USING VCI IN A ON-CHIP SYSTEM AROUND SPIN NETWORK

H. CHARLERY, A. GREINER, E. ENCRENAZ, L. MORTIEZ, A. ANDRIAHANTENAINA
UNIVERSITY PIERRE AND MARIE CURIE, PARIS

KEYWORDS: Network On Chip, VCI, SPIN, Wrapper, Formal Checking

ABSTRACT: The micro network SPIN (Scalable Programmable Integrated Network) is a packet-switched system-on-chip interconnection. This technology provides a very general communication mechanism between the different virtual components connected in the system. Moreover the bandwidth increases linearly with the number of embedded processors. This paper describes the main features of the SPIN micro-network (VCI/SPIN wrappers and routers). We also focus on parallel architecture problems such as deadlocks and memory coherency. Then we present some results about a comparison between SPIN and a bus system (PI-Bus).

INTRODUCTION

The SPIN concept [1][2] of high rate architecture of communication for system-on-chip drifts from the acquired experience in the parallel calculator domain. These machines have important requirements in bandwidth. They often use networks of multistage interconnection that are composed of point-to-point links and routers, as a substitute to the traditional "bus system" [3].

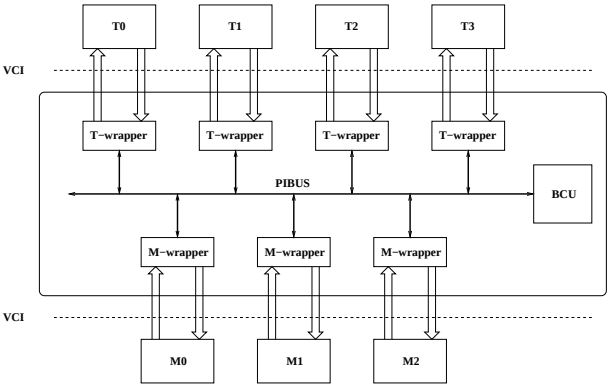

Fig. 1. Example of PI system interconnecting some VCI components

In system-on-chip (SoC), the communication bus is often the bottleneck, and this trend can only become more pronounced with the increase of the integration capacities [4]. The technology of the packet-switched micro-network SPIN is a possible answer to this problem. However, to permit an easy migration of an architecture using a bus toward an architecture using a switched network, it is necessary to permit the existing component reuse (IP cores). Therefore, the network must provide to the system designers the same interface and the same kind of services than a traditional bus. The VCI standard [5] (Virtual Component Interface), normalized by the VSIA consortium, defines a communication protocol like "shared address space", that can be as well implemented

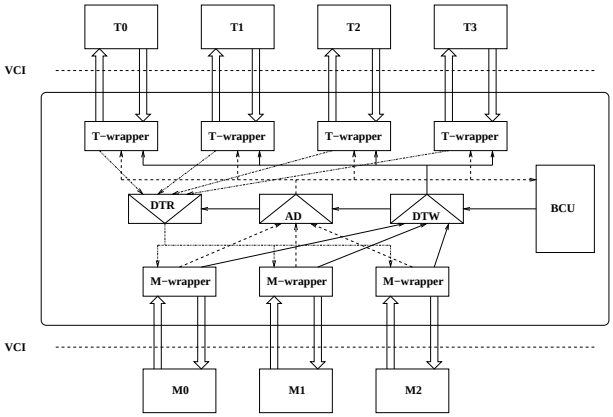

Fig. 2. Example of AMBA system interconnecting some VCI components

by a traditional bus system like PI Bus [6], Figure 1, or AMBA Bus [7], Figure 2, as by a packet-switched network, Figures 3. We will see how the technology of the packet-switched network SPIN can be used to provide a generic interconnection mechanism according to the VCI standard.

VCI has many advantages. The VCI interface introduces a complete delinking between the design or the choice of the system components (cores of microprocessor, DSP or specialized coprocesseurs), and the choice of the communication devices (classic bus, hierarchical bus, switched network, etc...). The extreme simplicity of the stream control mechanism defined by VCI, makes the access interfaces to the network easier to design. Finally, VCI supports the split transactions.

A connection to the SPIN network has the following aspect : a component with a VCI interface connected to a VCI/SPIN initator wrapper or target wrapper in order to switch from the VCI world to the SPIN world or inversely.

This article is divided into three parts. The first part explains how to provide VCI interfaces for SPIN, using

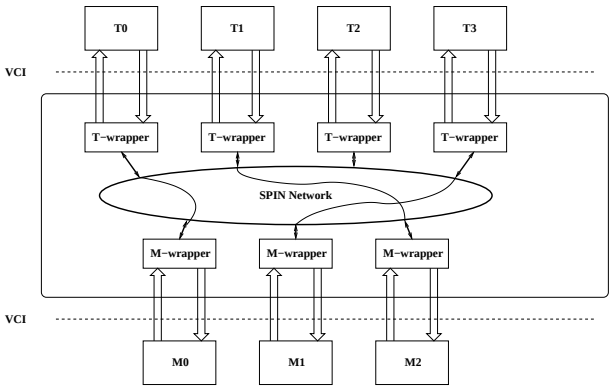

Fig. 3. Example of SPIN system interconnecting some VCI components

"VCI/SPIN wrappers" that achieves the translation between the two protocols (VCI and SPIN). The second part is about deadlock problems, the solution found. The memory coherency will also be treated. And we will finished with some results of a comparison between SPIN and the PI-Bus, a bus system. Two test boards will be presented, the first for a synthetic traffic, and the second for real application.

THE SPIN NETWORK

The Structure Of A SPIN Packet

The informations that circulate on the SPIN network are `packets`. A SPIN packet is a cells sequence. A cell, also named word is a set of 36 bits, Figure 4. The first word of a packet possesses a `Begin Packet(BP)` flag and the last word a `End of Packet(EP)` flag.
The 36 bits of a word are distributed as follow : 32 bits of data, 3 bits of tag that permits to mark the words, and a parity bit. The EP and BP flags are included in the tag field. Besides, it can take the value ERR, to indicate that the data inside the cell is not correct.

The Routing Of A SPIN Packet

The first word of a SPIN packet constitutes the header of the packet and contains necessarily the number of the destination port. This number, coded on 10 bits is used by the network to route the packet toward the destination. Every port is therefore identified by an unique number which is imposed by the network topology. This number cannot be modified.

The VCI Standard

The VCI standard makes the hypothesis that all components of the system share the same address space. The initiators send `read` or `write` requests toward the targets, that are identified by the address most significant bits (MSB). This is the same way as we proceed on a bus. Therefore, there are two types of VCI packets. The initiators send some *request packets* constituted of one or several addresses (in case of burst). The targets send back

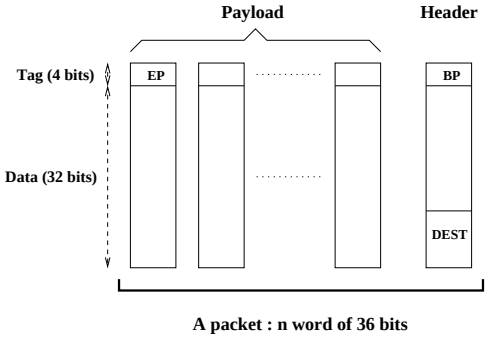

Fig. 4. Structure of a SPIN packet

the *response packets*. The packets have no fixed length, and are ended by the `End of Packet` flag. The *response packet* has the same length than its corresponding *request packet* and all requests receive responses. Every word of a *VCI request packet* is made of about 180 bits among which we can find stream control bits, data (in case of `write packets`), address and request's definition. Every word of a *response packet*, is made of about 60 bits which are the stream control bits, data, thread identifier, and acknowledge for the errors management.
The access mechanism to the network is extremely simple, since every component communicates with it such as it was a FIFO. Several initiators can therefore, send many simultaneous requests. And an initiator can send a new request (n+1) without for the previous request (n) answer. This possibility leans on a labeling mechanism of the pairs (request/response), and is used by *multi-threads processors* to hide the latency of the memory accesses.

THE VCI / SPIN WRAPPERS

General Principles

We call "wrapper" a material component that achieves a translation between two communication protocols.
Since two groups of VCI packets exist ("request" packets and "response" packets), it is necessary to define a specific translation mechanism for each of them. A `read request` doesn't contain the same information than a `write request` (because there are no data in a `read request`). The same technic is used for the responses. Therefore, there are 4 types of VCI packets:
- case a) a "read request" VCI packet of N cells (burst of N reads) will generate a SPIN packet of N+1 cells : a word by address, in addition to the header word.
- case b) a "write request" VCI packet of N cells (burst of N writes) will generate a SPIN packet of 2N+1 cells : two words by address, in addition to the header word.
- case c) a "read response" VCI packet of N cells will generate a SPIN packet of N+1 cells : a word by address, in addition to the header word.
- case d) a "write response" VCI packet of N cells will generate a SPIN packet of N+1 cells : a word by address, in addition to the header word.

Thanks to the optimisation signals present on the VCI interface, we can optimisate the traffic on the SPIN net-

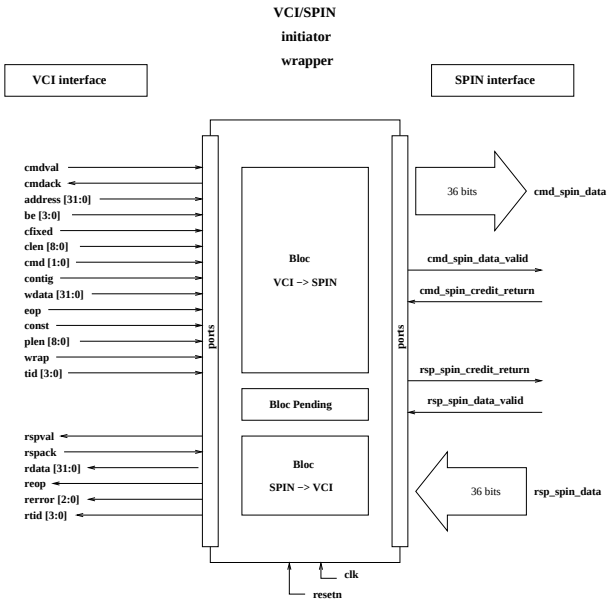

Fig. 5. The VCI/SPIN initiator wrapper

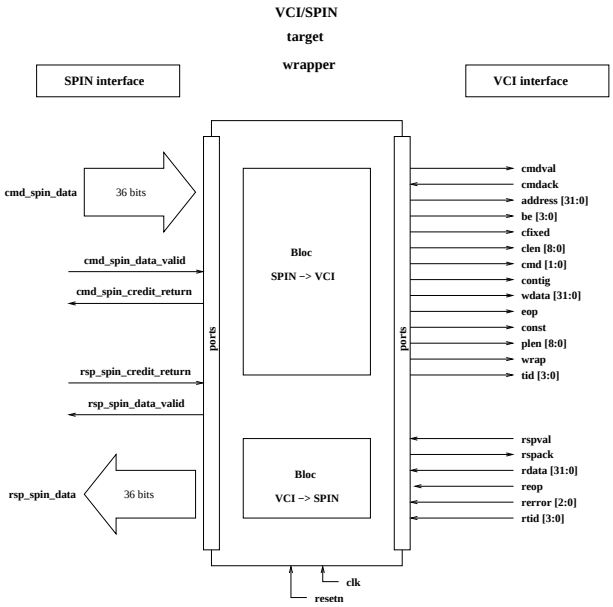

Fig. 6. The VCI/SPIN target wrapper

work.
The initiator wrapper, instead of sending n cells containing n adresses in a request packet, can just send 2 cells : one for the first address, and the other with informations such as the number of address and the way to obtain the next addresses (if they are constants or contiguous or wrapped).
Since there are two types of VCI components (the initiators and targets), there are therefore, two types of wrappers (Cf figures 5 and 6). The initiator's wrapper performs the translations VCI $\rightarrow$ SPIN in the cases a) and b), as well as the translations SPIN $\rightarrow$ VCI in the cases c) and d). The target's wrapper performs the translations VCI $\rightarrow$ SPIN in the cases c) and d), as well as of the translations SPIN $\rightarrow$ VCI in the cases a) and b).
Because the requests and the responses are fundamentally asynchronous, in each, of the two wrappers (initiator and target), there will be two independent Finite State Machine (FSM) to treat the "request" packets in one direction, and the "response" packets in the other.

The Initiator Wrapper

The initiator wrapper, in addition to the format translations described in the previous subsection, has two other important functionalities.
It decodes the address most significant bits and to determine the number of the destination port which will be placed in the header of the SPIN packet. It does this "`transcodage`" according to the "`routing table`" that defines the structure of the memory card. The memory card assigns to every target components a particular segment of the addressable space. These routing tables are obviously identicals for all the wrappers, because they constitute a global characteristic of the system. So, they must be inserted in every initiator wrapper. This function of the target selection, traditionally assumed by the bus controller, is inserted here in all initiator wrappers. It is one of the prices to pay to have an scalable network architecture. Incidentally, the initiator wrapper must indicate in the SPIN packet header its own port number, to make the target wrapper know the port number on which it must send the response packet.
Otherwise, the "advanced" VCI standard allows an initiator to send a second request without waiting for the previous one response. It uses then the Transaction IDentifier (TID) field of the VCI cell, that permits to define a request "number". Obviously, the wrapper must not transmit two requests with the same TID. Therefore, it manages a hanging request table, identified by their TID. The wrapper detects the routing errors, that occur in the case where the address most significant bits designate a nonexistent target.

The Target Wrapper

The target wrapper is simpler, because it transmits one by one the requests to the target : it waits to have received the last word of the response from the i request before beginning to transmit the first word of the request $i + 1$.
We can notice that two independant FSMs are absolutely necessary to treat the requests and the responses if we want to avoid some case of `deadlocks`.
Besides, a deeper study , using the formal verification techniques should allow to validate the different solutions adopted to suppress the deadlock cases.

DEADLOCK AND FORMAL CHECKING

Generality

In a packet switching network conveying both request and response packets on the same medium, the packets can be in a configuration such as none can arrive at its destination, blocking the ones others. We call this state `deadlock`.

Previously, we saw that a request packet generate a response packet.
We suggest that separate the network in two sub-networks, one for the requests, and the other for the responses, removes the deadlock risks. Requests and responses can't block the ones others anymore.
The formal checking techniques were used in order to validate this assumption. We want to demonstrate that the network which not distinguishes the requests and the responses, contains deadlocks, and the one that distinguishes them, doesn't contains deadlocks. For the modeling we used the `Promela` language, and for the checking as well as the simulation, the `Spin` model checking of Holzmann[8]. The choice of ProMeLa (for `Protocol Meta Language`) came from the need of being able to model distributed, parallel, and asynchronous systems in which concurrent processes communicate through channels.
After the modeling, the system, is simulated and checked with the `spin` tools, and its `xspin` graphic interface. These tools allow to seek possible blockings, portions of codes never reached, to find the invariants, the non-progression cycles and also to check Linear Temporal Logic (LTL) properties. The absence of deadlock can be modelled by a LTL property like the following : *each initiator will always be able to send a new packet* (what implies all packets is finally acknowledged).

Modelisation And Analyzes

This study was done on a SPIN micro-network containing four RSPIN routers and four subscribers. These routers are connected according to a binary fat-tree structure.
Each router has two upward ports and two downward ports. It is half less than the original version (SystemC) of the router, but this simplification was necessary in order to reduce the risks of combinatorial explosion. We consider that the solution found for a binary fat-tree structure can also be applied to a quaternary fat-tree structure.
In order to converge as soon as possible towards a deadlock, we decided to reduce as much as possible the storage capacity of the components.
The initiator must be able to receive words of the response packet without waitting to have sent all the words of the request packet. Moreover an initiator cannot address itself, or address another initiator.
These models of initiator and target components, were instanciated several times and connected to the network ports. The Figure 5, shows a deadlock in a system containing two initiators and two targets. The packets size was fixed at 8 words.
The subscriber `initiator 0` communicates with the subscriber `targets 3`, and `initiator 2` with `target 1`. In the router `RSPIN 2`, the two answers $1 \rightarrow 2$ and $3 \rightarrow 0$, are blocked by the two requests, respectively $0 \rightarrow 3$ and $2 \rightarrow 1$.
Deadlocks were highlighted by simulation and checking.

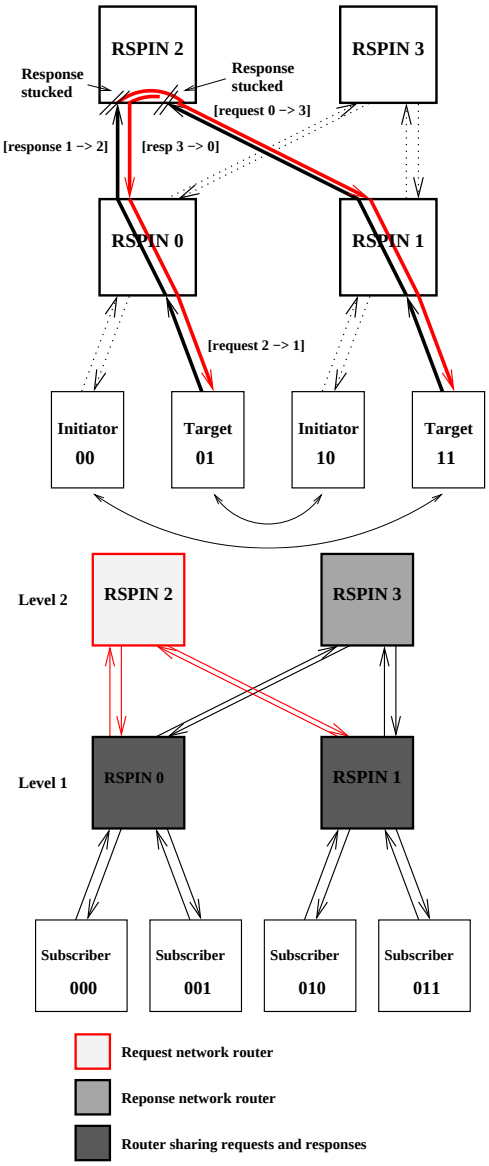

Fig. 7. Detailed scheme of a deadlock and the separation request/response as solution

The Topology Solution

The separation of the requests and the responses was concretely done by the division of the original network in two sub-networks, one for the requests, in red, and another for the responses, in black.
This division is carried out by the routers of first level, of which the higher ports are specialized for the requests or the responses [1]. In a first level router, superior left ports are reserved to the requests and those of right-hand side to the responses.
This topology is being validated.

SYSTEMC VALIDATION : THE OVERLOAD TEST

For the simulation, we use a tool developed in the LIP6 laboratory : the CASS simulator [9][10] (CASS for Cycle

[1]Cf Figure 6: Detailed diagram of a deadlock and the separation request/response

Accurate System Simulator). The inputs of this simulator are C/C++ models (RTL level), permitting very fast *cycle-true / bit-true* simulations.
We also validate the systems with SystemC. To do that, the models were a bit modified. Like CASS, SystemC permits a *cycle-true / bit-true* simulation. However, it is slower than CASS, but well-known by the industrials. It is considered now as a standard for the system-on-chip design and simulation.
The overload test consists in generating a synthetic traffic. Special initators named GAP (packet generator and analyser), send read packets to RAMs. The GAP parameters are the packet length, the number of packet to send, and the load.
The test board is composed by a 32 ports network (16 routers, 32 wrappers) connecting alternatively a GAP and a RAM, in order to provide a fairly distribution.
The results are presented below. The graphic shown on figure 8 represents the average latency versus the load. Each GAP sends 100000 packets of 8 words. We can see that the overload occurs later on SPIN (28%) than on PI-Bus (4%). But PI-Bus is better than SPIN for small or less loaded systems.

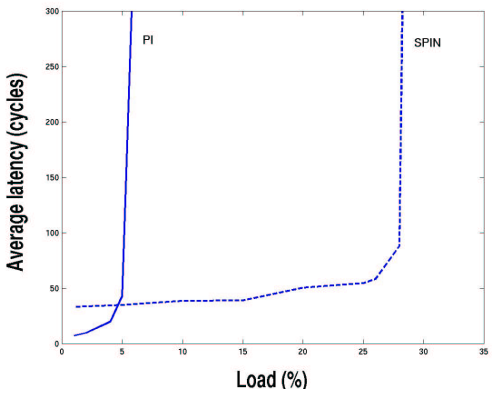

Fig. 8. Average latency versus the load

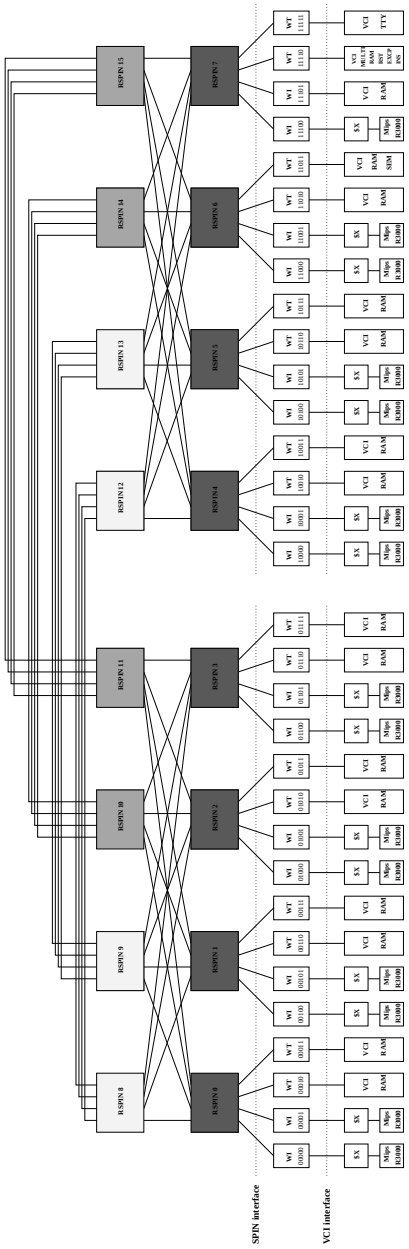

Fig. 9. The test board for the real application

SYSTEMC VALIDATION : A REAL APPLICATION

Memory Coherency

Memory coherency is ensured by a software implementation. Material solutions are too complex and too expensive for the time beeing.
The temporary solution adopted, was to consider the whole memory uncachable. This way to proceed was not realistic.
To make the memory coherent, two rules should be satisfied. The first one is that a thread is locked on a processor. In this way, we avoid their migration, thing permitted initially to allow an optimal use of the processors.
The second rule is to attribute to each thread a memory zone for its local datas.

The Application Used

Two major problems were solved for this application. The first one is about memory coherency, the second one deals with the datas distribution on several RAMs.
The goal of the application used is also to overload the interconnect by generating large simultaneous informations transfers. It is in the continuity of the test with the GAPs presented in the previous subsection. The difference stands in the nature of the traffic and the components used. The initateurs are MIPS R3000 processors, so the traffic is a little more varied than simple bursts of reading. The application performs copies of big arrays in memory segments. More than big bursts of reads or writes, we test also the semaphore engine, and the synchronization between threads.

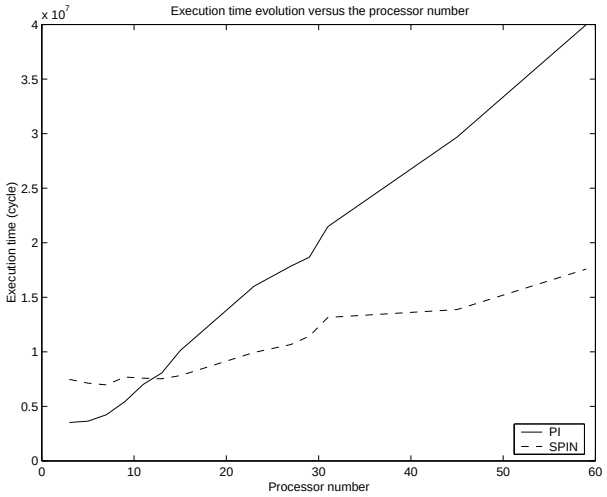

Fig. 10. Execution time evolution versus the logical processors number

Test Bench And Results

The test board used contains 32 subscribers. Our choice for the network size was made on a 32-port because a 16-port was too small and a 64-port too heavy for simulations. The 32 subscribers distribution is done according to an equitable way. Each router of the first level, is connected to two initiators and two targets, except on the last one, where there are one initiator and three targets [2].
The initiators are multicontext Mips R3000 processors, connected to data/instruction caches. The interest of that is to make the economy of VCI ports. The targets are RAMs and a TTY (display).
The results are shown on the graphic 10. It represents the simulation time (cycle) according to the processors number. The bandwidth increases with the number of processors, so we deduce that SPIN allows a higher bandwith than PI-Bus does. But under 12 processors, for this application PI-Bus is better. As we seen in the previous subsection, PI-Bus is better for small systems which doesn't need high bandwidth.

CONCLUSION

We have presented the SPIN micro-network and how it is used with the VCI protocol.
Now we are evaluating the impact of using VCI optimisation signals, on the traffic. We also focus on clustering techniques. It is better to put in the same cluster initiators and targets which have important communication together.
The comparisons SPIN versus PI-Bus show that it is very costly to put paralellism on small architecture, or on not overloaded networks. The latency becomes in that case an handicap.
The solution of separate requests from responses, to avoid deadlocks is still in validation. Perhaps we should demonstrate it theoricaly.
Finaly, all these works and results are in the public domain, under GPL Licence.

[2] Cf Figure 9: The test board for the real application

AUTHORS

Herve Charlery, Laurent Mortiez and Adrijean Andriahantenaina are PhD students in the ASIM Departement of the LIP6 Laboratory.
Pr. Alain Greiner and Dr. Emmanuelle Encrenaz are respectively professor and doctor in the ASIM Departement.

Email : Herve.Charlery@lip6.fr

REFERENCES

[1] P. Guerrier et A. Greiner, "A Generic Architecture for On-Chip Packet-Switched Interconnections", *Proceeding of the Design Automation and Test in Europe Conference 2000 (DATE 2000)*, March 2000.

[2] A. Adriahantenaina, H. Charlery, A. Greiner, L. Mortiez et C. A. Zeferino, "SPIN: a Scalable, Packet Switched, On-chip Micro-network", *Proceeding of the Design Automation and Test in Europe Designers' Forum 2003 (DATE 2003)*, Munich, March 2003.

[3] J. Hennessy et D. Patterson, "Computer Architecture, A Quantitative Approach - 2nd Edition", *Morgan Kaufmann Publishers*, San Francisco, CA, USA, 1996.

[4] H. de Man, "Education for the Deep Submicron Age : Business As Usual ?", *Proceedings of the 34th Design Automation Conference*, Anaheim, USA, March 1997.

[5] Virtual Socket Interface Alliance, "Virtual Component Interface Standard - Draft Specification, v. 2.2.0", *http://www.vsia.com* (document access may be limited to members only), August 1997.

[6] Open Microprocessor Systems Initiative, "PI-Bus Draft Standard Specification", *ftp://www.omimo.be/ftp/standard/omi324.ps*, 1994.

[7] ARM Company Ltd, "Advanced Microprocessor Bus Architecture Specification", *http://www.arm.com/Pro+Peripherals/AMBA/*, 1997-99.

[8] G. J. Holzmann, "The Model Checker Spin", *IEEE transaction on software engineering, vol. 23*, n5, May 1997.

[9] D. Hommais et F. Petrot, "Efficient Combinational Loops Handling for Cycle Precise Simulation of System on a Chip", *IEEE Euromicro Conference*, pp. 51-54, 1998.

[10] F. Petrot et al., "Cycle-Precise Core based Hardware/Software System Simulation with Predictable Event Propagation", *IEEE Euromicro Conference*, pp. 182-187, 1997.