SPIN: a Scalable, Packet Switched, On-chip Micro-network

Adrijean Adriahantenaina (UPMC/LIP6)
Hervé Charlery (UPMC/LIP6)
Alain Greiner (UPMC/LIP6)
Laurent Mortiez (UPMC/LIP6)
Cesar Albenes Zeferino (UFRGS)

Abstract

This paper presents the SPIN micro-network that is a generic, scalable interconnect architecture for system on chip. The SPIN architecture relies on packet switching and point-to-point bi-directional links between the routers implementing the micro-network. SPIN gives the system designer the simple view of a single shared address space and provides a variable number of VCI compliant communication interfaces for both initiators (masters) and targets (slaves). Performance comparisons between a classical PI-bus based interconnect and the SPIN micro-network are analyzed.

Keywords

Systems-on-Chip. Networks-on-Chip. Embedded Systems.

1. Introduction

The technology scaling improvements will allow the building of Systems-on-Chip (SoCs) with from several dozens to hundreds of components within a four-billion-transistor chip until the end of this decade [1]. This will also allow the development of new applications in the fields of telecommunication, entertainment and consumer electronics. Such systems will require communication templates providing several dozens of Gbit/s [2], which still must be reusable to meet time-to-market requirements.

The reusable communication templates typically used in current SoCs are based on the bus approach, using either a single shared bus [3] or a hierarchy of buses [4]. However, such approach has strong drawbacks: A bus does not scale with the system size as the bandwidth is shared by all the components attached to it. Furthermore, as the number of cores increases, the capacitance load grows, degrading the bus operating frequency.

Some recent works [1][2][5] have proposed the use of integrated switching networks as an alternative approach to interconnect cores in SoCs. The overall idea is that such networks, also called Networks-on-Chip (NoCs), meet three of the major key communication requirements for future SoCs: reusability, scalability, and parallelism.

This work presents the evaluation and comparison of two on-chip communication templates based on bus and NoC approaches. The communication architectures are compared by simulating cycle-true, RT-level models running synthetic workloads.

This paper is structured as follows. In section 2, we present the generic architecture used to compare the two communication templates, which are described in section 3. The bit-true, cycle-true simulation environment is presented in section 4. A first experiment analyzing the global latency as a function of the number of terminals is presented in section 5. A second experiment describing the latency as a function of the offered load is described in section 6. Finally, section 7 presents some concluding remarks.

2. The Generic Architecture

The architecture used in this work to evaluate and compare the performance of the two communication architectures is shown in Figure 1. It is based on two kinds of components: initiators and targets. The system can have different number of cores for each type. The "initiator" components are traffic generators, which send requests to the "target" components. The "target" component sends a response as soon as it receives a request.

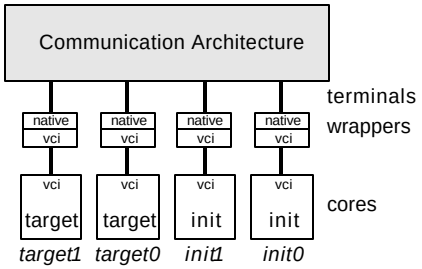

Figure 1. The reference system.

All the components in the system are VCI-compliant [6] and, since the communication architectures are typically based

on a native communication protocol, each one of them uses a wrapper to adapt both protocols, as is shown in Figure 1.

In the VCI standard, all initiators share the same address space. An initiator sends a read or write request to a target that is identified by the address msb bits or by using a point-to-point connection. The target processes the request and sends a response to the former. In the advanced VCI standard, an initiator can issue a new request without waiting for a response to the first request, due to the use of a split protocol. Requests (and responses) must be tagged with identifiers, which allows such requests and requests threads to be interleaved and even allows responses to arrive in a different order (which can occur in a network-based architecture using adaptive routing). Such feature is only available in the Advanced VCI, which is the VCI level used in this work.

3. Communication Architectures

The on-chip communication architectures evaluated in this work are based on real templates. The first one is PI-Bus [3], which is an on-chip bus specified by the Open Microprocessors Initiative (OMI). The other one, is SPIN [2], an experimental on-chip network developed by the LIP6 laboratory at University Pierre et Marie Curie in Paris. Such architectures are described in the following paragraphs.

3.1 PI-Bus

PI-Bus is a processor independent and demultiplexed architecture with data and address buses scalable up to 32 bits. It is multimaster capable and needs a bus controller for operation. Such controller must implement a mechanism to arbitrate which master is granted the requested bus ownership. The bus controller is also responsible to perform the address decoding, in order to determine the target of a bus operation, and other functions as time-out control and slave access control. Masters and slaves require VCI/PI wrappers to be connected to PI-Bus.

3.2 SPIN

SPIN (*Scalable Programmable Interconnection Network*) is a packet switching on-chip micro-network, which uses wormhole switching, adaptive routing and credit-based flow control. It is based on a fat-tree topology, which is a tree structure with routers on the nodes and terminals on the leaves, except that every node has replicated fathers. In a full 4-ary fat-tree topology, illustrated in Figure 2, there are as many fathers as children on all nodes (routers). Such topology produces a non-blocking network with a performance that scales gracefully with the system size.

Links are bi-directional and full-duplex, with two unidirectional channels. The channel's width is 36-bit wide, with 32 data bits and 4 tags bits used for packet framing, parity and error signaling. Additionally, there are two flow control signals used to regulate the traffic on the channel.

In SPIN, packets are defined as sequences of 32-bits data words, with the header fitting in the first word. A 8-bit field in the header is used to identify the destination terminal, allowing the network to scale up to 256 terminals. The payload has unlimited length defined by two framing bits (Begin Packet / End of Packet).

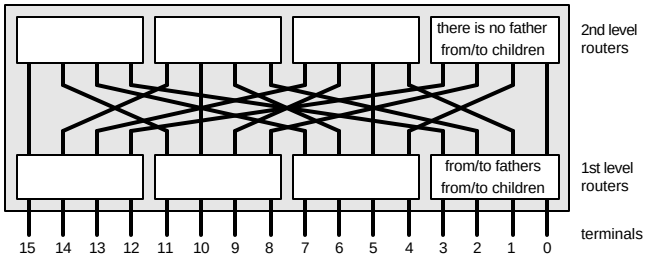

Figure 2. A 16-terminal SPIN network.

SPIN uses wormhole switching, which is a packet switching approach in which a router can forward a packet as soon it receives the packet header. The packet is broken up into flits (*fl*ow control un*it*), which are the small units over which the flow control is done, and them it is pipelined through the network at the flit level. In SPIN, a flit is only one word (36 bits). The input buffers have a depth of 4 words, which results in cheap routers.

Routing in SPIN is adaptive and distributed. When a packet flows up the tree, each router can choose anyone of the available links to forward the packet. When it reaches a router that is a common ancestor with the destination terminal, the packet is turned around and then it is forwarded by using a deterministic path (the only one between the ancestor router and the destination terminal). Such approach reduces contention in the network, improving the message latency and the network throughput.

The basic building block of the SPIN network is the RSPIN router shown in Figure 3. It includes eight ports, each one with a pair of input and output channels compliant with the SPIN link. Internally, RSPIN includes a 4-words buffer at each input channel and two 18-words output buffers, shared by the output channels. They have a greater priority when competing with the input buffer to use an output channel, which allows to reduce the contention, whilst minimizing the head-of-line blocking by freeing the queues in the input buffers.

RSPIN contains a partial 10×10 crossbar, which implements only the connections allowed by the routing scheme: all the packets flowing down the tree can be forwarded to children and only such packets can use the output buffers when the required output channel is busy. Nevertheless, only the

packets incoming from children can flow up the tree and be forwarded to the fathers.

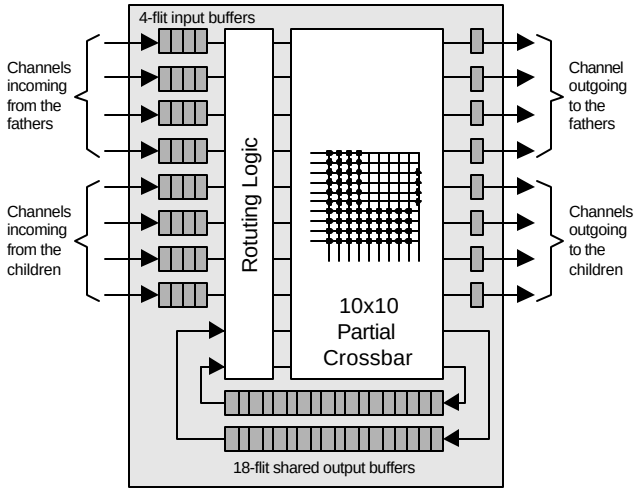

Figure 3. Organization of RSPIN.

4. The Simulation Environment

To evaluate and compare the communication architectures, we used a cycle-precise simulator named CASS (*Cycle-Accurate System Simulator*), developed at LIP6 [7].

The CASS simulator can be described as a simulation accelerator for the well-known SystemC simulation environment. This simulator engine is cycle-based and can be used for a special class of SystemC simulation models: The abstraction level must be RTL, and each component should be described as a set of communicating finite state machines. With the present version of CASS, the system architecture is specified as a structural VHDL file which instantiates the components and defines the interconnection net-list.

In this work, cycle-accurate CASS/SystemC simulation models have been written for the RSPIN router, and the VCI/SPIN wrappers. All those models were derived from the corresponding synthesizable VHDL models. Therefore, those models are both cycle-accurate and bit-accurate. The same approach was used for the PI/VCI wrappers, and the PI-bus controller.

5. Latency/Number of Terminals

This section presents some initial results for a basic workload applied to 4-, 8-, 16-, 32-cores symmetrical systems based on PI-bus and SPIN. Figure 4 shows a 4-terminals symmetric system with 2 initiators (*init*0 and *init*1) and two targets (*target*0 and *target*1).

5.1 The workload

In this first experiment, we intended to measure the number of cycles spent by the communication architecture to deliver all the request and response messages in a *pooling*. A *pooling* is defined as the messages exchanged when each initiator sends a request to each target. For instance, in a 4 components system like the one of Figure 4, a pooling will have 4 request messages and 4 response messages. In Figure 4, one can see the request messages sent by *init*0 to *target*0 and *target*1.

It must be highlighted that channels in SPIN are 32-bit words. Therefore a single-data VCI write transaction is packed into a 3-word SPIN packet (header+address+data).

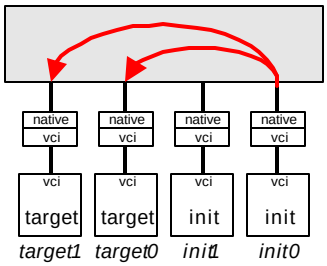

Figure 4. The pooling workload

5.2 Experimental results

Figure 5 presents the number of cycles spent by the PI-Bus and SPIN architectures to perform a pooling in symmetrical systems with different sizes. As one can see, for this workload, SPIN outperforms PI-Bus only in systems with more than a dozen of cores. However, the message size used in this work can be considered the worst case for a network like SPIN. For larger packets, such as a cache line transfer, the performance of SPIN is improved, because the costs of the packet header overhead decreases as the packet payload increases.

Figure 6 presents the number of cycles spent by the SPIN architecture when the initiators are disabled to use the split protocol of VCI. As one can see, if the split protocol is not used, the message latency becomes significantly greater, and if one compares the numbers for Tables 1 and 2, one can observe that SPIN will not overcome PI-Bus without using the split protocol.

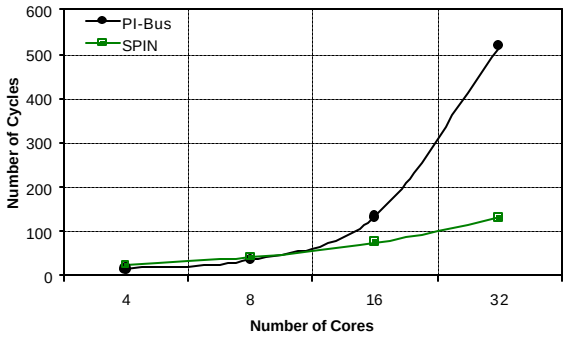

Figure 5. PI-Bus / SPIN for several system sizes

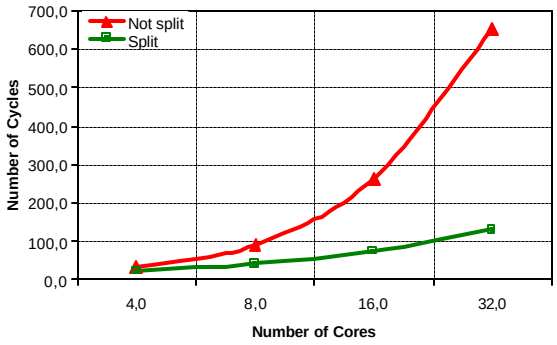

Figure 6. SPIN: split / not-split transactions

6. Latency/Load

In any interconnection architecture, there is a saturation threshold, appearing when too many initiators are trying to generate too much traffic. In this new experiment, we intend to measure the saturation threshold for both the PI-bus and the SPIN architecture, as a function of the offered load.

6.1 The workload

We focus on a symmetrical, 32 terminals network (16 initiators / 16 targets). This network contains 16 routers and 32 wrappers. Each VCI initiator sends request packets with a length L of 8 words (actually, a cache line read). The cache line addresses are randomly distributed between all VCI targets.

The variable parameter is the offered load. The gap G is the number of cycles between two successive request packets sent by the same initiator. In this workload, G is a random variable whose average value GM can be precisely controlled. The offered load is the percentage of the channel bandwidth used by each initiator. With the above definitions, the offered load is L / (L + GM). An offered load of 100% corresponds to the case GM = 0.

In order to take into account the contention of the interconnect architecture (PI-bus or SPIN), and have a meaningful latency measurement, the requests have a timestamp, and are stored in an "infinite" FIFO located in each initiator. The latency is measured from the date of the initiator request to the reception of the response packet by the initiator. We use the split transaction feature of the VCI interface.

6.2 Experimental results

Figure 7 presents the average latency versus offered load obtained for the 32 terminals SPIN network (16 initiators / 16 targets). The simulation has been done for about 100K request/response transactions. In this experiment, the shared output buffers of the RSPIN router were disabled. It appears that the SPIN micro-network has a latency of about 30 cycles, and saturates for an offered load of 28 %.

As expected, for very small load, the PI-bus has a lower latency than the SPIN network, but the saturation threshold appears for an offered load larger than 4%.

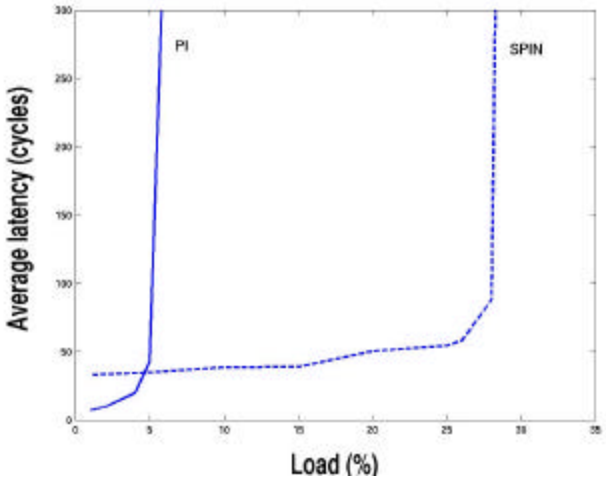

Figure 7. SPIN latency / load

7. Conclusion

This work presented a performance comparison between two on-chip interconnect: a classical system bus, and the SPIN micro-network. The initial results show that for a workload with short messages, SPIN outperforms PI-Bus in systems with a dozen of terminals or greater. Moreover, for a 32 terminal architecture, the SPIN micro-network supports an offered load of 30%, where the PI-bus saturates for an offered load larger than 4%.

References

[1] L. Benini, G. De Micheli, «Networks on Chips: A New SoC Paradigm», *IEEE Computer*, Jan. 2002, pp.70-78.

[2] P. Guerrier, A. Greiner, "A generic architecture for on-chip packet-switched interconnections, *DATE'2000*.

[3] Siemens, *OMI 324: PI-Bus – Ver.0.3d*. Munich, Siemens AG, 1994.

[4] *IBM CoreConnect Bus Architecure*, http://www-3.ibm.com/chips/products/coreconnect/index.html

[5] W. J. Dally and B. Towles, «Route Packets, Not Wires: On-Chip Interconnection Networks», *DAC'2001*.

[6] Virtual Socket Interface Alliance, *Virtual Component Interface Standard, OCB 2.1.0*, March 2000.

[7] Pétrot Frédéric, Hommais Denis, Greiner Alain, "A Simulation Environment for Core Based Embedded Systems" *Proceeding of the 30th Annual Simulation Symposium*, Atlanta, Georgia, April 1997, pp. 86-91.