PARAMETRIC VHDL MODELS OF ARBITERS FOR NETWORKS-ON-CHIP

Frederico G. M. E. Santo, Cesar A. Zeferino

Universidade do Vale do Itajaí – Centro de Ciências Tecnológicas da Terra e do Mar
Rua Uruguai, 458 – 88302-202 – Itajaí – SC – Brazil
E-mail: {fremariani, zeferino}@inf.univali.br

ABSTRACT

Networks-on-Chip (NoCs) will meet the major requirements of future Systems-on-Chip (SoC) offering, at the same time, reusability, scalability and parallelism, while coping with other important issues, like power consumption and clock distribution. One of the major subsystems of NoCs is the arbiter used to build their routers. It is responsible to schedule the packets incoming the input channels of the routers in order to forward them to their output channels. In this paper, we present some arbiter implementations, which were modeled in VHDL and synthesized to FPGA. The major advantages of our models relies on the use of parametric structures, which allows to reuse a same model for different arbiter sizes. Results show the costs and operating frequency of three arbiter models for different router sizes.

1. INTRODUCTION

The last submicron technologies has allowed the building of entire systems on a single chip, including, processor(s), embedded memories and I/O controllers [1]. Such systems are named Systems-on-Chip (SoCs) and they are usually based on the reuse of pre-designed and pre-verified components, which are called cores or IP (Intellectual Property) blocks.

In the current SoCs, the cores are usually interconnected by means of both centralized or hierarchic bus architectures. However, as it is known, the bus-based approach presents several limitations is respect to its communication performance and power consumption, and it will not meet the requirements of future SoCs, which will integrate from dozens to hundreds of cores on a single chip [2]. To meet such requirements, a new approach has been proposed, and promises to be the major solution for the problem of interconnect cores in future SoCs. This approach is named Network-on-Chip (NoC), and is based on the concepts used in interconnection networks for parallel computer (multiprocessors and multicomputers).

A NoC can be defined as a set of routers and point-to-point channels interconnecting cores in a SoC. The cores are attached to the routers and these ones are interconnected among them. Cores communicate by exchanging request and response messages, which are forwarded by the routers and channels in the sender-to- receiver path.

The major component of a NoC is its router, which is composed by a crossbar, memory buffers (or FIFOs) and scheduling controllers. Such controllers are responsible to schedule an output channel for a message that incomes an input channel of a router (output scheduling), or to schedule an incoming message for an output channel when there are several ones requiring the use of the same output channel (input scheduling). The output scheduling is performed by routing circuits which analyze the header of the incoming messages, select and output channel for each message, and send a request to the selected output channels. The input scheduling is performed by arbitration circuits (or arbiters) which receive requests from the routing circuits, apply some arbitration algorithm based on a given priority criteria, and grant the selected requests.

The performance of a router is strongly dependent on the performance of its arbitration circuits. As fast the arbiter is, faster the message forwarding is. Furthermore, these circuits must ensure a fair use of the output channels by the input ones, and that none message will stay indefinitely waiting for an opportunity to be forwarded to its destination (problem known as *starvation* [3]).

This paper presents the development of parametric VHDL models of arbiters for routers of NoCs. In the following sections, we present some architecture details of three arbiter architectures and results of synthesis in FPGA.

2. ARBITER ARCHITECTURES

The arbiter of a router can be implemented in a centralized or in a distributed way. In the first one, the arbitration is done by a single module. It evaluates all the requests emitted by the routing circuit(s), runs the scheduling algorithm having a global view of the output channels usage, selects some of the simultaneous requests, and grants the selected ones. In the distributed approach, there is an arbitration module for each output channel, which performs the same actions, but has a local view of the usage of the arbitrated output channel. Nevertheless, its architecture is simpler and allow the building of faster arbiters.

In this paper, we focus on the development of parametric VHDL models of distributed architectures. Each model has an input parameter named *n* which defines the number of requests to be scheduled. Varying the value of *n* for a given instance, one can reuse a same model to synthesize arbiters with different sizes.

In this paper, we considered three architectures, which has the same interface and similar structure (Fig. 1).

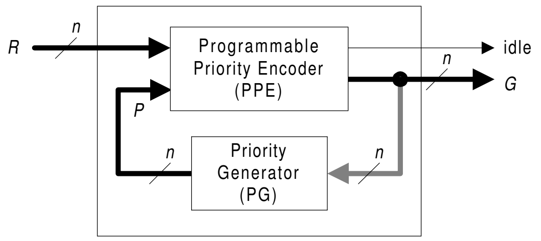

Fig. 1. Arbiter's interface and internal structure.

All the three arbiter architectures use a dynamic priority scheme, but one of them is based on a random approach, while the other two are based on a round-robin (*rr*) algorithm that ensures fairness and starvation freedom. The three arbiters are named: *random*, *rr_ripple* and *rr_comb* [4].

As is shown in Fig. 1, their implementations are based in a programmable priority encoder (PPE), which receives *n* requests (*R*) and selects on them by considering the current *n*-

bit priority code (P) generated by the priority generator (PG). The PPE's outputs are the n grant signals (G), which select only one of the input requests, and the state signal *idle*, which says if the arbitrated resource is free or busy .

2.1. Programmable Priority Encoder

The PPE's behavior is the same for all the three architectures, where the scheduling is performed by a set of arbitration cells. It receives an n-bit one-hot code from the PG block, and the first cell to begin the arbitration cycle is the only one for which P equals 1. Each cell j receives the request $R(j)$, the priority signal $P(j)$ and a signal, named $ImedOut(j–1)$, that says if the previous cell in the arbitration order has granted or not their request. Also, each cell j has an output signal $G(j)$ set only if $R(j)$ equals 1, and if $P(j)$ or $ImedOut(j–1)$ equals 1.

Internally, the cells of the *random* and *rr_ripple* arbiters are structured in a wrap-around ripple loop, where the cell 0 looks for the *ImedOut* of the cell n-1. In the *rr_comb* there is no such loop, which eases the synthesis and timing analysis in automatic tools that does not support combinatorial loops [4]. In fact, the *rr_comb* arbiter is composed by two set of arbitration cells. The first one (named *ppe_not_round*) implements a programmable priority encoder without wrap-around connection. It performs the arbitration beginning in the cell j for which $P(j)$ equals 1, and finishing in the cell n-1. If none of them grants any request, them the requests 0 to j-1 are arbitrated by using the second set of arbitration cells, which is based on a static priority encoder (named *spe*).

2.2. Priority Generator

The PG architecture is not the same for all the three arbiters. In the *random* scheduler, it is based on a ring counter loaded with a one-hot code at reset – $P(0) = 1$. At each clock cycle, it performs a shift operation, changing the priority code sent to the PPE. It does not depend on the current state of the grant outputs (the gray solid width line in Fig. 1).

In the round-robin arbiters, the PG block is based on a n-bit parallel-parallel register also loaded with a one-hot code at reset – $P(0) = 1$. However, at each arbitration cycle (not at clock cycle), it is updated by a function applied over the new state of the grant lines (the gray solid width line in Fig. 1). Such function ensures that the request granted in the current arbitration cycle will have the lowest priority level in the next arbitration cycle.

3. SYNTHESIS RESULTS

The arbiters were described in VHDL by using parametric structures which allow the automatic resizing and the synthesis of instances with different number of request lines by only changing the input parameter n.

They were synthesized to the FPGA Altera EPF10K30ETC144-1x by using Quartus II Web Edition 2.2 [5]. Fig. 2 shows the number of LCs spent by each architecture for configurations varying from 2 to 10 requests.

The results show that the random arbiter is the cheapest architecture. However, it is known that it does not offer fairness and can carry a packet to suffer starvation. In the other side, round-robin arbiters ensure fairness and starvation freedom, but are more expensive. Comparing the two round-robin, architectures, the *rr_comb* is the cheapest one and, furthermore, has no combinatorial loop, which is hard to be synthesized or analyzed by some tools. Its architecture is detailed in [4], which gives a good overview about arbiters for routers of interconnection networks.

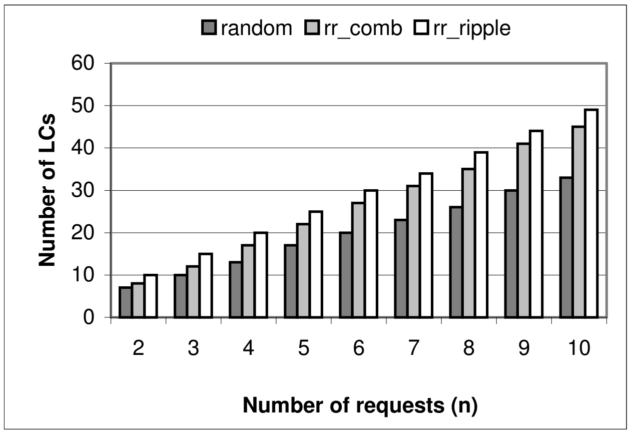

Fig. 2. Synthesis results.

To give some idea about the costs of the presented arbiters, we compare its costs with the entire costs of a deterministic router for NoCs presented in [6]. Such router uses a 4-request exhaustive round-robin arbiter [1]. In a 32-bit, 5-port, 4-word buffers configuration, the router uses 30 LCs in the same FLEX10K device. If we consider the costs of five arbiters for the three architectures presented here, the *random* would occupy 85 LCs, the *rr_ripple* would demand 100 LCs, and the *rr_comb* would spend 110.

4. CONCLUSIONS

In this paper, we presented the implementation of VHDL parametric models for three distributed arbiter architectures for Networks-on-Chip. Results shown that the models allows the automatic generations of arbiters. Next works include the implementation of these models in SystemC to perform a system-level evaluation in order to measure the impact of the arbiter architecture in the network performance.

5. ACKNOWLEDGMENTS

This work was supported by the "Assembléia Legislativa do Estado de Santa Catarina" ("Artigo 170" program).

6. REFERENCES

[1] F.G.M.E. Santo, C. A. Zeferino, Projeto e Avaliação de Árbitros para Redes-em-Chip . Hífen. Uruguaiana-RS, 2002. pp.85-90.
[2] P. Guerrier and A. Greiner, "A Generic Architecture for on-Chip Packet-Switched Interconnections", *DATE'2000*, IEEE Press, 2000. pp.250-256.
[3] J. Duato et al., *Interconnection Networks: an Engineering Approach*, IEEE CS Press, 1997. 515p.
[4] P. Gupta, P.; N. McKeown, N. "Designing and Implementing a Fast Crossbar Scheduler", *IEEE Micro***,** v.19, n.1, Jan.-Feb. 1999. pp.20-28.
[5] Altera. Quartus II Installation & Licensing for PCs Manual. San Jose : Altera Corporation, 2002. 54p.
[6] C. A. Zeferino, A. A. Susin. "SoCIN: A Parametric and Scalabe Network-on-Chip". *to appear in SBCCI'2003.*