Rapid design and verification experience using flexible cycle-accurate NoC simulator

Mostafa Rizk, Jean-Philippe Diguet, Amer Baghdadi

Rapid design and verification experience using flexible cycle-accurate NoC simulator

Mostafa Rizk[‡§◇], Jean-Philippe Diguet[†], Amer Baghdadi [‡]
† CNRS, IRL CROSSING, Adelaide, Australia
‡ IMT Atlantique, CNRS Lab-STICC, F-29238 Brest, France
§ Lebanese International University, CCE Department, Lebanon
◇ Lebanese University, Department of Physics and Electronics, Lebanon

***Abstract*—Network-on-chip (NoC) has been introduced as a novel communication interconnection structure to overcome the limitations of traditional structures in terms of bandwidth, latency, and scalability. Quick feasibility assessment of various NoC structures demands the availability of high abstraction level simulation tools. However, existing NoC simulator solutions are either too slow to address the functional simulation of real-life applications or not flexible enough to allow NoC-based design space exploration. Furthermore, the necessity to run the application tasks while observing the data-dependent traffic is often missing. In this regard, the simulation tools modeling NoC interconnection structures have to cope with various NoC parameters and allow easy modeling of applications. In addition, these tools should be cycle-accurate and provide flexibility and capability of configuration and customization. This paper addresses the requirements of NoC simulator that enable rapid design and verification of NoC-based systems. It illustrates the design experience using flexible cycle-accurate NoC simulator to implement and verify emergent applications.**

***Index Terms*—NoC, MPSoC, simulator, design experience**

I. Introduction

Recently, multiprocessor architectures have been the key trend in implementing efficient hardware architectures to run complex and demanding applications. Such systems have been deployed in order to extend the applicability of Moore's law. Connecting several processors through a communication network grantees the enhancement of the system efficiency by exploiting the advantage of parallel execution of multiple processes. On the other hand, multiprocessor architectures suffer from different limitations such as inter-processor communication, synchronization, and load balancing.

System on chip (SoC) emerges by integrating all of the system components on the same chip. SoC provides closer coupling between system components and increases the yield of the chip fabrication. This reduces the overhead in terms of implementation area, power consumption and cost. The term multiprocessor system-on-chip (MPSoC) is commonly used to describe SoC which uses multiple processors. MPSoC incorporates the components necessary for an application such as processing elements (PEs) with specific functionalities reflecting the requirement of the expected application domain, memory blocks, timing resources, peripherals, and power management circuits in addition to the interconnection between all modules. MPSoCs embody an essential and distinct branch of multiprocessors [1]. They are not simply traditional multiprocessors fitted to a single chip but have been developed to fulfill the desired unique requirements of specific applications.

MPSoC technology has been increasingly emerging as the best effective solution for running complex and demanding applications. The use of heterogeneous multiprocessors bring an added value by allowing customization of processors to fit with the functionality of the different tasks of the embedded application. Such customization leads to fulfill the ever increasing requirements in terms of shrinked implementation area, enhanced performance and reduced energy consumption.

Nowadays, embedded applications include a large number of distinct processing tasks that communicate continuously. For example, the widespread video streaming application encompasses several tasks which require large storage and multitude processing elements to handle large amount of data. In addition, emergent applications support real-time processing with high data rates. Moreover, modern devices are multi-functional; i.e. execute concurrently different applications. These facts impose that a large number of diverse modules have to be used to constitute the computational resources of MPSoC. An efficient interconnection strategy has to be adopted to ensure fast and robust communication infrastructure between various resources integrated on the chip.

Different on-chip communication modes have been recently devised. The traditional on-chip communication structure consisted of conventional bus, which is a common wire branch that interconnects multiple system modules with the aid of an arbiter that manages the access to the bus. Although different generations have been developed such as buses with crossbars and decoupling buses, bus structures suffer from limitations in terms of latency, bandwidth, and poor scalability. Latency arises when in progress transactions lead to stalling high priority accesses. Bandwidth limitation is due to the clock-frequency which refers to the critical path that depends on the length of the wires. Furthermore, for additional connected modules communication bottleneck is reached and arbitration becomes complex due to the bus poor scalability. Thus, shared bus can not be considered a long term solution for on-chip communication. On the other hand, dedicated point-to-point connections approach ensures optimal solution for bandwidth, latency, and power consumption. Nevertheless, the number of independent links increases exponentially with the increase of

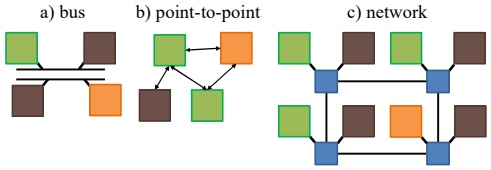

Fig. 1. SoC communication modes: a) traditional bus b) dedicated point-to-point links c) network

the connected modules. This leads to overhead in terms of routing and implementation area. Fig. 1 (a) and Fig. 1 (b) presents respectively the traditional bus-based and dedicated point-to-point communication modes.

The structure of the on-chip communication links becomes more critical as the number of modules increases. In fact, in MPSoC, the communication needs rise dramatically to the reach the point of preventing the system to meet with the desired requirements in terms of performance, latency, implementation area, and energy consumption. In order to circumvent the communication bottleneck, network-on-chip (NoC) has been introduced as candidate solution to implement the interconnection architecture of MPSoCs [2] [3] [4] [5]. Several advantages of structure, performance, and modularity is attained when using a network to replace global wiring [2]. NoCs ensure better performance, bandwidth, and scalability when compared to bus-based and dedicated point-to-point interconnection architectures [6]. The NoC architecture serves as an embedded network of interconnected switches with resources connected to these switches. A resource can be a computation unit (embedded processor, DSP core, FPGA block, intellectual property (IP) block etc.), storage unit (RAM, ROM, etc.) or their combination [5]. Whereas, a switch routes and buffers messages between resources.

Functional verification of MPSoC and performance evaluation are crucial processes in chip design cycle. Both processes play an important role in reducing development costs and in getting designs out faster. Often verification and evaluation processes are performed using logic simulation and/or FPGA prototyping. Despite their effectiveness, both methods are based on the hardware description language (HDL) source code (Verilog or VHDL), which is error-prone and requires long development time in case of large and complex designs. In addition, logic simulation is often not fast enough for large designs and hardware verification operates slowly when probes are added directly to the RTL design. Hence, the increase in design complexity and the increase in pressure to validate designs out faster impose to model a complete SoC at a high abstraction level prior to modeling it with lower level abstractions. High abstraction level modeling is performed before HDL to confirm the correctness and feasibility of the system to be implemented leading to significant reduction of the time and efforts expended for verification as compared to HDL-based methods.

Recently, multitude simulation tools have been developed to model NoC such as Garnet [7], SICOSYS [8], ATLAS [9], Booksim2 [10], DARSIM [11] and Noxim [12]. These simulators differ in their corresponding features regarding supported flexibility, cycle-accuracy, capability of configuration and extension, and provided results and statistics. Other existing multi-core system-level architecture and processor micro-architecture simulators such as gem5 [13][14] and Sniper [15] are wildly used in research activities. However, these simulators are either too slow to address real-life applications or not flexible to allow NoC-based design space exploration. The literature contains many surveys that discuss available NoC simulation tools including comparative studies [16][17][18][19]. However, a comprehensive study of exiting related works highlighted the lack of analysis of NoC simulation requirements. The objective of this paper is to specify the set of requirements for the simulation tools that model NoC structures in order to facilitate research in on-chip communication for researchers and developers when experimenting emerging technologies. These requirements are specified according to the authors practical expertise in using NoC simulators for a range of advanced applications [20] [21] [22] [23]. Two case studies are presented to illustrate the importance of exploiting simulating tools embedding the specified requirements.

The rest of the paper is organized as follows. Section 2 presents an overview about NoC structure. Section 3 discusses the simulator requirements. Section 4 presents two application case studies where NoC simulator that meets the requirements is utilized. Finally, Section 5 concludes the paper.

II. NoC BACKGROUND

This section provides an overview about NoC structure, basic properties that contribute in characterization and classification of various NoCs, and the performance parameters considered in evaluation.

A. NoC structure

A network-on-chip typically consists of three main blocks: the links, the routers, and the network interfaces (NIs) [24][25]. The links interconnect physically the nodes and actually implement the communication. The NIs establish the connection between the network from one side and the storage resources or processing elements from other side. The router, which implements the communication protocol, performs several functionalities: (1) receiving incoming packets; (2) storing packets; (3) delivering these packets to a given output port; and (4) sending packets to others routers. Fig. 1(c) illustrates NoC architecture.

B. NoC basic properties

1) Topology: It is the structure of the routers connections in a network and it is mainly represented by a graph that relates the routers and channels. Topologies are classified into direct or indirect topologies [26]. In direct topology, nodes are created by coupling the routers to a computational or storage resource. Whereas in indirect topology, some routers are utilized only to propagate the messages through the network, while others are associated with PEs or storage units. Recently, various topologies for NoC architecture have been proposed including mesh, torus, ring, butterfly, octagon, star, tree, etc. [27]. Irregular topologies can be obtained by adopting several topologies in a hierarchical, hybrid, or asymmetric fashion.

2) Routing algorithm: It specifies the path of a packet between the source and the destination nodes [28]. Relying on the algorithm logic, an output port is selected to forward an arriving packet according to the routing information provided in the packet header. In the literature, several routing algorithms are available and are classified according to different criteria: deterministic or adaptive, static or dynamic, unicast, multicast, or broadcast, and minimal or non-minimal [28] [29].

3) Arbitration logic: It is the control mechanism that selects the input port when several packets reach the router at the same time instant and request to reserve the same specific output port. The arbiter implementation can be distributed or centralized and can rely on static or dynamic priorities among ports [24] [29].

4) Switching: It determines the way data is transferred between the source destination nodes. Two main switching approaches exist: the circuit switching and packet switching. In the circuit switching, the payload is not sent until all the total path is previously determined and reserved; whereas, in packet switching, the payload is forwarded to next router when the connection is establishes between two neighbor routers.

5) Forwarding strategy: Three forwarding strategies are commonly used: store-and-forward, wormhole, and virtual cut-through. The node has to store all the packet prior forwarding it to the next node when store-and-forward strategy is adopted. On the other hand, in wormhole strategy the payload follows the header to next node, which identity is determined locally in the current node. Virtual-cut-through strategy only differs from wormhole strategy by that the current node demands a confirmation that all payload to be sent can be accommodated in the next node.

6) Flow control policy: It is defined as the mechanism that determines the packet movement along the network path [30]. Flow control is adopted to ensure the correct operation of the network and optimal utilization of its resources. Relayed policies are classified into centralized (NoC-level) or distributed (router-level). Virtual channels concept is the foremost flow control policy that is used to avoid deadlocks, optimize wire usage, and enhance overall performance.

7) Buffering policy: It is the method utilized to save information in the router when congestion occurs in the network such that packets cannot be transferred. These methods differs according to the number, location and size of the buffers. There are two main buffering methods: implementing a single buffer in the router which is shared by all input ports, or implementing one buffer per port.

C. NoC Performance Parameters

1) Bandwidth: It is the maximum rate of data transfer in the network and it is usually measured in bit per second.

2) Latency: It is the time duration between sending the packet from the source node and receiving it at the destination node. Latency is measured in time units (clock cycles). Mainly average latency of all packets is computed rather than single packet latency.

3) Throughput: It is the maximum amount of information delivered to the network per time unit. It is measured in bit per time unit (clock cycle) per node.

Focusing on a single performance parameter is not sufficient to evaluate the performance of the NoC. Latency-throughput curve is commonly used to measure the performance of interconnection networks.

III. Simulator requirements

The evolution and adoption of emerging NoC technology depends on the availability of accurate modeling that allows to conduct real simulations required to evaluate the performance and estimate the overhead of proposed NoC designs. This section specifies the needed requirements of NoC simulation models to enable rapid verification leading to rapid prototyping of NoC-based systems.

A. Accuracy

Accuracy is an essential feature of NoC simulators since these models have to demonstrate the state of the modeled NoC at every time instant. Although cycle accurate simulations may need more time to be accomplished, they does not introduce any approximation. Thus, the obtained results are exact and represents the actual performance of the modeled NoC. HDL-based RTL model, as in [31], are accurate but can take hours to process many cycles. Lately, high-level hardware verification languages have been introduced with the growing complexity of chips. These languages mimic the HDLs and provides real-time environment, which supports system-level modeling, architectural exploration, performance modeling, and functional verification.

B. Flexibility

Flexibility is an important feature for simulation models especially for research uses. As mentioned in previous section, the NoC parameters are multitude. The NoC specifications may encompass a wide range of parameters related to topology, dimension, routing algorithm, flow control policy, switching, buffering policy, etc. Thus, characterizing different NoC architectures leads to performance and design trade-offs. The simulation model should be powerful for architectural exploration which demands the varying of the parameters and characteristics of an architecture and then evaluate correspondingly the change in the obtained results. In addition, flexibility requires to enable the user to devise his own specifications and add it to the model. For example, a user interested in experimenting a new irregular topology form can develop its code and easily add it to the model.

On the other hand, flexibility supports iterative refinement and improvement of the tailored architecture. Hence, the designer can start with an initial parameters characterizing the NoC structure or the functionality of the connected resources. Then, the designer can iteratively modify the parameters and use the simulation model in order to obtain a new set of results. In each iteration, the designer can focus separately on a specified design metric (clock, parallelism, power consumption, implementation area, injection rate, latency, etc.). At the end, taking into consideration the obtained results, the designer chooses the architecture that fit most with the requirements.

C. Functional simulation

One important requirement of a NoC simulator is the capacity to address the functional simulation of real-life applications. The simulator should enable the user to describe and integrate easily the functions of different tasks of the application. It should also allow efficient execution of these tasks. Furthermore, the user should be able to choose and modify the scheduling method to run the implemented tasks. In this context, the necessity to run the application while observing the data-dependent traffic is a key requirement which is often missing in existing NoC simulator solutions.

D. Extendability

Extendability refers to the ability to implement new components to the modeled architecture. It allows the designer to develop new modules and add them easily to the simulator and then conduct simulations without the need to modify the remaining parts of the simulator. If the simulation model is characterized by modularity, its modular architecture allows the designers to implement their hardware and software components simply. Hence, the functionality and behavior of each added module (PE, IP cores, memory, etc.) can be individually described.

E. Configurability

Configurability deals with the ability to adjust individual NoC components in order to mimic the behavior of special architectures. Unlimited configurability allows the designer to change the modularity, functionality of NoC blocks such as packet maker and unmaker, adapters, etc.

F. Programing language

The programing language utilized in modeling the simulator has several requirements. If the simulator is based on standard programing language, then the users can describe their configurations and extendable modules without the need of acquiring expertise in particular HDLs. In addition, this capitalize on the extensive and available infrastructure of compilation and debugging tools. Furthermore, an object-oriented programing language-based simulator would enhance design productivity and facilitates re-usability through making use of capabilities such as inheritance and templates. This feature allows rapid verification of NoC-based systems. Also, the framework of the selected language must support fast simulation speed and reduced system requirements especially in terms of memory and CPU.

G. Provided results and statistics

In order to promote the evaluation methodology of studied systems, an NoC simulator model have to provide several performance metrics such as average latency, injection rate, number of switch conflicts, average hop number, and estimated energy dissipation. In addition, the simulator flexibility should allow the designer to extract his own comparison parameters.

Offering these requirements increases the life cycle of the simulator, as the researches are continuously looking for more powerful simulators, with more computational capabilities and degrees of modularity and accuracy.

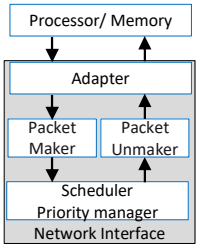

Fig. 2. The generic structure of the network interface in the simulator

IV. Rapid verification experience using NoC simulator

In our work, the feasibility assessments have been conducted for multitude NoC-based MPSoC architectures in various domains such as smart memories, dynamic run-time remapping of tasks over heterogeneous platforms, and memory-based computation. The simulator is characterized by its cycle-accuracy, flexibility, extendability, and modularity which are important features especially for research uses. Furthermore, the simulation model permits to configure the NoC components and the interconnection structure. The model has been developed using SystemC, which has been introduced as a system-level modeling language based on the well-known object oriented C++ language that guarantees accuracy of models and evaluation results [32]. SystemC enables the designer to build systems with hierarchical modules that can work concurrently and intercommunicate via ports using simple or complex communication channels.

The devised simulator model allows to define the number of processors and memory elements and to attach them to specific routers. Each router is considered to have five ports to interconnect with other routers and to attach memory or processing elements. For each router, the routing algorithm can be selected as well as for the arbitration. Each node consists of a network interface and memory/processing element. The structure and size of memory modules can be specified. The processing elements can execute several functions according to the predefined scheduling method. Fig. 2 illustrates the generic structure of the network interface adopted in the simulator. The back-end part of the NI includes a packet maker and un-maker, which are used to assemble and disassemble the packets, and a priority manager to synchronize packet transmission and reception. The front-end of an NI includes an adapter that can be configured in order to implement application-specific functions. Fig. 3 depicts a typical 4×4 2D mesh NoC modeled using the simulator.

In the following subsections we demonstrate how the simulator features enable the rapid design and verification of Notifying memory concept and NoC-MRAM architecture for memory-based computing.

A. Notifying memory concept

Notifying memories (NM) concept has been introduced in [21] to reduce the overhead of increased communication latency due to NoC adoption. NM concept eliminates useless memory accesses to decrease latency which is penalizing for data-flow applications. The devised approach transforms memories into masters, which notify the processing elements

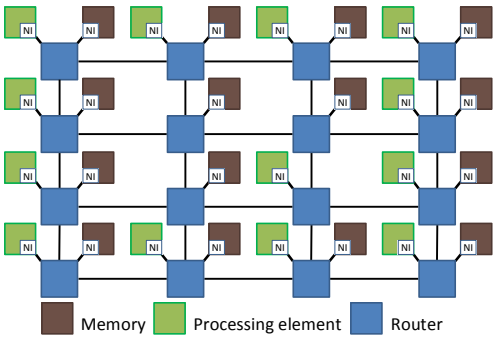

Fig. 3. Typical 2D mesh NoC with its resources

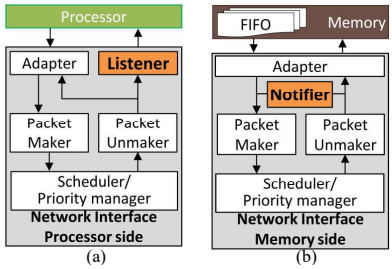

Fig. 4. The structure of the NIs adopted for processor and memory nodes to demonstrate the NM concept

whenever the required data is ready. In order to assess the NM concept relevancy, the NoC simulator has been extended by augmenting memory modules with notification and processing elements with listening mechanisms. These mechanisms are implemented in the network interface (NI) in order to be independent from memory and processor architectures. The structure of the NIs adopted for processor and memory nodes to demonstrate the NM concept are illustrated in Fig. 4. Readers interested in the detailed architectures of the notifier and listener module can refer to [21].

The utilized NoC is a 4×4 2D mesh-based network similar to that presented in Fig. 3. It interconnects 13 PE and 15 memory nodes. It adopts deterministic XY routing algorithm, wormhole packet switching mode, and flow control policy without virtual channels (VCs). In the routers, one arbiter is coupled per each port and only one buffer is allocated for each input port.

On the other hand, the PEs have been modeled to execute the MPEG-4 part 2 Simple Profile decoder (MPEG4-SP), which is developed by the Moving Picture Experts Group. The structure is partitioned into three parts, which each corresponds to a dedicated processing: parsing, residual decoding and motion compensation. The parser separates the processing of each image components in three parallel paths (Y, U and V) in order to increase the parallelism. At the end of the processing, the image components are then merged back. In this work, the ORCC tool is utilized for compiling and software synthesis [33]. Our work makes use of the generated C-code for multi-core platforms. The adopted simulator tool allows to functionally simulate real applications. We exploit this feature in order to run the real MPEG application and obtain accurate results concerning data transfer and timing. The configurability feature of the adopted simulator allows us to modify easily the structure of the NIs to mimic exactly the real architectures of the notifier and the listener modules.

TABLE I
NOTICATION MEMORY GAIN FOR DECODING DIFFERENT VIDEO SEQUENCES [34]

Video sequence	bridgefar	grandma	bus	bus	foreman
Format	QCIF	QCIF	QCIF	CIF	CIF
# Frames	2099	870	150	150	300
Latency	-67.22%	-68.55%	-67.44%	-75.14%	-74.92%
Througput	+9.69%	+9.32%	+9.58%	+9.04%	+9.93%
Injection rate	-45.90%	-53.96%	-33.03%	-34.35%	-47.53%
Switch conflicts	-70.81%	-75.88%	-61.87%	-62.38%	-71.62%
Flits number	-51.14%	-58.25%	-39.45%	-40.29%	-52.74%

Multiple simulations have been conducted using the simulation model in order to decode several real-life video sequences from [34] with different formats and characteristics. In addition, the results obtained when adopting NM are compared with the ones obtained when adopting ordinary memories while using identical NoC features. Table I shows the comparison results in terms of latency, throughput, injection rate and switch conflicts. The comparison shows significant reductions in terms of latency, injection rate , switch conflicts and total number of flits along with remarkable throughput improvement.

B. NoC-MRAM architecture for memory-based computing

NoC-Memory Based Computing (NMBC) architecture has been introduced in [22] and [23] as an efficient solution for memory-based computing. The novel architecture relies on a NoC and power-gated distributed magnetoresistive random access memories (MRAMs). The feasibility of the proposed approach has been demonstrated through a relevant case study of a database application implemented with neuromorphic architecture based on sparse-neural-network (SNN). The SNN depends on two distinct phases: (1) the message learning and (2) the information retrieval. In the learning phase, each field of each record is mapped to a single neuron in each cluster. In the information retrieval phase, the best candidate neuron for each cluster that corresponds to an unknown field is chosen by means of known neurons in other fields. This selection is done according to the winner-take-all ranking algorithm.

The tailored architecture is based on an NoC that interconnects three types of IP blocks: memory modules, processing elements and managers. Each memory module encompasses a cluster of memory blocks that store the connection information between the neurons of all clusters at the learning phase. The manager is responsible for processing requests and sending them as packets to memory clusters. The processing elements apply the winner-take-all computational principle in order to find the neuron with the highest score.

To model the architecture, the utilized NoC simulator has been modified to interconnect 6 memory modules, 11 PE, and 1 or 2 manager(s) as shown in Fig. 6. The adopted NoC is a 4×4 2D mesh-based network that uses a wormhole packet switching mode, a deterministic XY routing algorithm, and a flow control policy without VCs. The implemented routers have one buffer per input port and use one arbiter per output port.

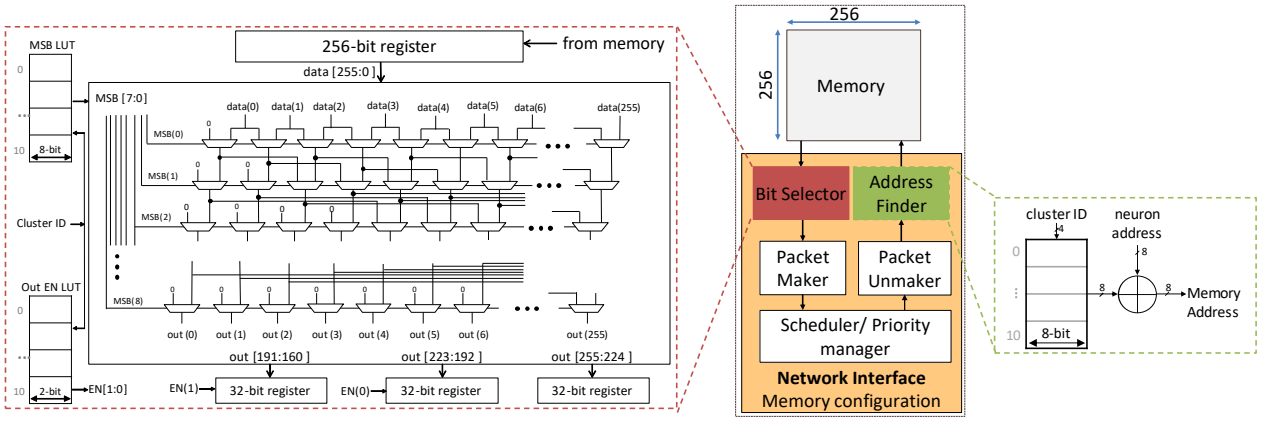

Fig. 5. The structure of the configured NI employed in NMBC architecture

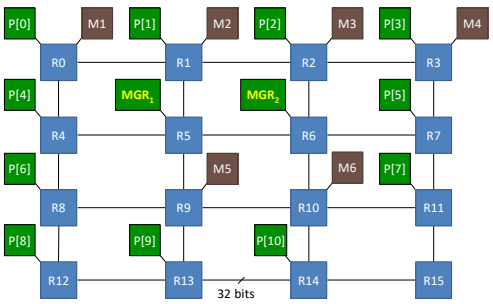

Fig. 6. The structure of the NoC employing power-gated distributed MRAMs

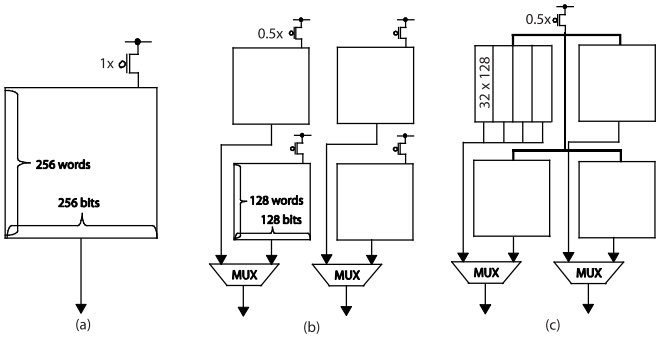

Fig. 7. Power-gating types and granularity in 256-bit × 256-word MRAMs: (a) Type I, (b) Type II and (c) Type III [23].

Three different types of 256-bit × 256-word MRAMs (Type I, Type II, and Type III) with different power gating (PG) policies have been modeled. These types are shown in Fig. 7. Type I controls one 256×256 MRAM; whereas, Type II and Type III control four 128×128 MRAMs. Type I adopts one PG transistor for a 256×256 MRAM. Type II adopts four PG transistors for four 128×128 MRAM subblocks. The maximum write bitwidth for Type III is 32 bits, whereas the maximum read bitwidth is 256 bits as for Types I and II. Adopting PG reduces the leakage current of the cell array compared to that of an SRAM cell array. The peripheral circuits can be power-gated using the pMOS transistor when the MRAM is at the idle state. In this work, two different policies of PG have been addressed: only cell PG (OCPG) and full PG (FPG). In OCPG, the the MRAM cells are power-gated at the idle state and the peripheral circuits are always at the active state. IN FPG, the whole MRAM is power-gated at the idle state. FPG reduces the leakage current in comparison with the OCPG while a wake-up operation is required for the peripheral circuits, which cost extra energy dissipation and delay time. For more information about the adopted MRAM memory types, interested readers can refer to [23].

Furthermore, the PEs have been modeled to execute the winner-take-all computational principle. In addition, the structure of NIs associated to each memory module is upgraded in order to decode the requests sent by the manager, handle the communication of results to the processors, and provide managers with monitoring data about bandwidth usage and processor usage. These services are dedicated to NIs in order to remain compliant with any existing memory and to be independent from NoC parameters. Hence, the front-end of the NI has been configured to encompass the functionalities of a bit selector and an address finder as shown in Fig. 5.

The configurability of the adopted simulator allows to model the desired structure of the memory modules and NIs. Its flexibility has been exploited to test the application using one and two managers. The simulator enables to track the transferred data and allows to check the content of transmitted and received packets. Accordingly, precise hit rates are computed. The simulator accuracy allows to determine precise timing information required to calculate the energy consumption. Several simulations have been performed over the MRAM-based computing architecture targeting hundreds of database queries of the Yeast database (Cellular Localization in Proteins) from the UCI Machine Learning Repository [35]. In order to evaluate the efficiency of the proposed NoC-MRAM architecture has been compared to SRAM ones while using identical NoC features. The results show hit rates of about 95%. Fig. 8 presents the energy consumption for different memory types and PG policies. The comparison shows an impressive power reductions when compared to SRAM for all cases with FPG policies.

V. Conclusion

In order to satisfy the tight constraints on implementation area and power consumption and fulfill nowadays requirements in terms of performance and throughput new design paradigms have been devised in designing MPSoCs which

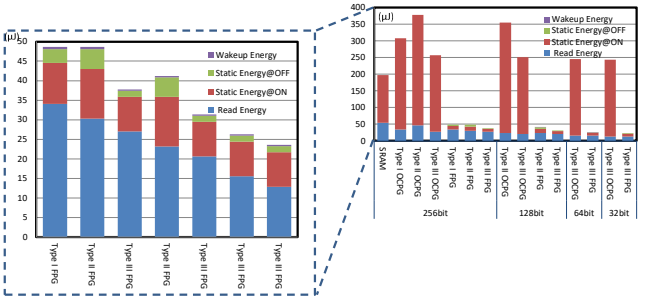

Fig. 8. Energy consumption for different memory types and PG policies with zoom on FPG policies cases.

incorporate modularity and explicit parallelism. In this context, NoC is considered as a key concept for interconnecting emergent MPSoCs. However, NoC structure encompasses a wide spectrum of specifications and customizations, which lead to trade-offs with regard to latency, energy dissipation, throughput and implementation area. On the other hand, the emerging applications adopting NoC-based architectures are becoming more complex. Hence, exploring the design space of a NoC and assessing its performance require the availability effective simulation tools. In this paper, the requirements of an effective simulation framework have been addressed. Simulators endorsing these requirements are pioneer models for architectural exploration and for assessing the performance of multiprocessing platforms mapped to different NoC interconnect architectures. Design experiences using flexible cycle-accurate NoC simulator to implement and verify emergent applications have been illustrated.

REFERENCES

[1] W. Wolf, A. A. Jerraya, and G. Martin, "Multiprocessor system-on-chip (MPSoC) technology," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 10, pp. 1701–1713, Oct. 2008.

[2] W. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. of the Design Automation Conference (DAC)*, June 2001.

[3] D. Wingard, "Micronetwork-based integration for socs," in *Proc. of the Design Automation Conference (DAC)*, June 2001, pp. 673–677.

[4] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vincentelli, "Addressing the system-on-a-chip interconnect woes through communication-based design," in *Proc. of the Design Automation Conference (DAC)*, June 2001, pp. 667–672.

[5] S. Kumar, A. Jantsch, J. P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani, "A network on chip architecture and design methodology," in *Proc. of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2002, pp. 105–112.

[6] L. Benini and G. D. Micheli, "Networks on chips: a new SoC paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70–78, Jan. 2002.

[7] N. Agarwal, T. Krishna, L. S. Peh, and N. K. Jha, "Garnet: A detailed on-chip network model inside a full-system simulator," in *Proc. of the IEEE International Symposium on Performance Analysis of Systems and Software*, Apr. 2009, pp. 33–42.

[8] V. Puente, J. A. Gregorio, and R. Beivide, "Sicosys: an integrated framework for studying interconnection network performance in multiprocessor systems," in *Proc. of the Euromicro Workshop on Parallel, Distributed and Network-based Processing*, 2002, pp. 15–22.

[9] A. Mello, N. Calazans, and F. Moraes. (2011) ATLAS-an environment for NoC generation and evaluation. [Online]. Available: http://www.date-conference.com/files/file/date11/ubooth/125.pdf

[10] N. Jiang, J. Balfour, D. U. Becker, B. Towles, W. J. Dally, G. Michelogiannakis, and J. Kim, "A detailed and flexible cycle-accurate network-on-chip simulator," in *Proc. of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Apr. 2013, pp. 86–96.

[11] M. Lis *et al.*, "DARSIM: A parallel cycle-level NoC simulator," in *Proc. of the Annual Workshop on Modeling, Benchmarking and Simulation*, Nov. 2010.

[12] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Noxim: An open, extensible and cycle-accurate network on chip simulator," in *Proc. of IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, Jul. 2015, pp. 162–163.

[13] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, p. 1–7, Aug. 2011. [Online]. Available: https://doi.org/10.1145/2024716.2024718

[14] gem5 Simulator website. [Online]. Available: https://www.gem5.org

[15] Sniper Multi-Core Simulator website. [Online]. Available: https://snipersim.org

[16] A. Ben Achballah and S. Ben Saoud, "A survey of network-on-chip tools," *International Journal of Advanced Computer Science and Applications(IJACSA)*, vol. 4, no. 9, Oct. 2013.

[17] M. S. Alalaki and M. O. Agyeman, "A study of recent contribution on simulation tools for network-on-chip," *International Journal of Computer and Information Engineering*, vol. 11, no. 4, 2017.

[18] S. Khan, S. Anjum, U. A. Gulzari, and F. S. Torres, "Comparative analysis of network-on-chip simulation tools," *IET Computers and Digital Techniques*, vol. 12, pp. 30–38(8), Jan. 2018.

[19] K. Gaffour, B. Kamel, A. E. H. Benyamina, P. Boulet, T. Djeradi, and A. K. Singh, "Survey of network-on-chip simulators," in *Proc. of the 1st IEEE International Conference on Embedded and Distributed Systems (EDiS)*, Oran, Algeria, Dec. 2017.

[20] J. P. Diguet, M. Strum, N. L. Griguer, L. Caetano, and M.-J. Sepúlveda, "Scalable NoC-based architecture of neural coding for new efficient associative memories," in *Proc. of IEEE International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Sept. 2013, pp. 1–9.

[21] K. Martin, M. Rizk, M.-J. Sepúlveda, and J.-P. Diguet, "Notifying memories: a case-study on data-flow applications with NoC interfaces implementation," in *Proc. of the Design Automation Conference (DAC)*, June 2016.

[22] M. Rizk, J.-P. Diguet, N. Onizawa, A. Baghdadi, M.-J. Sepulveda, Y. Akgul, V. Gripon, and T. Hanyu, "NoC-MRAM architecture for memory-based computing: database-search case study," in *Proc. of the IEEE International New Circuits and Systems NEWCAS Conference (NEWCAS)*, Strasbourg, France, June 2017.

[23] J.-P. Diguet, N. Onizawa, M. Rizk, J. Sepulveda, A. Baghdadi, and T. Hanyu, "Networked power-gated MRAMs for memory-based computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 12, pp. 2696–2708, 2018.

[24] E. Cota, A. de Morais, S. Alexandre, and M. Lubaszewski, *Reliability, availability and service ability of Networks-on-Chip*. Springer, 2012.

[25] G. D. Micheli *et al.*, "Networks on Chips: from research to products," in *Proc. of the Design Automation Conference*, Nov. 2010, pp. 300–305.

[26] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Proc. of the IEEE Design, Automation and Test in Europe (DATE)*, 2000, pp. 250–256.

[27] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 1025–1040, Aug. 2005.

[28] A. V. de Mello, L. C. Ost, F. G. Moraes, and N. L. V. Calazans, "Evaluation of routing algorithms on mesh based NoCs," Faculty of Information Technology UCRS, Brazil, Tech. Rep., 2005.

[29] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computing Surveys*, vol. 38, no. 1, June 2006.

[30] L.-S. Peh and W. J. Dally, "A delay model for router microarchitectures," *IEEE Micro*, vol. 21, no. 1, pp. 26–34, Jan. 2001.

[31] N. Banerjee, P. Vellanki, and K. S. Chatha, "A power and performance model for network-on-chip architectures," in *Proc. of the Design, Automation and Test in Europe Conference and Exhibition*, 2004.

[32] T. Grotker, S. Liao, G. Martin, and S. Swan, *System Design with SystemC*. Springer, 2002.

[33] ORCC. The open rvc-cal compiler : A development framework for dataflow programs. [Online]. Available: http://orcc.sourceforge.net

[34] Xiph. Xiph.org video test media. [Online]. Available: http://media.xiph.org/video/derf/

[35] "UCI Machine Learning Repository," https://archive.ics.uci.edu/ml/.