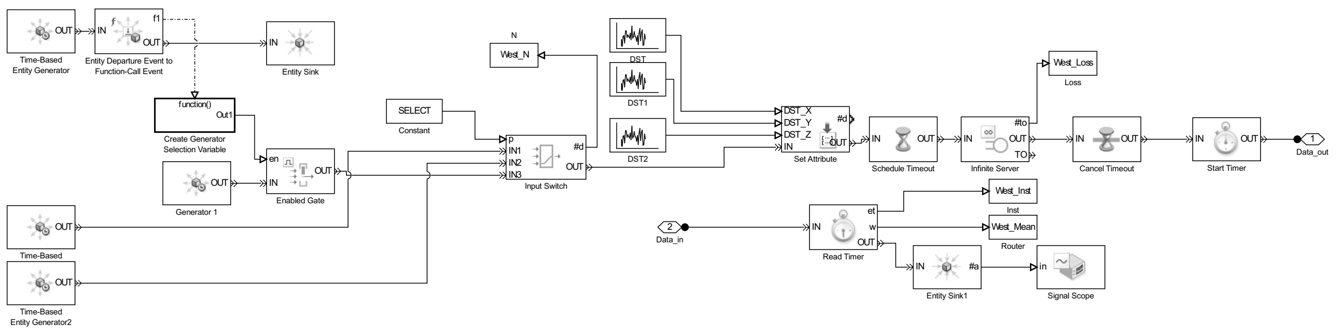


Fig. 8: Synthetic Traffic Generator/Traffic Sink (STGTS) implementation with SimEvents.

that acts as both a traffic generator and a traffic sink. For the simulation scenario, the traffic type is set to uniform, the considered clock frequency and jitter are $500MHz$ and 1% respectively, and the arbitration and routing delay are set to 10 cycles.

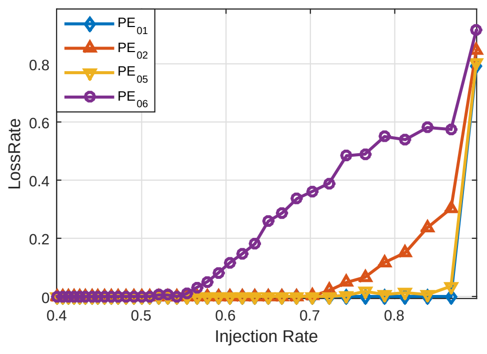


Fig. 10: Loss rate versus injection rate for PE_{01}, PE_{02}, PE_{05} and PE_{06}

The loss rate, or the rate of packets lost due to input buffer occupancy, of the simulated NoC is illustrated in Fig. 9.

For the corner, side and ring nodes, the loss rate is illustrated in Fig. 10. The global loss rate per core can be subdivided into three regions. For low injection rates, there is no losses. Beyond 53% injection rate, the loss rate becomes non-null an progresses in a linear manner with a slope equals to 0.75 approximately. Starting from 86% injection rate, the third region begins with a steeper slope (3 approximately). The results are close to those generated by booksim2 [5].

The main advantage of the proposed simulation framework is its flexibility and the possibility of gathering any type of relevant statistics from any node of the network. In Fig. 11, the local mean throughput traversing the corner, side and ring nodes is illustrated in function of time, with the results confirming that the edge and corner nodes are less stressed than the ring nodes. The loss rate varies according to the PEs position within the network as shown if Fig 10. The loss rate in function of the node position (Fig. 12) shows a similar trend for both 100% and 86% injection rates.

In order to demonstrate the statistics gathering capabilities of the simulator, the instant throughput of PE_{01}, PE_{02}, PE_{05} and PE_{06} are illustrated in Fig. **??**. We notice that the traffic across all these nodes is bursty, in contrast with the periodic injection traffic pattern.

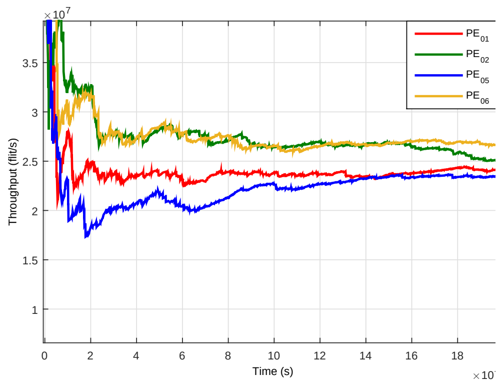


Fig. 11: Mean throughput of flit traffic tansiting through PE_{01}, PE_{02}, PE_{05} and PE_{06} for a 4×4 2D mesh.

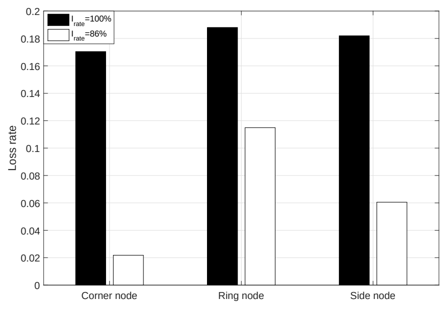
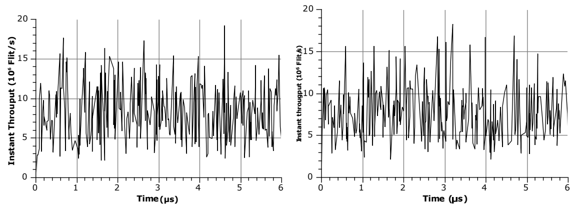
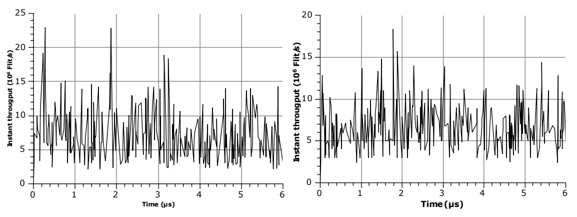


Fig. 12: Loss rate for the corner, ring and side nodes for 100% and 86% injection rates.



(a) PE_{01} (b) PE_{02}



(c) PE_{05} (d) PE_{06}

Fig. 13: Instant throughput as observed by PE_{01} , PE_{02} , PE_{05} and PE_{06}

6.2 Simulation Results of 3D NoC Router

In a similar manner, we implemented a $4 \times 4 \times 3$ NoC containing 48 routers and PEs arranged in three layers containing 16 routers and PE each. The simulation parameters are identical to those used for the 2D mesh in the previous section. 6.1

The obtained mean flit delay, as shown in Fig. 14, with a delay starting to peak at nearly 50% injection rate.

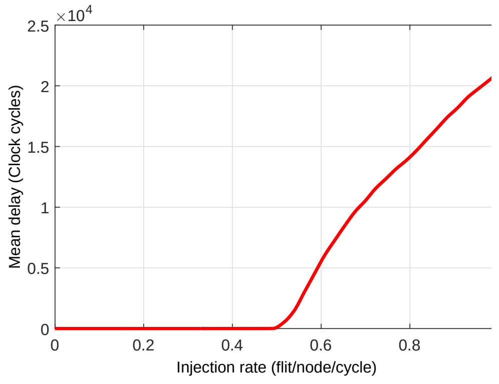


Fig. 14: Mean flit delay in clock cycles for the whole NoC.

As for the individual nodes, Fig. 15 shows that the Corner and Side nodes have close performances compared to the Middle nodes. This can be explained by the routing algorithm and the structure of the NoC. The XYZ routing algorithm prioritizes the movement withing the layer which could be reduced to an XY routing algorithm. Due to the remoteness of the Corner and the Side nodes, the Middle nodes get more traffic but the flits that are destined to them have to traverse less routers in order to reach them. The Z part of the routing isn't relevant in this case, as it only depends on the distance between the source layer and the destination layer.

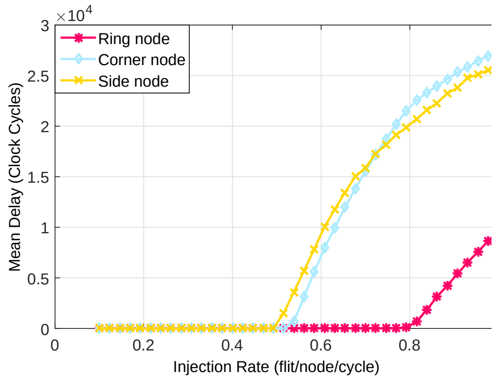


Fig. 15: Mean loss rate versus injection rate per core with periodic traffic injection.

7. CONCLUSION

In this paper we presented a simulation framework based on Mathworks SimEvents, the Discrete Event Simulation Engine. The framework is both flexible and easy to used. Two proof of concept NoC implementations, 2D 4×4 mesh and 3D $4 \times 4 \times 3$ mesh namely, were tested using the framework, and the simulation results for the 2D mesh have been validated using the Booksim2 NoC simulator. Parametric simulation is automated and covers a number of NoC parameters such as input FIFO depth, clock frequency, clock frequency variations due to thermal and manufacturing variability, routing algorithm, etc... The simulator is capable of measuring usual NoC performance metrics which are flit loss per injection rate, mean flit delay per injection rate and throughput per injection rate at a global and router level. Instant throughout and packet latency can, also, be measured providing additional information on the NoC behavior depending on its parameters.

8. REFERENCES

[1] Partha Pratim Pande, Cristian Grecu, Michael Jones, Andre Ivanov, and Resve Saleh. Performance evaluation and design trade-offs for network-on-chip interconnect architectures. *Computers, IEEE Transactions on*, 54(8):1025–1040, 2005.

[2] Graham Schelle and Dirk Grunwald. Onchip interconnect exploration for multicore processors utilizing fpgas. In *2nd Workshop on Architecture Research using FPGA Platforms*, 2006.

[3] Andrew B Kahng, Bin Li, Li-Shiuan Peh, and Kambiz Samadi. Orion 2.0: a fast and accurate noc power and area model for early-stage design space exploration. In *Proceedings of the conference on Design, Automation and Test in Europe*, pages 423–428. European Design and Automation Association, 2009.

[4] Mieszko Lis, Keun Sup Shim, Myong Hyon Cho, Pengju Ren, Omer Khan, and Srinivas Devadas. Darsim: a parallel cycle-level noc simulator. In *MoBS 2010-Sixth Annual Workshop on Modeling, Benchmarking and Simulation*, 2010.

[5] Nan Jiang, Daniel U Becker, George Michelogiannakis, James Balfour, Brian Towles, David E Shaw, John Kim, and William J Dally. A detailed and flexible cycle-accurate network-on-chip simulator. In *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*, pages 86–96. IEEE, 2013.

[6] Ruqaiya Al-Badi, Maha Al-Riyami, and Nasser Alzeidi. A parameterized noc simulator using omnet++. In *Ultra Modern Telecommunications & Workshops, 2009. ICUMT'09. International Conference on*, pages 1–7. IEEE, 2009.

[7] Dhiman Ghosh, Prasun Ghosal, and Saraju P Mohanty. A highly parameterizable simulator for performance analysis of noc architectures. In *Information Technology (ICIT), 2014 International Conference on*, pages 311–315. IEEE, 2014.

[8] Holger Blume, Thorsten von Sydow, Daniel Becker, and Tobias G Noll. Application of deterministic and stochastic petri-nets for performance modeling of noc architectures. *Journal of Systems Architecture*, 53(8):466–476, 2007.

[9] J. Silveira, P.C. Cortez, G. Cordeiro Barroso, and C. Marcon. Employing a timed colored petri net to accomplish an accurate model for network-on-chip performance evaluation. In *Quality Electronic Design (ISQED), 2014 15th International Symposium on*, pages 55–59, March 2014.

[10] Kris Heid, Haoyuan Ying, Christian Hochberger, and Klaus Hofmann. Latest: Latency estimation and high speed evaluation for wormhole switched networks-on-chip. In *Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2014 9th International Symposium on*, pages 1–7. IEEE, 2014.

[11] George S Fishman. Principles of discrete event simulation.[book review]. 1978.

[12] Vijay S Pai, Parthasarathy Ranganathan, and Sarita V Adve. Rsim: An execution-driven simulator for ilp-based shared-memory multiprocessors and uniprocessors. In *Proceedings of the Third Workshop on Computer Architecture Education*, volume 178. Citeseer, 1997.

[13] Yaniv Ben-Itzhak, Eitan Zahavi, Israel Cidon, and Avinoam Kolodny. Nocs simulation framework for omnet++. In *Proceedings of the Fifth ACM/IEEE International Symposium on Networks-on-Chip*, pages 265–266. ACM, 2011.

[14] SimEvents.

[15] Lionel M. Ni and Philip K. McKinley. A survey of wormhole routing techniques in direct networks. *Computer*, 26(2):62–76, 1993.

[16] Makoto Motoyoshi. Through-silicon via (tsv). *Proceedings of the IEEE*, 97(1):43–48, 2009.

[17] Brett Feero and Partha Pratim Pande. Performance evaluation for three-dimensional networks-on-chip. In *VLSI, 2007. ISVLSI'07. IEEE Computer Society Annual Symposium on*, pages 305–310. IEEE, 2007.