μSpider: A CAD Tool for Efficient NoC Design

Samuel Evain, Jean-Philippe Diguet, Dominique Houzet

µSpider: a CAD Tool for Efficient NoC Design

Samuel Evain[1&2], Jean-Philippe Diguet[1] and Dominique Houzet[2]

[1] LESTER, FRE 2734 CNRS, UBS Univ.
Rue Saint-Maudé, 56325 Lorient Cedex - France
E-mail: evain@univ-ubs.fr

[2] IETR–INSA, UMR 6164 CNRS, Rennes 1 Univ.,
20, av. Buttes de Coësmes,
35043 Rennes Cedex - France

Abstract:

In this paper, we present a generic router and a tool that allow the designer to easily and quickly customise a NoC in order to meet the requirements of a set of applications. Our router addresses what we consider as the main features of a realistic and useful NoC. Firstly, it supports the management of different levels quality of service (QoS) allowing guaranteed throughput service in addition to the classical best effort service. Secondly, it is associated to a CAD tool, which can fully configure and generate the NoC VHDL code at the RTL level. The paper presents the router architecture and its various custom characteristics as well as their impacts on the performance of the system.

1. Introduction

Future Systems-on-Chip (SoC) for multimedia-telecom will contain a great amount of processing elements as GPP, DSP and dedicated HW, connected themselves and with other elements like RAM and peripheral I/O. Traditional approaches for communication are the use of bus or multi-bus architectures, however they don't meet the future requirements mainly depending on both performance and economic aspects [1]. The leading features are scalability, flexibility, reusability, and reprogramming in particular for online debugging.

Regarding the amount of available transistors on upcoming (reconfigurable) chips, the NoC will naturally become a viable solution. However, CAD tools are needed firstly to tune NoC parameters before synthesis with regards to the target set of applications and secondly to provide the synthesis tool with a reliable HDL specification. In this paper we address the second issue. Applications which would require NoC facilities also need some real time (embedded multimedia, software radio[2]) and reconfiguration capabilities (online debugging, upgrades). Therefore, NoC, which is the heart of the system, must assume guaranteed throughput (GT) besides traditional Best Effort (BE) services.

The rest of the paper is organised as follows, in the next section we present what are the real important points regarding a NoC specification. In section 3 we detail the different configuration parameters offered by our framework and the way they can be used to meet the application requirements. In section 4 we present our CAD tool. In section 5 we present how our solution distinguishes from existing work. In section 6 we comment synthesis results.

2. NoC generalities

2.1 NoC Basic concepts

NoC is based on two basic elements: the routers and the network interfaces (NI). In a wormhole packet switching network, messages are divided into packets. Routers switch channels to carry packets from their source to their destination in the network. NIs connect processing elements to the NoC. Figure 1 shows a simple NoC example with four routers.

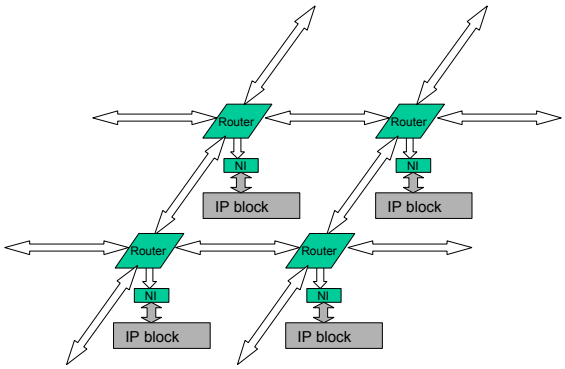

Figure 1. NoC example with connected elements.

2.2 The traditional approach

Packet switching network started with parallel computing where real time and GT aspects were almost never taken into account. For this reason, the most usual benchmark referred is the "uniform random distribution of packet destinations". By simulation the average latency of packets in function of the traffic load in the network is observed to consider the performance of a network. Consequently, it doesn't consider the communication specificity of applications that may lead to hot spot in the network. However, in practice GT or QoS levels are necessary for the class of applications that justify the usefulness of NoCs. Thus, the capability to deal with GT is not a nice option but a real necessity to assume minimum QoS in addition to the BE traffics.

Another real constraint that is usually underestimated is the question of clock distribution. In most NoC solutions, the clock is considered unique, synchronous without any skews. If we admit that NoCs are useful for large distributed systems on chip with recent or future technologies, it means that the die size doesn't allow a uniform clock distribution over the chip. Different approaches exist to cope with this question; the more realistic one is to implement globally asynchronous communication between locally synchronous digital elements.

3. µSpider generic features

3.1 The generic router architecture.

µSpider is based on a trade-off between guarantied and BE capabilities. It is customisable through an associated CAD tool.

Router ports are composed by unidirectional opposite channels as shown on Figure 2. The number of ports of a router is configurable. By this way network topology is flexible and there isn't unnecessary port created which leads to save area and power consumption.

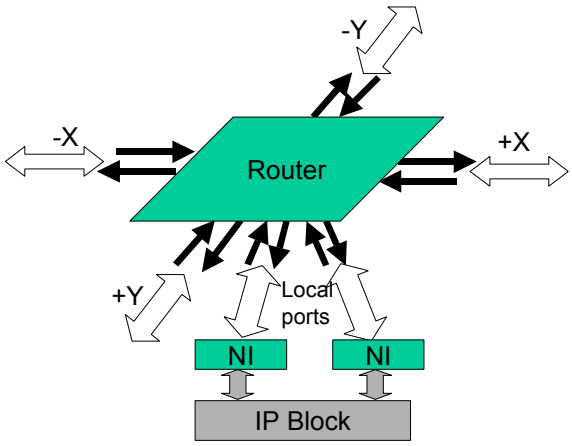

Figure 2. A 2D router with 2 local ports

Topology & routing choices: First, routing techniques can be separated in two main types, deterministic and adaptive. Adaptive routing avoids some contention by changing the path of packet in function of current network conditions. But it implies a reordering of packets at destination interface leading to an increase of memory required by the interface. To avoid deadlocks, it may be necessary to add virtual channels, increasing the router complexity. Moreover, routing decision is more complicated and slower in the case of adaptive routing than in the one of deterministic routing. So we made the choice of a deterministic routing. Secondly, we opt for source routing instead of a distributed one to avoid full rule table distribution and to facilitate possible run-time configuration. Finally, the routing technique can be simplified if the NoC topology is regular. However, it can be useful to have a routing technique that can be independent from the topology. So, a choice between two routing techniques is available. The first technique is called "dimension-ordered routing", it can used only in the case of a regular n dimension mesh topology. In dimension-ordered routing, each packet is routed in one dimension at a time. It must arrive at destination coordinate in each dimension before proceeding to the next dimension. Our framework provides two implementations for this technique. In the first case the routing information carries the destination coordinates, the second one carries the difference between the destination and source coordinates.

The second technique, called "street-sign" allows the NoC to be independent from its topology. But it imposes at source interface some tables containing paths to the destinations. The routing information only carries the addressees of routers where turns must be done and their associated directions. After being proceeded by the router, turn information is removed from the packet header.

Packet header routing information in the both routing cases is shown on Figure 3. In summary, two topologies types are allowed and two routing techniques are adapted for: 1) dimension-ordered routing which is cheaper and need a regular n dimension mesh topology, and 2) Street sign routing which can support any NoC topology.

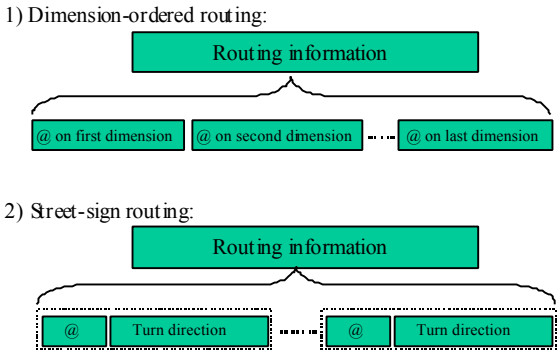

Figure 3. Routing information in packet header with dimension-ordered routing & street-sign routing.

In addition to the standard n dimension mesh topology, our router accepts multiple local ports (Figure 2). Thus, it allows taking advantage of the locality of communications that we can found classically in a SoC around a processor for example.

Virtual channels (VC)**:** The designer can choose the number of virtual channels he wants in the NoC. Virtual channels make possible the separation between different traffics despite the use of a common physical medium [3]. In our case, packet priority corresponds to its VC number. Virtual channels provide the ability to deliver guaranteed communications throughput because they allow a packet having a greater priority than other packets to interrupt and overtake them. Virtual channels are multiplexed on a single physical communication. Each virtual channel possesses its own buffer, decoding, arbitrary and flow control units as show on Figure 4. So they are expensive and must be used carefully. For each virtual channel, it is possible to configure independently from other virtual channels: the buffer depth, the routing technique, the arbitration technique, and the flow control technique.

Buffers: Input queuing allows the router to buffer some flits before it is switched. It also allows storing packets when they can't progress in the NoC and so avoid them to stay in the way of other packets. The buffer depth acts on the reduction of contentions between progressing packets and so can reduce traffic latency.

Arbitration technique: Arbitration allows the router to make a choice when packets coming from several inputs request the same output channel. Three arbitration techniques are offered: "no arbitration", "static channel priority", or "round robin". In each router output port, virtual channel arbiters of this port are connected by a freeze signal to give priority to the first virtual channel on the second virtual channel, and so on. By this way a packet on the first channel can pre-empt the output port and freeze the other virtual channels as long as it uses this output port.

Flow control technique: Flow control manages flit exchanges between neighbouring routers. Three Flow control techniques are available: "no back control", "handshake" and "credit based".

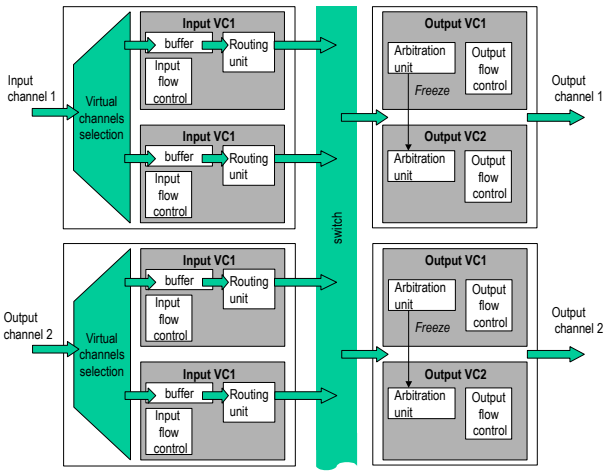

Figure 4. Partial router architecture showing 2 input ports with 2 virtual channels (VC) and 2 output ports

Synchronous or asynchronous exchange between neighbouring routers. Synchronous exchanges are based on the assumption that the clock is the same (synchronous) in all the NoC. This assumption is not realistic in many real complex SoCs, so we have added an asynchronous mode based on asynchronous buffers within input channels.

µSpider examples for different traffic classes: By implementing VC, the designer can make a NoC able to support different levels of guaranteed services.

- If only BE traffic is required, no extra VCs are needed. The arbitration technique may be "round robin" or "channel priority".
- If the designer wants a GT in addition, he must add a VC with a higher level of priority than BE traffic. By this way, GT can pre-empt an output port even if a BE is using it. To re-obtain the output, the BE must wait until GT doesn't need it any more. To avoid contention between GT, they must be pre-scheduled and their paths must be dissociated. In this case arbitration technique can be "no arbitration" and flow control can be "no back control" because of the assumption of impossible contention for GT. The buffer size is minimised in order to only buffer data during the delay due to the input decoding and the output allocation steps in the router.
- Other intermediate traffic classes may be supported between the GT and the BE. In that case some additional priority levels are inserted. The GT class keeps the highest priority level in standard working cases. The intermediate traffic classes may be used to carry message that must have a short latency but have not been pre-scheduled and so don't belong to the GT. Those messages can be control information. By this way they will have priority on the basic BE traffic.

Table 1. Properties in function of traffic classes

Traffic classes	GT	BE 1	BE 2	BE n
Traffic priority	1	2	3	n
VC number	1	2	3	n
Prescheduled	Yes	no	no	no
Arbitration	No	yes	yes	yes
Flow control	No	yes	yes	yes

Table 1 summarises the previous explanations. Arbitration and flow control can be omitted for the traffic with the greatest priority if it is prescheduled.

3.2 Network interfaces

Network interfaces must be flexible and configurable to be adapted with the connected processing elements. They must also implement the network protocol (Figure 5).

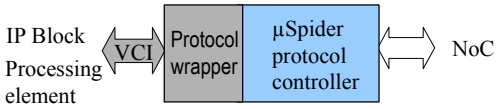

Figure 5. Network interface

For IP standardisation reasons, we made the choice of the VCI interface [4] for the communication between NI and IPs. NI uses a table to transform VCI addresses map in packet header routing information according to the NoC configuration. The interface architecture is strongly related to our soft scheduling technique that requires a time slots controller and buffering capabilities. Because of limited space, this point would be detailed in a future paper.

3.3 Packet format

Packet format is represented in Figure 6. All flits (Flow Control digit) are composed of flit type, virtual channel number and payload fields. Only the header flit carries the routing information. The other flits carry the message.

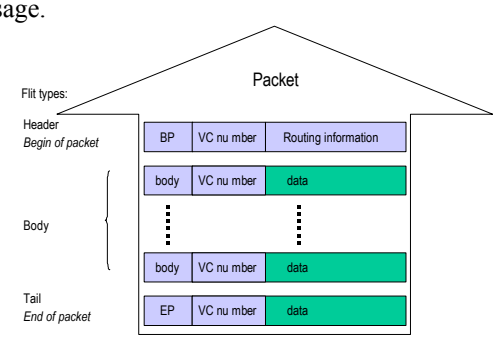

Figure 6. Flits of a packet.

4. IP tool for fast NoC design

This tool allows the designer to easily and quickly customise a NOC with a graphical PHP. Finally, this tool generates an optimised dedicated NoC VHDL code at RTL level, synthesisable on usual FPGA platforms. The different design options are summarised in Figure 7.

- Data word wide in bits.
- Synchronous or asynchronous communication mode between neighbouring routers.
- Topology configuration:
  - Mesh or torus.
  - number of dimensions (1D, 2D, 3D,…).
  - Size on each dimension.
  - Number of local ports by routers (allow to have multiple local ports).
- Number of virtual channels.
- Independent virtual channel configuration choices:
  - Flow control technique.
  - Routing technique.
  - Arbitration technique.
  - Buffer depth.

Figure 7. NoC configuration choices

5. Our contribution and related work

Many packet switching network already exist in both academic and industrial area. To our point of view, the more complete and comparable to our work are: SPIN[5], RASoC [6] and AEthereal [7]. SPIN was a pioneer and so probably the more well developed NoC but integrates few configuration capabilities. RASoC is implemented as a parameterised VHDL code implementing only BE traffics. RASoC can tune a limited number of architectural parameters, while our approach is firstly more complete and secondly consists in a functionality generation (e.g. arbitration policy) that indirectly modifies the NoC architecture and its behaviour. Æthereal is an interesting solution including two traffic classes GT and BE. To our knowledge, our solution proposes more configurations options (e.g. routing technique), an asynchronous communication mode and a CAD framework. The table 2 summarises our contribution.

Table 2. NoC comparison

NoC name	SPIN	RASoC	Æthereal	µSpider router
Topology	multi-level Fat-tree	2-D grid or torus	any	n dimension mesh or any
Router ports (I/O) number	8	5	1 to 6	Generic
Data width (in bits)	32	Generic	32	Generic
A/Synchronous System Clock	Synchronous	Synchronous	Unknown	Both available
Link flow control protocol	Credit based	Handshake	Credit based for BE, no back control for GT	Credit based , Handshake or no back control
Buffering	Input buffer + 2 shared output buffers	Input buffer	Input buffer	Input buffer
Routing algorithm	Adaptive upward routing. Deterministic downward routing.	Deterministic source-based routing algorithm (XY).	Source routing	2 deterministic source-based routing algorithms: dimension ordered or street-sign
Output Arbitration	Round robin (RR)	RR	RR for BE. No arbitration for GT	RR, fixed channel priority or no arbitration
Traffic classes supported	BE	BE	BE + GT	BE + GT + intermediate QoS classes
Virtual channels number	1	1	Equivalent to 2 VC	Generic
Specification Input	No	VHDL code	No	Explicit CAD tool

6. Experimental results

The NoC VHDL generated code has been implemented on a Xilinx XCV400E FPGA (Speed-7) including 4800 slices.

The common configuration of the NoC router is:

- Five ports router.
- Data wide = 8 bits.
- Buffer depth = 3 words.
- Flow control technique = Handshake.
- Routing technique = Dimension-ordered routing.
- 1 VC to have only BE traffic class.

The µSpider tool generates a VHDL RTL code of approximately 2700 lines. We use ISE 6.2 Xilinx synthesis tool and observe the "place and route" report

Table 3 shows that the arbitration technique impacts on the router critical path and the area of the router.

Table 3: Arbitration technique cost comparison

Arbitrage mode	No arbitration	Fixed channel priority	Round-robin priority
Max. router frequency (MHz)	96	93	86
Nb of slices	293	354	380

The arbitration technique is involved with the critical path. It impacts the router maximum frequency and the area cost.

Table 4 figures out how the use of different traffic classes combinations influences the design cost.

Table 4. Traffic class combination cost comparison

	GT No back flow control, No arbitration	PrioBE Handshake flow control, Fixed channel priority	BE Handshake flow control, Round-robin priority	Frequency (MHz)	Number of SLICEs	Number of bits by unidirectional link	Link max bandwidth (Mbits/s)
BE	No	No	yes	86	380 out of 4800 7%	12	523
GT	yes	No	No	96	283 out of 4800 5%	12	782
GT +BE	yes	No	yes	58	794 out of 4800 16%	14	410
GT + PrioBE + BE	yes	yes	yes	50	1153 out of 4,800 24%	16	343

The use of both GT and BE traffics induces a slice cost higher than the sum of the area of two independent routers GT and BE because additional logic is needed to multiplex and manage interaction between this two classes of services. The maximum frequency decreases with the addition of traffic services and so reduces the maximum link data bandwidth. However, combining both techniques enable to share a single set of communication channels with a light penalty regarding additional control bits.

7. Conclusion

The NoC designer is facing a huge and tedious design space. In this paper, we have presented a complete and generic NoC architecture for NoC code generation. The various choices allowed in the router and network interface have been described. The necessity of dealing with different guaranteed traffics has been demonstrated. A NoC configuration tool allowing rapid prototyping has been presented. Finally, results obtained for various traffic classes show that a trade-off must be find between QoS and maximum throughput and consumed chip resources.

However, we currently work on two other necessary steps. The first point is an area, power and delay estimations based on design reuse, namely with previous designs stored in a library. Secondly, a decision tool is underdevelopment in order to select the NoC parameters regarding application requirements in terms of GT for instance. Moreover, a transaction level modelling (TLM) SystemC model will soon available for SoC validation by fine simulation.

References

[1] W.J.Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks", *DAC'01*, pp. 684-689.

[2] J. Mitola, "The software radio architecture", *IEEE Communications Magazine*, pp. 26-38, May 1995.

[3] W. J. Dally, "Virtual-Channel Flow Control", *IEEE Trans. on Parallel & Distributed Syst.*, Vol. 3, no. 2, March 1992.

[4] Virtual Socket Interface Alliance, http://vsi.org Aliance.

[5] P. Guerrier and A. Greiner, "A generic architecture for On-chip Packet-switched Interconnections", *ACM*, pp. 250-256 Series-Proceeding-Article, 2000.

[6] C. A. Zeferino et al. "RASoC: A Router Soft-Core for Networks-on-Chip", *Proc. DATE*, pp.16-20, February 2004.

[7] K. Goossens et al. "Networks on Silicon: Combining Best Effort and Guaranteed Services", *Proc. DATE*, March 2002.