A Low-Power Multiprocessor Architecture For Embedded Reconfigurable Systems

C. Amerijckx, J.-D. Legat

Microelectronics Laboratory
Université Catholique de Louvain
Place du Levant, 3
B-1348 Louvain-la-Neuve
Belgium
Tel: +32.10.47.80.61
Fax: +32.10.47.25.98
Email: Amerijckx@dice.ucl.ac.be

1 Abstract

In this paper, we introduce the architecture of a new embedded field programmable processor array (E-FPPA) which consists of a low-power multiprocessor system embedded with standard programmable logic blocks and memory. Each block (processor, programmable logic,...) is coupled to a transfer controller (TC) responsible of all the transfers between blocks. Instead of using a classical crossbar interconnection network, we propose a low cost hierarchical ring which combines simple interface and high performance communications when data locality is observed. Based on the E-FPPA, high performance reconfigurable systems can be easily built and we demonstrate that this architecture is an interesting alternative to traditional DSP for low-power applications. By using the 8-bit CoolRisc processor [1,2], an E-FPPA including a cluster of 16 processors, 16 TC, working respectively at 25 and 50 MHz, and 1kbytes data SRAM for each processor, consumes 2 W with a peak performance of 1200 Mops. The chip size has been evaluated in 0.35 µm to 52 mm^2.

2 Introduction

Currently, state-of-the-art FPGA offers more than 250K gates on a chip. In [3], they are showing that we can expect more than 4 millions gates on a single FPGA by 2010. Two problems will arise with such vast arrays of random logic:

- The interconnection becomes a real bottleneck if flat technology is used.
- Random logic is inefficient to map most of the computation that requires datapath (as in signal processing, cryptology,...).

Different alternative FPGA architectures have been recently proposed to overcome these limitations. Aggarwal et al. [4] introduced an architecture

overcoming the interconnection problems. They are using a hierarchical FPGA (HFPGA) architecture in which areas of classical random logic are connected by fast long-distant communication busses. It should be noted that, in this case, we are in the same conditions as in multi-FPGA systems: the design has to be partitioned between areas of logic cells.

Datapath implementation is also an intensive field of research. The DP-FPGA proposed by Cherepacha [5] is an architecture in which all functional units are composed of a 4-input 4-output lookup table (LUT), a fast carry chain and an output latch. This architecture is well suited for applications requiring intensive arithmetic calculation on large numbers. The main drawback is that control logic can not be easily implemented because it mainly requires functional units on one bit. Another way to implement datapath structures is to change the design of the functional unit itself. In [6], [7], [8] for example, the functional unit is replaced by a reconfigurable ALU, combined with a special routing architecture. A similar approach can be found in [9], [10] and [11] where they are using multi-bit ALU. Even if these architectures support more complex operations, they are not well suited for non-standard operations and mapping of these functions could fail.

In accordance with Hauck who recently reveals the future of reconfigurable systems [12], we propose a new type of field programmable architecture where the basic LUT-based functional units are replaced by more complex blocks which can be small processor core, memory, reconfigurable logic, … (Figure 1). This architecture consists of an array of processors embedded with other elements and appears as an extension of emergent field programmable processors arrays (FPPA). Embedded-FPPA or E-FPPA will allow a hardware/software co-design of systems which is in case of complex applications an attractive alternative to standard powerful DSP or large multi-FPGA boards. Moreover, E-FPPA by using low-power processor units consumes much less than equivalent DSP chip.

In FPPA architecture, the main problem is the design of the interconnection network which cannot be standard programmable connections as in classical FPGA. Crossbar-based networks are generic in term of connections but suffer from the lack of scalability. On the other hand, hierarchical ring networks which combines simple interface and high performance communications when data locality is observed have been lately studied [16]. These studies have shown the superiority of hierarchical ring networks over mesh networks. From our point of

view, hierarchical ring network is well adapted to FPPA and a low-power implementation of this kind of networks will be described.

We first present the E-FPPA architecture and the interconnection network in section 3. Then, in section 4, the implementation of a processor-based block will be described. It will allow to discuss the consumption issue of such new architecture in comparison with up-to-date DSP. Finally, conclusions will be given in section 5.

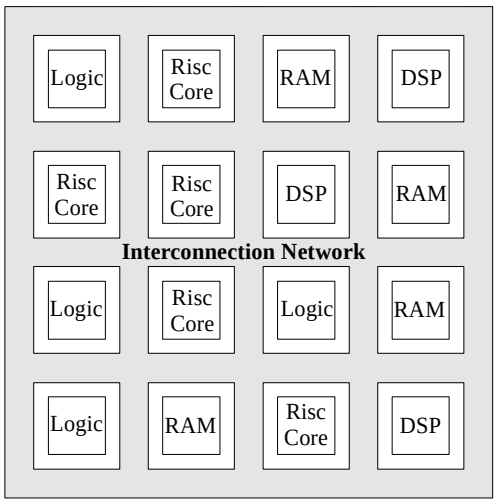

Figure 1: Proposed architecture

3 E-FPPA Architecture

3.1 Interconnection network

As already point out in the introduction, the main challenge in FPPA is the interconnection network which has to combine simplicity, efficient communication, generic message protocol and scalability.

A single bus such as the PCI bus [13] does not meet these requirements. The latency of the bus increases with the number of blocks in the system. It is not easily scalable (complex bridge have to be developed). Only one transmission can be performed on the same time on the bus which induces a low bandwidth/Watt. It is also necessary to design complex arbitration bus system. A way to overcome the latency problem would be the use of a crossbar architecture. Unfortunately, crossbars are not scalable and the implementation cost is high.

Direct network architectures are an interesting alternative to traditional bus systems. Direct networks could be seen as a mesh of dimension n. A mesh of

dimension n has k_0 x k_1 x ... x k_{n-1} nodes along each dimension i, with k_i>1. The most common notation is k-ary n-cube network. As mentioned in [14], one of the main advantages of this architecture is its high scalability. The second advantage is the small point to point connections allowing to work at a very high frequency. Moreover, these networks and their performances are well know [15].

The interconnection network which has been selected for the E-FPPA is a k-ary 1-cube, which are ring with k nodes. More precisely, the blocks are interconnected thanks to a hierarchical ring architecture (Figure 2a). Ravindran et al. [16] have proven that small hierarchical rings are much more efficient than mesh of higher dimension.

In this architecture, each block (B) is connected to a ring of level-(i) by a transfer controller (TC) which handles all the interface between the block and the ring network. Each level-(i) ring is connected to a level-(i-1) by an inter-ring transfer controller which manages the transfer between rings. Of course, this system works well if data locality is observed. This means that most of the transfers should occur on a same ring; transfers through the ring of lower level should only occur occasionally. If this property is not observed, the frequency of the lower level ring can be increased to have better performances [17]. An implementation example is shown in Figure 2b.

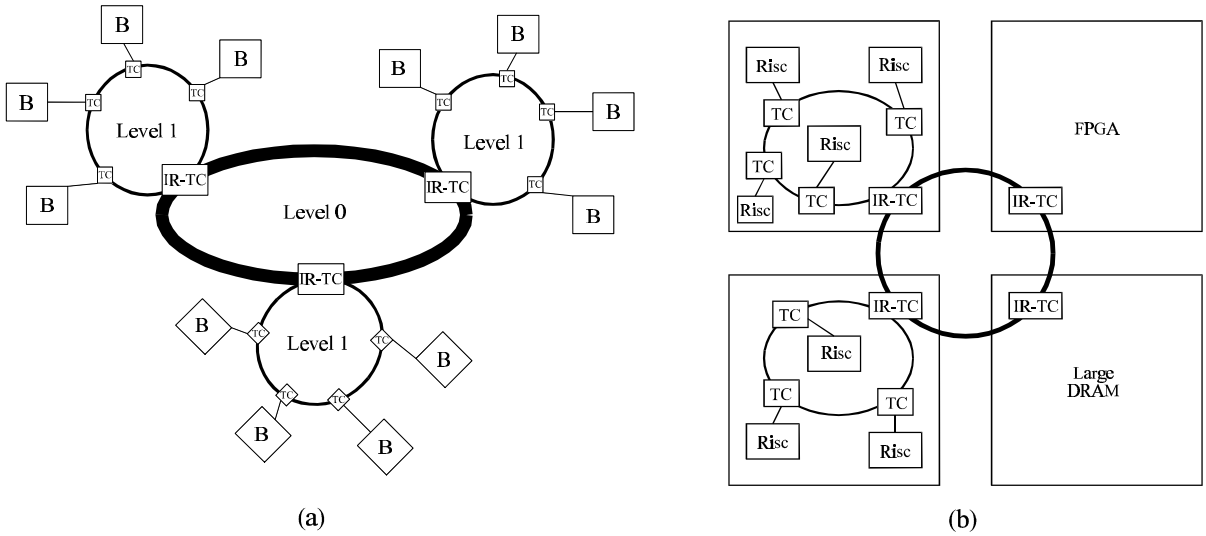

Figure 2: Hierarchical ring in a E-FPPA and its implementation

3.2 Flow control and ring access

In order to have fast communications, the way all the blocks are accessing the ring is important. The most known flow control is the token (token ring). In this architecture, all the blocks are connected to a single bus. The ring access control is done by a token. If a block has the token, he owns the access to the ring and can transfer data. The main disadvantage of this ring is that only one block can use the ring at a time.

In a slotted ring [18], [19], the bus is divided into a certain number of slots. A block has to wait for a free slot to be able to transfer data. In other words, the slots have the same role as the token in a token ring; as soon as a block detects a free slots, a transmission can occur. The main advantage of the slotted ring in comparison with the token ring is that many blocks can transfer at the same time, allowing a better use of the bandwidth. In the register insertion ring, register (FIFOs) are inserted in each TC (Figure 3). A message arriving at TC(i) will be directly forwarded to TC(i+1) if TC(i) is not transmitting and if the message is not for block(i). It will be stored in the input FIFO if TC(i) is transferring and if the message is not for block(i). Finally, it will be stored in the block(i) FIFO if the message is for block(i). If a TC(i) wants to transfer data, it will directly forward the message to TC(i+1) if the bus is not busy or store the data in the FIFO if the bus is busy.

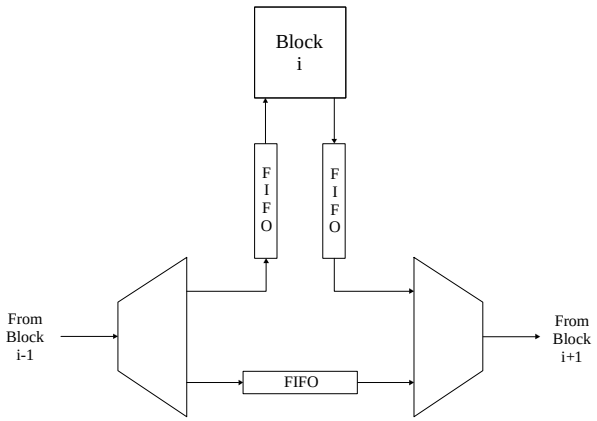

Figure 3: The TC in a register insertion ring

A comparison of these three rings with similar routing protocol is given in [20]. The token ring is always the worst one. Slotted and register insertion rings have similar results, even if the register insertion rings shows better results for small flows. As opposed to the slotted ring, only the busses where a transfer is occurring are active in the register insertion ring. This implies a lower power

consumption when long busses are involved. This reason justifies our preference for the register insertion ring network.

3.3 Switching techniques

To completely define the interconnection network, an appropriate switching technique has to be selected. The packet is defined as a set data including a header and the data itself. *Store and forward switching* is a routing technique in which a whole packet is stored in a TC before being sent to the next TC (Figure 4a). This implies the buffers in each TC can become large, depending on the allowed size of the packets.

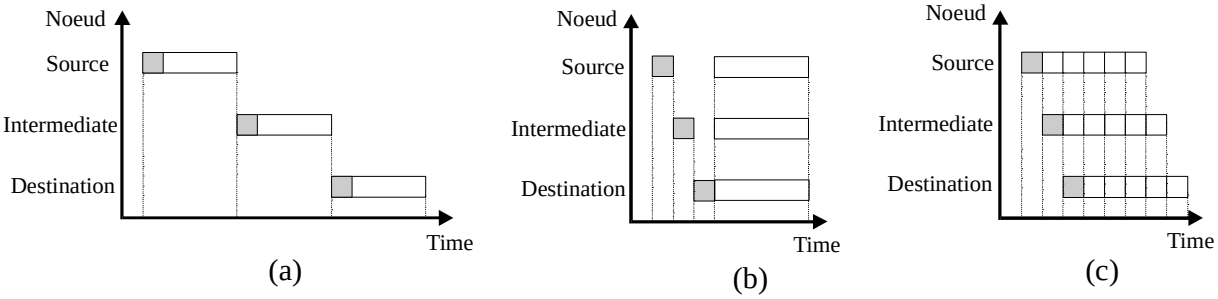

Figure 4: Switching techniques

In *circuit switching*, the header is first sent to the receiver in order to reserve the resources between the sender and the receiver. Once this is done, the data block is sent (Figure 4b). In comparison to store and forward routing, the latency of this routing algorithm is lower [14]. In *wormhole routing* (Figure 4c), a packet is divided into a certain number of flits (flow control digits). The size of a flit is system dependent, for example on the channel width. The first flit is the header flit and governs the route of the message; the next flits follow the same way in a pipeline fashion. If the header flit is stopped in a TC because the next channel is busy, the TC does not have to buffer all the remaining flits. He can leave them in the buffers along the established route. This is an important advantage in comparison with store and forward routing where the complete message have to be buffered. The size of the buffer, in wormhole routing, can be as low as one flit. In comparison with circuit switching, the resources are released earlier. For all these reasons, we choose the wormhole routing for the E-FPPA interconnection network.

3.4 Inter-ring Transfer Controller

The IR-TC is responsible of data transfers between rings of different levels. The simplest way to achieve the transfer is to use a 2x2 crossbar with buffers at the input (Figure 5a). However, Karol et al. [21] have shown that, in some cases, the output flow can be limited to 58.6% of its maximal value. Actually, packets can

be blocked inside the FIFO, even if their output is free, because previous packets stored in the FIFO do not have a free output.

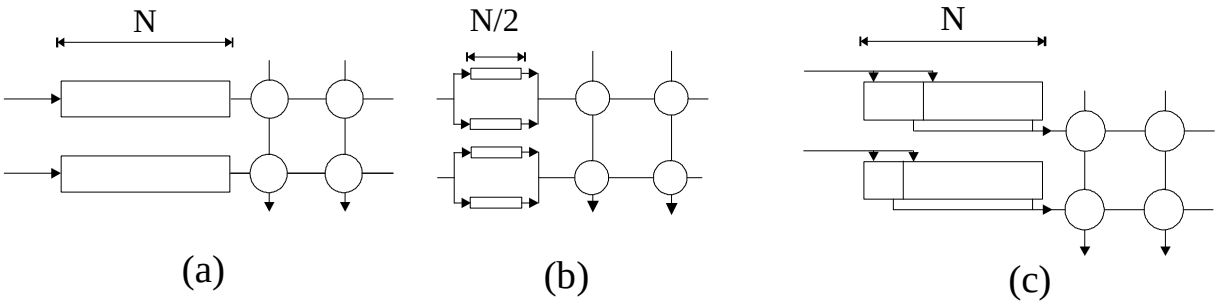

Figure 5: IR-TC architectures

A solution to this problem has been proposed by Tamir [22]. Instead of having only one FIFO at each input, they are placing as much FIFO as outputs (Figure 5b). For a 2x2 crossbar, we will have 2 FIFO at each input. Mc Keown [23] has shown that this architecture is optimal in terms of flow. But this solution is not optimal in terms of memory allocation because the input FIFO is statically allocated. The solution to this problem is to dynamically allocated each FIFO [22], [24] (Figure 5c). In this case, we will have, at each input, two FIFO whose size is variable. The sum of both sizes remains fixed. This last architecture has been selected for the IR-TC because it gives the best flow with the best memory allocation.

4 Processor-based Block Implementation

The implementation of a block containing a small RISC processor, its data memory, its program memory and the TC has been evaluated in a 0.35µm CMOS technology. The CoolRisc processor [1,2] is used for its low power consumption. As we can see in Table 1, it has a exceptional MIPS/Watt ratio. The CoolRisc is a small 8-bit harvard architecture processor with a separated bus for input and output data, a multiplier, an ALU and 16 static registers.

This processor offers many possibilities to reduce as much as possible the power consumption. For example, the data and the program memories are divided into a small memory and a large one (Figure 6a). Pieces of code or of data which are frequently accessed are placed in the small memory while infrequently accessed parts are stored in a larger one. Since most of the time only the small memory is read, the power consumption is reduced.

Microprocesseur	Frequency at 1MIPS	CPI	MIPS @ f	VDD	Power @ 1 MIPS	MIPS/Watt
80C31	12	12	1.7 @ 20Mhz	3.0	30 mW	33
68HCxx	4	4	1 @ 4MHz	3.0	8 mW	125
PIC16Cxx	5	5	1.6 @ 8MHz	3.0	7.5 mW	133
PIC17Cxx	5	5	3.2 @ 16Mhz	3.0	14 mW	71
CoolRisc816	1	1	10 @ 10 MHz	3.0	0.4 mW	2500

Table 1 : Comparison of processors

Another way to reduce the power consumption is to use the Freq instruction which allows to divide the internal clock frequency by a factor of 2, 4, 8 or 16. The power consumption can be further reduced by placing the processor in low-power standby mode with the Halt instruction. It will restart when an event or an interrupt occurs.

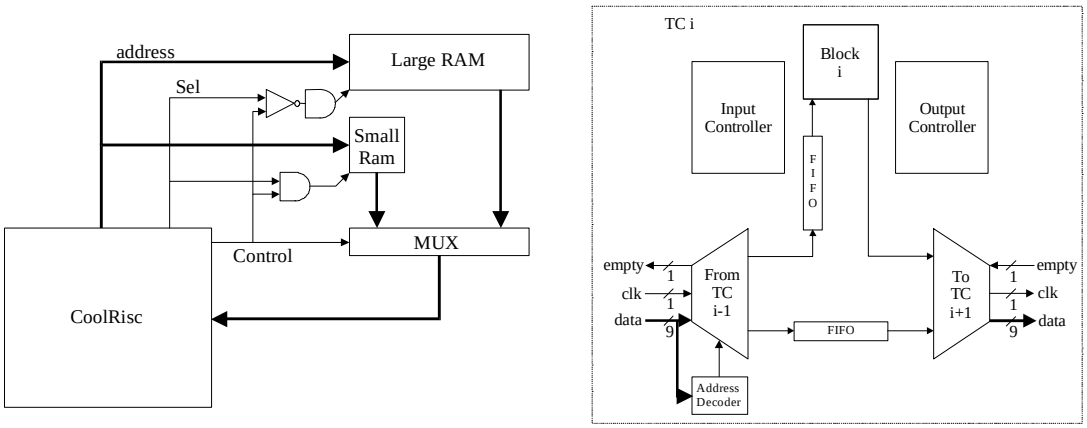

Figure 6: (a) Hierarchical memories, (b) Transfer Controller

The total size of both data and program memories in this evaluation are respectively 1k x 8 bits and 1k x 22 bits. These memories are implemented by using dual ported Ram.

The transfer controller (Figure 6b) includes two FIFO of 4 bytes depth, an address decoder at the input to route the message to the block i or to the next TC, an input controller (decoding and transferring the messages to the block) and an output controller (coding and transferring the messages to the next TC). The data bus includes 8 bits for the data and one bit for header flit strobe. An empty signal is used to indicate the FIFO status. The clock signal is the strobe for the flits.

Based on these hypothesis, an E-FPPA including a cluster of 8 processor-based blocks with one IR-TC consumes 2 W at 25 and 50 MHz, for respectively the processor and the TC, with a peak performance of 1200 Mops. This gives a ratio of 568 Mops/Watt. The chip size has been evaluated to 52 mm² in 0.35 µm.

These results can be highlighted by given corresponding ratios for some popular DSP from TI (Figure 7).

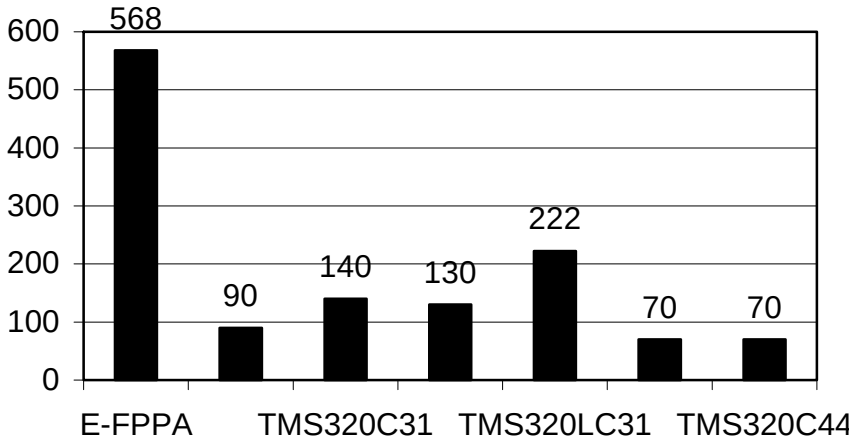

Figure 7: MOPS/Watt comparison

5 Conclusions

A new embedded field programmable processor array has been presented. It is particularity well suited to low power application where hardware/software co-design can be applied. The complexity of both the 8-bit processor and its associated transfer controller is 4500 gates. This block can deliver up to 75 MOPS, corresponding to an attractive performance of 3000 Mops/Watt without including any memory. The size of this elementary block is 0.24 mm2 in 0.35 µm.

6 Acknowledgment

This work has been funded by the Mivip European Esprit project LTR 22527.

7 References

[1] J.-M. Masgonty at al., "Low-Power Design of an Embedded Microprocessor Core", in *Proceedings on the European Solid-State Circuits Conference 96,* Sep. 17-19, 1996.

[2] C. Piguet at al., "Low-Power Design of 8-b Embedded CoolRisc Microcontroller Cores", in *IEEE Journal of Solid-State Circuits*, vol. 32, no. 7, pp. 1067-1078, July 1997.

[3] "The National Technology Roadmap for Semiconductors", in *Semiconductor Industry Association*, 1994.

[4] A.A. Aggarwal, D.M. Lewis, "Routing Architectures for Hierarchical Field Programmable Gate Arrays", in *International Conference on Computer Design*, pp. 475-478, 1994.

[5] D. Cherepacha, D.M. Lewis, "DP-FPGA: An FPGA Architecture Optimized for Datapath", in *VLSI Design*, vol. 4, no. 4, pp. 329-343, 1996.

[6] R.W. Hartenstein at al., "An FPGA Architecture for Word-Oriented Datapath", in *Proceedings of Canadian Workshop on Field Programmable Devices FPD'94*, June 13-16, 1994.

[7] R.W. Hartenstein at al., "A New FPGA Architecture for Word-Oriented Datapaths", in *Proceedings of the 4th Workshop on Field Programmable Logic and Applications FPL'94*, September 7-10, 1994.

[8] R.W. Hartenstein, J. Becker, "A Two-level Co-Design Framework for Xputer-based Data-driven Reconfigurable Accelerators", in *Proceedings of the 30th Annual Hawaii International Conference on System Science*, January 7-10, 1997.

[9] C. Ebeling, D.C. Green, P. Franklin, "RaPiD - Reconfigurable Pipelined Datapath", in *International Workshop on Field-Programmable Logic and Applications*, pp. 126-135, 1996.

[10] E. Mirsky, A. DeHon, "MATRIX: A Reconfigurable Computing Architecture with Configurable Instruction Distribution and Deployable Resources", in *IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 157-166, 1996.

[11] C.A. Moritz, D. Yeung, A. Agarwal, "Exploring Optimal Cost-Performance Designs for RAW Microprocessors", in *IEEE Symposium on Field-Programmable Custom Computing Machines*, 1998.

[12] S. Hauck, "The Future of Reconfigurable Systems", in *Proceedings of the 5th Canadian Conference on Field Programmable Devices*, June 1998.

[13] T. Shanley, D. Anderson, "PCI System Architecture", Addison-Wesley Publishing Company, 1995.

[14] L. M. Ni, P. K. McKinley, “A survey of Wormhole Routing Techniques in Direct Networks, in *IEEE Computer*, pp. 62-76, Feb.93.

[15] W. J. Dally, “Performance Analysis of k-ary n-cube Interconnection Networks”, in *IEEE Transactions on Computers*, vol. 39, no. 6, pp. 775-785, June 90.

[16] G. Ravindran, M. Stumm, “A Performance Comparison of Hierarchical Ring- and Mesh-connected Multiprocessor Network”, in *Proceedings of HPCA’97*, pp. 58-69, 1997.

[17] Z.G. Vranesic, M. Stumm, D. M. Lewis, R. White, “Hector: A Hierarchically Structured Shared-Memory Multiprocessor”, in *IEEE Computer*, pp.72-79, Jan. 91.

[18] L. A. Barroso, M. Dubois, “Cache Coherence on a Slotted Ring”, in *Proceedings International Conference on Parallel Processing 91*, St Charles, Illinois, vol1, pp. I230-I237, Aug. 91.

[19] L. A. Barroso, M. Dubois, “Performance Evaluation of the Slotted Ring Multiprocessor”, in *IEEE Transactions on Computers*, vol.44, no. 7, pp. 878-890, July 95.

[20] L. N. Bhuyan, D. Ghosal, Q. Yang, “Approximate Analysis of Single and Multiple Ring Networks”, in *IEEE Transactions on Computers*, vol. 38, no. 7, pp. 1027-1040, July 89.

[21] M. Karol, M. Hluchyj, S. Morgan, "Input versus output queueing on a space division switch", in *IEEE Transactions on Communications*, 35(12), pp. 1347-1356, 1987.

[22] Y. Tamir, G.L. Frazier, "Dynamically-Allocated Multi-Queue Buffers for VLSI Communication Switches", in *IEEE Transactions on Computers*, vol. 41, no. 6, pp. 725-737, June 92.

[23] N. McKeown, "Achieving 100% throughput in an Input-Queued Switch", in *Proceedings of IEEE Infocom'96*, March 96.

[24] Y. Tamir, H.C. Chi, "Symmetric Crossbar Arbiters for VLSI Switches", in *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 1, 1993.