World Academy of Science, Engineering and Technology
International Journal of Electrical and Computer Engineering
Vol:1, No:7, 2007

A Generic and Extensible Spidergon NoC

Abdelkrim Zitouni, Mounir Zid, Sami Badrouchi, and Rached Tourki

***Abstract*—**The Globally Asynchronous Locally Synchronous Network on Chip (GALS NoC) is the most efficient solution that provides low latency transfers and power efficient System on Chip (SoC) interconnect. This study presents a GALS and generic NoC architecture based on a configurable router. This router integrates a sophisticated dynamic arbiter, the wormhole routing technique and can be configured in a manner that allows it to be used in many possible NoC topologies such as Mesh 2-D, Tree and Polygon architectures. This makes it possible to improve the quality of service (QoS) required by the proposed NoC. A comparative performances study of the proposed NoC architecture, Tore architecture and of the most used Mesh 2D architecture is performed. This study shows that Spidergon architecture is characterised by the lower latency and the later saturation. It is also shown that no matter what the number of used links is raised; the Links×Diameter product permitted by the Spidergon architecture remains always the lower. The only limitation of this architecture comes from it's over cost in term of silicon area.

***Keywords*—**Dynamic arbiter, Generic router, Spidergon NoC, SoC.

I. Introduction

SINCE, data synchronization problems arise in multi-clock domain SoC and operating clocked interconnects becomes increasingly more difficult, large NoC are treated as Globally Asynchronous Locally Synchronous (GALS) systems, calling for suitable interconnects beyond conventional synchronous buses. GALS paradigm not only avoids the problem of clock skew but also leads to lower power consumption.

Communication between system modules is done by handshaking, where signals are exchanged on the control path in order to arrange the exchange of data. The packet router may use a bundled data that are controlled and transmitted asynchronously [1]. The shared bus solutions used for connecting the functional units by the commercial SoC [2-5] are not a suitable NoC interconnect since it can provide limited connectivity. The NoC are a better alternative than the classic architectures based on busses. Every node is the point of passage of several plots coming of different claimants.

Several works has been focused on the synchronous NoC by using the 2-D mesh, tore, fat tree and hierarchical topologies [6-11]. The synchronous network suggested by Dally [12, 13] is based on a Tore 2D topology and uses the virtual channels technique to improve the necessary quality of services. The cut-trough and packet switching techniques have been used for the interconnect design. Other synchronous routers are discussed in [14]. A synchronous 5-port router provides for two service levels (best effort and guaranteed throughput) has been presented in [15]. The networks STNoC [16] and GeNoC [17] respectively developed by the STMicroelectronics Company and the TIMA Laboratory make the possibility of meeting the increasing requirements of the designs of current and future SoC. These NoC are based on flexible and evolutionary packets, designed according to a layers based methodology. These networks are respectively based on the Spidergon and the Octagon topologies and whose conceptual simplicity results in the best costs of silicon implementation of the routers and the networks interfaces. The notable advantage of STNoC compared to GeNoC is that it integrates adapters of interface making it possible to convert any protocol IP, OCP [18] or STBus out of communication packets.

Only a few works have been focused on the design of asynchronous NoC. Synchronous routers using round-robin arbitration and supporting asynchronous interconnect are presented in [19, 20], though synchronization issues are ignored. The networks Proteo [21] and Chain [22] are two networks which support the Globally Asynchronous Locally Synchronous (GALS) formalisms and thus allow the power minimization. The major disadvantage of these architectures is that they are not generics since they integrate well defined and not flexible protocols and topologies.

Separately the networks GeNoC, STNoC and SoCIN, the other networks presented are based on fixed data structures and do not adopt the generic concept. Thus, only these networks can be adapted with the change of the applicative aspect whose sizes of the handled data as well as the number of the input/output ports can beings variable. In order to develop networks whose dimensions of architectures are adaptable, by fixing the number and the sizes of the communication links according to the traffic of the application, a new architecture called Spidergon is proposed. This architecture use the 4-phase protocol for the communication between the routers and it can be interconnected to a series of architectures going from the tree structure to the simple ring. The generic router used by this architecture integrates a sophisticated dynamic arbiter, a Wormhole routing technique, the Aloha retransmission technique and allows the error cheeking according to the CRC technique. The sizes and the depths of the FIFO contained in this router, the number of input/output ports, the number and the time of retransmission and the maximum numbers of the requests sent to the arbiter are also generic. All these characteristics make the proposed NoC flexible and extensible according to the applicative aspect and thus improve the quality of service required by the application to be mapped on it.

Section 2 presents the Spidergon architecture and its corresponding generic router. Section 3 presents some experiments that study the performances of the proposed architecture compared to other NoC with similar architectures. Finally, section 4 concludes the paper.

Authors are with Electronics and Micro-Electronics Laboratory (Lab-IT06), Faculty of Sciences, Monastir, 5019, Tunisia.

II. Spidergon NoC Architecture

The proposed Spidergon NoC architecture is constructed based on an elementary polygon network which is a combination of the star and the ring architectures (Fig. 1). This elementary network is formed by 4R+1 (R = 1, 2, etc.) routers including a central router that is connected with the 4R peripherals routers via point to point links. The peripherals routers are connected to each other in the form of a ring. The elementary network is characterised by its valence (m = 4R) that represents the number of the peripherals routers. These routers necessitate 2m links to be connected to the central router. Each peripheral router is connected to 4 input/output ports and the central router is connected to m+1 input/output ports.

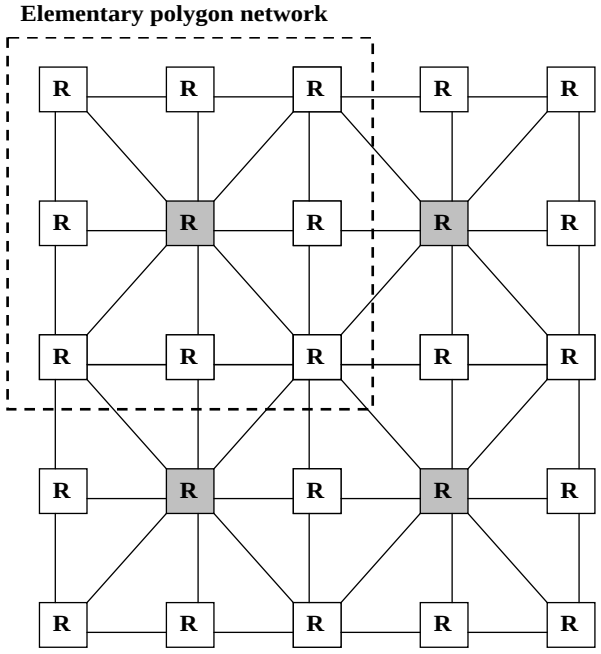

Fig. 1 Example of a Spidergon architecture of valence m = 8

The Spidergon architecture is constructed from a set of elementary polygon networks (Fig. 1) that are organized as a two order matrixes. Such architecture is characterized by the valence m of their elementary networks. A Spidergon architecture with a valence m is constituted by 3m+1 routers that necessitate 7m point to point links to be connected with each other.

This NoC contains 4 types of routers that have 9, 6, 5 and 4 input/output ports. This diversity of router kinds necessitate a generic router in term of number of input/output ports as well as other parameters like channel width, depth of the FIFO, etc.

A. Generic Router

The basic module of the proposed NoC is an asynchronous router with N generic input/output ports, having each a bi-directional exchange bus suitable for the proposed generic NoC architecture. All inter-module communications are carried out in messages witch are divided in packets. These packets are partitioned into small flits, which are sent through the NoC using the wormhole routing technique. The router implements many communication protocols such as flow control, Aloha retransmission, dynamic arbitration, CRC checking, etc.

Each node of the network is identified by a single number which is used as source or destination addresses in the packet heading according to whether the router is transmitter or receiver. This architecture is constituted by 4 modules (port management unit (PMU), dynamic arbiter, routing table and the switch) that communicate with each other by using an asynchronous 4-phases protocol.

1. Packet fields

To support varying communication requirements, three types of flits are considered in the proposed router: a header flit, body flit and a tail flit, indicating End-of-Packet (EOP). Each packet header (Fig. 2) carries information such as the nature of the flit Nat (control or data), related to data communication requirements.

Nat	QoS_id	Destination	Source	P	Nbre	CRC
2 bits	4 bits	6 bits	6 bits	2 bits	4 bits	8 bits

Fig. 2 The packet header fields

The header contains also the quality identifier (QoS_id) of the service to be assured by the router, the address of the target router, the address of the source router, the priority of the packet, the number of the flits, and the CRC code. The priority field is composed of two bits. The “11” code is associated to the signaling packets such as urgent messages, short packets, interrupt and control signals that require low transport latency and represents the highest priority packets. The “10” code is associated to the real-time application packets. The “01” code is associated to the RD/RW packets such as short memory and register access. The “00” code is associated to the block transfer packets such as long messages and blocks of data that represents the lowest priority packets.

A data flit is composed of 4 fields (Fig. 3). The data field contains the data to be transmitted. The Nbre field indicates the flit ordering number. The CRC field indicates the CRC checking code.

The EOP flit is a particular data flit which carries the last data of the packet. The flits format presented in Fig. 3 is dimensioned in the case of 32 bits but in general it is generic.

Nat	Data	Nbre	CRC
2 bits	18 bits	4 bits	8 bits

Fig. 3 Data flit fields

2. Port Management Unit (PMU)

The port management unit (PMU) implements the routing mechanisms and the services which the router must offer to the communication. It integrates a routing function that calculates according to the information transported by the packet header, the address of the output port to which the flit will be transmitted. It also manages the data traffics through the port to which it is associated and performs the communication with the PMU units of the neighbour routers. The PMU units of each router operate concurrently witch minimizes the routing latency of the NoC by allowing the flits to arrive simultaneously by any input port. As

illustrated in Fig. 4 in the case of a router with 4 input/output ports each PMU unit is composed by four modules: a flow control unit, a clock generator, a memory unit (FIFO), and a routing and services unit.

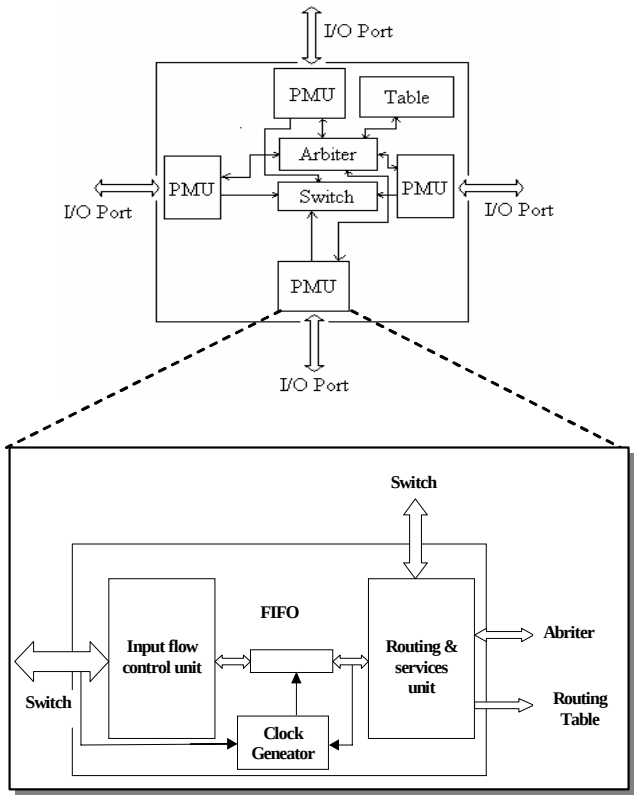

Fig. 4 The router and the PMU internal architectures

The clock generator is intended to generate a stoppable clock signal to rhythm the PMU modules. When a given data transaction is initiated, the clock signal is activated. This signal is stopped only when all the flits contained in the FIFO are evacuated towards the routing and services unit.

The flow control unit, implements the flow control mechanism that ensure that the received flits are well ordered and they are not duplicated. Such a mechanism allows to a router receiving a flit to indicate to its neighbour router, sender of this flit, if it has or not a sufficient memory to store this flit. If the receiver has a sufficient memory it stores the transmitted flit and sends an acknowledge signal to the sender. The detection of this signal by the sender allows it to liberate the buffer or to reload it to send a new flit if it has others about it. If a given deterioration occurs on the level of a flit or that the input FIFO of the receiver are full, this flit is rejected by the router and no acknowledge signal is sent to the sender.

The flits integrity checking mechanism is based on flits counting and the CRC calculation. The flit counting mechanism uses the Nbre field of the header of the packet to be transmitted and allows the identification of EOP. The CRC checking uses an 8 degree polynomial code that is added as a suffix in each flit. The receiving router that has recomputed the CRC compares it with the suffix of the received flit. If the value of the recomputed CRC code is different from the transmitted one, the receiving router wait until this flit is transmitted again based on the Aloha retransmission technique ore reject it when the number of possible transmission is reached.

The flit counting mechanism, the Aloha retransmission technique, the dynamic arbitration and the CRC cheeking are actually the unique QoS services that are implemented in the proposed router. Other QoS can be added and they are coded in the QoS_id field of the header. For example the CRC code is “0001”, etc.

The role of the generic FIFO module is to store the transmitted packets. All the flits coming from the flow control unit are pipelined through the FIFO to the routing and service unit when they are transmitted.

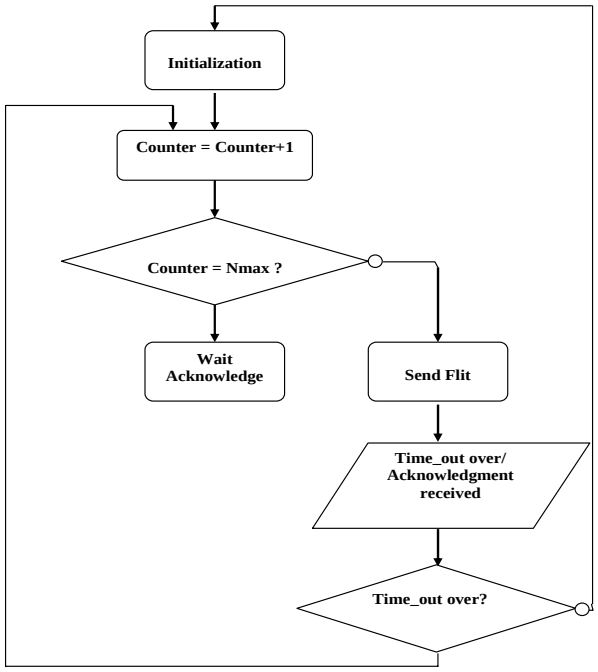

Fig. 5 Flow chart of the retransmission Aloha protocol

After receiving a given flit and if this flit is a header, the routing and service unit takes the necessary information to determine the direction towards the data will be sanded. Then it consults the arbitration unit which manages the access to the selected port. If the routing and service unit receives an acknowledge signal from the arbiter module it sends the header in the corresponding direction and all the flits will be sanded in the same direction. The requester that is granted by the arbiter switches the heading towards the given output port. This port remains reserved until all packet flits are carried out. If during the routing step, this output port is blocked, the router stores the received flits on the corresponding input port in order to send them later when the output port becomes free. Each requester not having access to the output port and whose request was rejected by the arbiter increments its internal counter and wait for a Td time. Once this time is over, the requester starts again the same request with the arbiter. When the counter reaches a maximum value Nmax, the routing and service unit determines other output port, and a new request cycle will begin. The Td time and the Nmax value are generics.

The aim of the Aloha retransmission mechanism is to resolve the problem of loss or the reception delaying of the acknowledgment signals transmitted by the router. This

problem is fatal for the NoC since the transmitter can stop the transmitting, for lack of emission credit. The conflict which results from it can retro propagated until blocking the NoC completely. The advantage of the Aloha retransmission protocol lies in its simplicity. The flow chart of this protocol is given by Fig. 5 where Nmax represents the maximum number of retransmission and time-out indicates the maximum latency of the acknowledgement signals.

3. Dynamic arbiter

In order to resolve access conflict to the output ports, a dynamic arbiter that allows the resolution of access conflict problems of each output port starting from the priority information of each incoming packet has been designed.

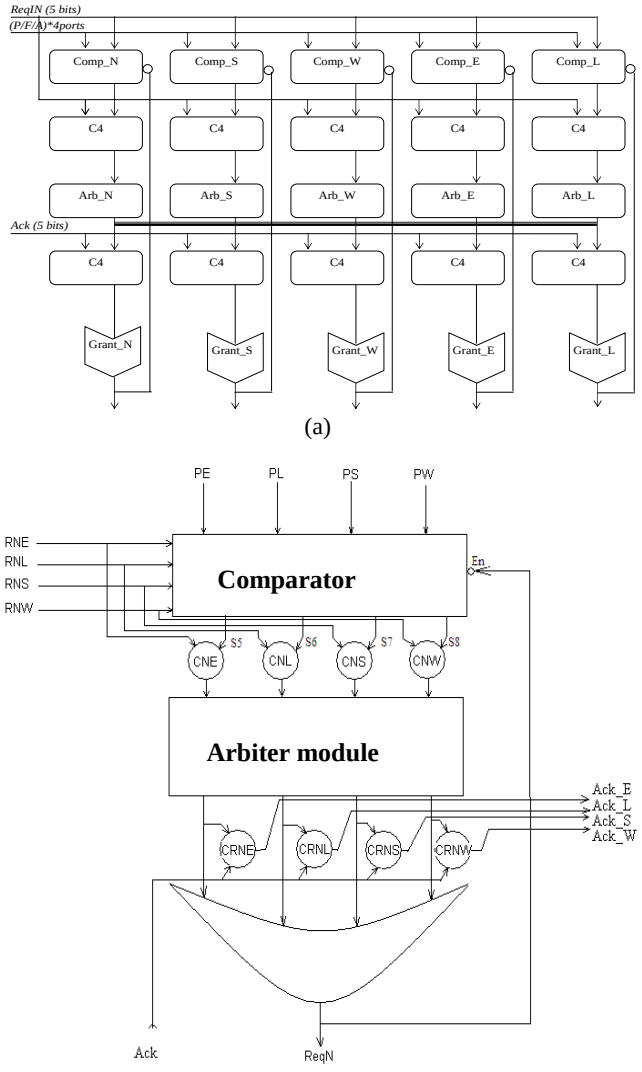

Fig. 6 Architecture of the dynamic arbiter

This dynamic arbiter is constituted by a speed independent round-robin arbiter and a comparator module. Thus, a router with N input/output ports integrates N dynamic arbiters with N round-robin arbitration modules and N comparators as presented by Fig. 6.a in the case of a 2D mesh NoC arbiter that contains 5 input/output ports (Local, East, West, North and South). Each dynamic arbiter serves the 4 demands that are addressed to every port. The requests effectively allocated by the different comparators are treated by the corresponding arbiter modules.

Each dynamic arbiter is constituted by m comparators and m round robin arbitration modules interconnected by m C-elements (m represents the number of requesters) and an OR gate as shown by Fig. 6.b in the case of an arbiter with 4 requesters. After receiving the request signals from the requesters, the priority comparator stores the priority values from the priority busses of the active requests and compares them. After the comparison step, the comparator sends out a set of internal signals that correspond to the requesters that have the highest priorities.

After being combined with 4 C-elements, the output signals are transmitted to the arbiter module. Thus, only the requesters that have requested the access and that have been selected by the comparator module will be treated by the arbiter. If there is only one requester that has been selected by the comparator it will be granted the access automatically by the arbiter and the priority list is shifted in a circular way. If there is two or three requesters that have the same priorities orders, then the requester that has the last highest priority order will gained the access and the priority list is shifted in a circular way. The outputs of the C-elements that enter to the arbiter module will be activated only if the requesters have requested the access and they are selected by the comparator. If a requester has gained the access the comparator module will deactivate the generated internal signals. This scheme is implemented by an enable (En) signal that is generated by an OR gate with 4 inputs (Grant signals). The outputs of the C-elements that enter to the arbiter still high until the active requests become low. Also the comparator will be activated again only if the enable signal becomes active (the actual requester has released the bus).

4. Routing table and switch

After the arbitration phase, the arbiter sends a request to a table which checks the state of the concerned output ports and acknowledges the arbiter if one of these ports is free. The state of each output port (free or occupied) is memorized in a register.

The switch allows commutating the flits coming from the PMU units towards the wearing of selected destinations. Each PMU communicates with the switch thanks to two signals UX and ADRX. UX is the port with 32 bits which conveys the flits coming from the PMU unit X and ADRX is the address coded on 3 bits chosen by the PMU unit. Address ADRX is communicated by the PMU unit X with the switch to choose the address of the output port through which the flit must be transferred. This address also makes it possible to the switch to acknowledge the external core towards which the flit is intended.

B. Generic Parameters of the Spidergon NoC

The proposed Spidergon NoC is a flexible, easily extensible network and offers a variety of services to the communication owning to the fact that it is completely generic. It makes the possibility of choosing and modifying parameters such as the width of interconnection, the depth of the FIFO, the sizes of the fields of the flits, the maximum number of retransmission of a flit and a request of the arbiter

and the propagation times of the signals which prove very important for the correct operation of the total system. Moreover, it is generic in terms of supported number of cores. The development of this network is based on a library of generic models of VHDL blocks. The files of this library contain protocol (number of retransmissions, allowed requests, time out, degrees of adaptability and size of each field forming the various types of flits) and physic (width and depth of the FIFO, number of input/output of the routers and the valence m of the network) parameters.

These files also contain all the functions used by the VHDL blocks like the path calculation function, the CRC checking function, etc. The generation of the Spidergon architecture is done automatically by indicating the valence m in the package file by using the VHDL GENERATE clause. The portion of the VHDL code of Fig. 7 shows how to generate the peripheral routers in an elementary Polygon network of valence m.

```
Generate: For i in 1 to m generate
Perif_Router: Router generic map(width,i)
        port map(R0=>Request_in(i),
        Data_in0 =>   Input((i+1)*width-1 downto i*width),
        --
          …);
End generate;
```

Fig. 7 Clause GENERATE for generating peripheral routers of the elementary polygon network

C. Routing Function

The Spidergon network has a strategy of distributed routing. Each router is independent to the others routers and makes the decisions of switching and memorizing of the flits in each input port without the intervention of a central synchronization. Switching is carried out thanks to an adaptive routing function which uses the heading data of the packet, in particular the destination address, to calculate the output port of the flit. The routing function depends on the position of the router in the network as of the number of the input/output ports that it contains. The central router intervenes if there is congestion.

For routing a packet received on an input port i of a given router, the routing unit of this port decodes the packet heading and extracts the address of the destination module to be reached by the packet. Then it calculates the number of the output port to be used by the flits by using the router and the target core addresses as parameters.

III. Experiments

The performances of the proposed NoC are studied and compared with two other NoC with similar architectures (Mesh and Tore).

A parameterised network model was constructed using HASE (Hierarchical Architectural Simulation Environment) [23]. The underlying simulation system is multi-threaded and even-driven. Each tile or node generates packets with random destinations. Packets are generated at a constant rate and queued until they are able to enter the network. The interval between the creation of individual packets is random (geometric distribution) to prevent packets being injected into the network synchronously. Network latency is measured from the time the first flit is created to the time the last flit in the packet is received at its destination, including any time spent buffered at the source node. Each node injects 1000 packets into the network and performance statistics are gathered after an initial warm-up period of 100 packets/node. Packets are 64 flits in length.

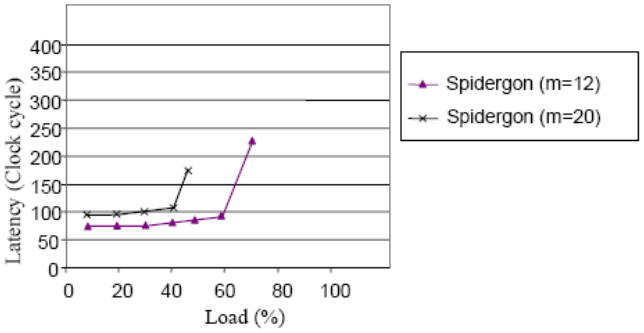

Fig. 8 Latency versus load for two Spidergon architectures

Fig. 8 shows the evolution of average latency according to the load for two sizes of the Spidergon architecture. The Spidergon architecture of valence m=12 contains 37 routers and the Spiergon architecture of valence m=20 contains 61 routers. It can be seen that the latency increases with the size of the network. Indeed, for weak loads the average way borrowed by the packets increases with the number of router. Moreover, the network saturates more quickly. A Spidergon network of valence m=20 saturates starting from a load equivalent to 40% whereas a Spidergon network of valence m=12 saturates with a load equivalent to 60%. Indeed, a larger network emits more packets. It proposes also more buffers for stoking these packets in the event of conflicts. But in an important network the ways are requested by more packets simultaneously thus a conflict affects more packets and saturation intervenes earlier.

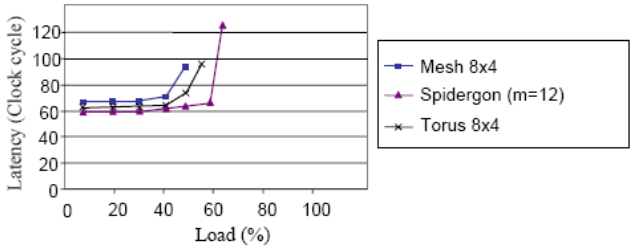

Fig. 9 Latency versus load for three architectures (Spidergon, Mesh and Tore)

The Fig. 9 shows the evolution of average latency according to the load for architecture Spidergon of valence m=12 (37 routers) and two other similar architectures Mesh 2D and Tore with 32 routers (8x4). The Spidergon architecture is characterized by a lower latency than the two others architectures. This difference is increasingly large after saturation. Also the network Spidergon saturates later than the two others architectures. Indeed, the packets cross less routers number in the Spidergon network than in the network Mesh 2D and Tore. In fact it is noticed that no matter what the number of the used links is high the Links×Diameter (L×D) product of the Spidergon NoC remains always lower than that of the Tore and the Mesh 2D networks (Fig. 10).

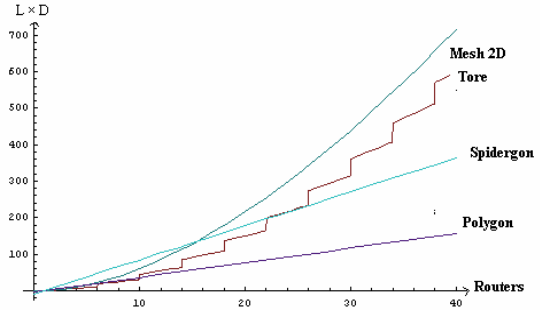

Fig. 10 Comparative study of the proposed NoC with Mesh 2D and Tore architectures in terms of Links×Diameter parameter

Fig. 11 shows the evolution on the area of the networks Mesh 2D, Tore and Spidergon according to the number of router in technology CMOS 0.35 µm for buffers of 6 words. The more important the network is, the more the differences between areas of the three networks are large. This is due to the central routers of the Spidergon network which have (m+1) buffers, whereas in a Tore architecture all the routers have 5 buffers and in the Mesh network the peripheral routers have only 3 or 4 buffers.

These results show that the proposed Spidergon architecture is powerful in term of latency comparatively to the other two architectures. Its limitation comes from it's over cost on the area particularly for an important network.

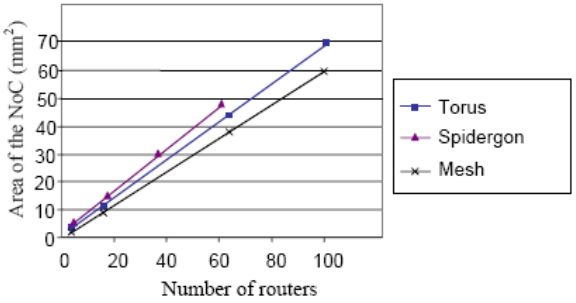

Fig. 11 Area of the NoC versus number of routers in the case of the Tore, Mesh and Spidergon architectures

IV. Conclusion

This paper presents a flexible and easily extensible asynchronous NoC architecture that offer a variety of SoC communication services owing to the fact that it is completely generic. The router that is integrated in this architecture and that is based on a dynamic routing function allows to modify and to choose starting from a parameterized library different protocol and physic parameters. The protocol parameters consists on the Aloha retransmission time-out, the requests and their retransmission numbers, the adaptability degrees, the size of each field forming the various types of flits, etc. The physical parameters consist on the width and the depth of the FIFO, the number of the input/output ports, the valence m of the NoC, etc.

Compared with other NoC with similar architectures, the value added by the Spidergon architecture, resides in its capacity to handle a suitable cost/performance compromise in the field of NoC. This thanks to its wide generic character and its constant and low diameter. It was shown that the Spidergon architecture is characterised by the lower latency and the later saturation. It is also shown that no matter what the number of used links is raised; the Links×Diameter product permitted by the Spidergon architecture remains always the lower. The only limitation of this architecture comes from it's over cost in term of silicon area.

In order to allow the use of the Spidergon architecture at various levels of abstraction we are under modelling it in SystemC language at TLM (Transaction Level Modelling) that is suggested by OCP-IP (Open Protocol-International Core Partnership). Moreover we are also under modelling powerful adapters of protocols and/or levels which allow a fast and accurate communication. Indeed routing of the data with the IP which treat them remainder in the systems on chip a source of congestion.

References

[1] I. Sutherland, "Micropipelines," *Comm. of ACM*, vol. 6, 1989.
[2] Sonics, Incorporated, http://www.sonicsinc.com.
[3] W. Peterson, "Design philosophy of the wishbone SoC architecture," 1999. Available: http://www.silicore.net/wishbone.htm
[4] D. Flynn, "Amba: enabling reusable on-chip design," *Intl. J. IEEE Micro*, pp. 20-27, 1997.
[5] IBMCoreConnect Information, 2000. Available: http://www.chips.ibm.com/products/powerpc/cors
[6] J. Liang, S. Swaminathan, and R. Tessier, "A SoC: a scalable, single-chip communications architecture," *IEEE Intl. Conf. Parallel Architectures and Compilation Techniques*, pp. 524-529, 2000.
[7] H. Ho, and T.M. Pinkston, "A methodology for designing efficient on-chip interconnects on well-behaved communication patterns," *The 9th Intl. Symposium on High-Performance Computer Architecture (HPCA'03)*, pp. 377, 2003.
[8] S. Kumar, A. Jantsch, J. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyria, and A. Hemani, "A network on chip architecture and design methodology," *Proc. IEEE Computer Society Annual Symposium on VLSI*, pp. 105-112, 2002.
[9] J. Hu, and R. Marculescu, "Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures," *Proc. Design, Automation and Test in Europe Conference*, 2003.
[10] T.T. Ye, L. Benini, and G.D. MICHELI, "Packetized on-chip interconnect communication analysis for MPSoC," *Proc. Design Automation and Test in Europe*, pp. 344-349, 2003.
[11] W.J. Dally, and B. Towles, "Route packets, not wires: on-chip interconnection networks," *Proc. the 38th Design Automation Conference*, 2001.
[12] L. Peh, and W.J. Dally, "A delay model and speculative architecture for pipelined routers," *7th Intl. Symp. High-Performance Computer Architecture (HPCA)*, 2001.
[13] R. Mullins, A. West, and S. Moore, "Low-latency virtual channel routers for on-chip network," *Proc. 31st Intl. Symp. Computer Architecture*, 2004.
[14] H. Wang, L.S. Peh and S. Malik, "Power driven design of router microarchitectures in on-chip networks," *Proc. MICRO-36*, 2003.
[15] E. Rijpkema, K. Goossens *et al*., "Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip," *IEE Proc.-Comp. Digit. Tech*., pp. 294-302, 2003.
[16] L. Benini, G.D. Micheli, "Networks on chips: a new SoC paradigm," *IEEE Computer*, vol. 1, pp. 70-80, 2002.
[17] J. Schmaltz, D. Borrione, "A generic network on chip model," *Research reports*, TIMA Laboratory, Grenoble, France, 2005.
[18] OCP International Partnership. Open Core Protocol Specification. 2.0 Release Candidate, 2003.
[19] N. Banerjee, P. Vellanki and K.S. Chatha, "A power and performance model for network-on-chip architectures," *Proc. DATE*, 2004.
[20] P. Vellanki, N. Banerjee and K.S. Chatha, "Quality-of-service and error control techniques for network-on-chip architectures," *Proc. GLSVLSI'04*, Boston, USA, pp. 45-50, 2004.
[21] W.J. Bainbridge, S.B. Fuber, "Chain: a delay insensitive chip area interconnect," *IEEE Micro*, vol. 22, pp. 16-23, Sep./Oct. 2002.
[22] Bainbridge, S.B. Fuber, "Asynchronous macrocell interconnect using marble," *International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 122-139, IEEE Press, Avril 1998.
[23] P. Coe, F. Howell, R. Ibbett, and L. Willams, "A hierarchical computer architecture design and simulation environment," *ACM Transactions on Modelling and Computer Simulation*, 8(4), October 1998.