Experimental Testing of PLATO, a Reconfigurable Active ATM Network Node

6 AUTHORS, INCLUDING:

Dionisios N. Pnevmatikatos
Foundation for Research and Technology - …
95 PUBLICATIONS **1,392** CITATIONS
SEE PROFILE

Euripides Sotiriades
Technical University of Crete
30 PUBLICATIONS **183** CITATIONS
SEE PROFILE

Stamatis G. Kavvadias
Technological Educational Institute of Crete
24 PUBLICATIONS **78** CITATIONS
SEE PROFILE

Experimental Testing of PLATO, a Reconfigurable Active ATM Network Node

Apostolos Dollas, Dionisios Pnevmatikatos[1], Nicolaos Aslanides,
Euripides Sotiriades, Stamatios Kavvadias, Sotirios Zogopoulos

Department of Electronic and Computer Engineering
Technical University of Crete
73100 Chania, Greece
Email: Dollas@mhl.tuc.gr

Abstract. An active, reconfigurable network node, named PLATO, has been designed and implemented. Version V.X1.0 was developed with a Xilinx Virtex XCV-1000 FPGA as a PCI board, with the capability of 256MBytes of SDRAM, 512KBytes of SRAM, and a UTOPIA level 2 based interface to 4 bidirectional 155 Mbps ATM links. This paper presents the implementation and testing of the PLATO system, as well as a priority enforcement scheme for transmission of TCP/IP packets over ATM networks.
Keywords: Active network, reconfigurable architecture, prototype

1 Introduction - Motivation

Active networks are those networks which, in addition to regular operations like routing of packets based on their header, also operate on the contents of the packets. The operations, which are performed on the packet contents with knowledge of the network environment, may be result into improved routing, faster operation of the protocol (protocol boosting, deployment of new protocols on existing hardware), or data extraction/modification. In general, research on active networks is performed in software due to its flexibility. Although the software approach is excellent for experimentation, it disallows for real-time applications on high-speed networks. Hardware for active networks has been developed in three separate projects, the ANN project at the Washington University of St. Louis [15][16], the Programmable Protocol Processing Engine (P4) project [18][19] (with several partners), and the PLATO project [29][30]. The PLATO project, the hardware development of which will be described in this paper, was developed by the Technical University of Crete, the Institute of Computer Science ICS-FORTH, and the Technological Educational Institute of Crete. All three hardware implementations of active networks are based on reconfigurable logic (Field Programmable Gate Arrays - FPGA's), with additional features like embedded processors (ANN project), multiple FPGA's for extra resources (P4), and multiple buffer memories (PLATO). Considering that research on active networks is at its infancy, we will not compare at length these systems, because what is more significant is the capabilities that such systems offer in general. We consider the three platforms to be generally similar in terms of

[1] Also at Institute of Computer Science, FORTH, Greece

their broad characteristics, but of different capabilities with respect to specific applications, and hence each system is useful in its own right.

PLATO, its architecture and key applications have been reported in [29]. The use of the PCI Pamette [1] as a rapid prototyping tool, in parallel with the development of the actual PLATO hardware has been reported in [30]. The work presented in this paper focuses on the first version of the PLATO system hardware, which is operational, and the methods by which it was tested. This work also presents a new application, not reported previously in [29] and [30], namely, the enforcement of TCP/IP priorities in TCP/IP traffic over ATM networks.

Section 2 of this paper briefly presents general research directions in active networks. Section 3 briefly presents the architecture and applications of the new system, while section 4 describes the implementation and testing of the PLATO v.X1.0 prototype. Section 5 presents the TCP/IP priority enforcement scheme, followed by our conclusions. Sections 2 and 3 include material that has been described in more detail elsewhere [29], but is deemed necessary to understand the remainder of the paper.

2 Related Work

Extensive work has been performed in the area of Active Networks. The ANTS project at MIT [4,5] introduced the notion of capsules. An ANTS capsule integrates a cell's data and a reference to a Java forwarding routine, which must be loaded (if not present) on the active node before the capsule can be processed. Smart Packets at BBN [6], also uses capsules to extend the diagnostic functionality in the network, using a specially developed language (Spocket) and run-time environment. The SwitchWare project at the University of Pennsylvania and Bellcore [7] relies on active packets, carrying limited functionality programs written in PLAN [8], and resident or loaded out-of-band active extensions to provide additional functionality and an active router infrastructure. In the CANES project at Georgia Tech and the University of Kentucky [9,10], the functionality of an active network node is divided between the Node Operating System and Execution Environments running on top of it, the latter of which correspond to composition methods for defining new services (e.g. ANTS, PLAN). Open signaling (Opensig), refers to an instance of programmable networks that clearly separates network control from information transport [11]. The Xbind [12] broadband kernel abstracts node resources and supports the programmability of the management and control planes in ATM networks, based on CORBA [13] middleware to incorporate multiple vendor switches. The Genesis project [14] at Columbia implements the Virtuosity kernel, which automates the spawning of network architectures in a process of *profiling*, *spawning* and *management*.

To our knowledge, only two projects have considered hardware platforms for active networks. The ANN project [15,16] at the Washington University in St. Louis devised a hardware implementation for a scalable active network node architecture for gigabit environments. Their platform consists of a general purpose CPU, one or two FPGA's, 64MB of memory and an ATM Port Interconnect Controller (APIC) chip connected on a PCI bus. In addition, their Router Plugins [17] research software platform is used as a framework for the development of a NodeOS. The Programmable Protocol Processing Pipeline (P4) project [18,19], has implemented a platform comprised of a set of ALTERA FLEX8000 FPGAs acting as processing elements (PEs) and a switching array selecting devices meant for processing.

Active network applications include active reliable multicast [5], self-organizing network caching [10], multicast video distribution [15], online auctions [4], active bridging [7], protocol boosters [19], active congestion control, distributed firewalls, and packet filtering. Software approaches to such services (e.g. ANTS, Smart Packets, PLAN and CANES), have historically been implemented over IP networks. In the context of ATM networks, even though QoS features are inherent, signaling complicates service creation.

3 Architecture of PLATO

The architecture of PLATO will be presented together with one particular design, as shown in Figure 1. This is deliberate, because the platform is mainly aimed to be a tool for experimentation and design library development, and hence was derived from a minimum set of functionality requirements, as follows:

- Ability to be connected to fiber or copper ATM networks, ability to be connected in the future with Ethernet (10/100 or Gigabit)
- Minimal delay in hardware processing of the cells, and ability to implement protocols with no need to communicate with the host processor
- On-board buffer space for streams of cells, on the payload of which, processing will be performed
- Ability to communicate with a general purpose computer, for further processing, downloading of statistics, or partial reconfiguration of the FPGA
- Extra connectors for future expansions

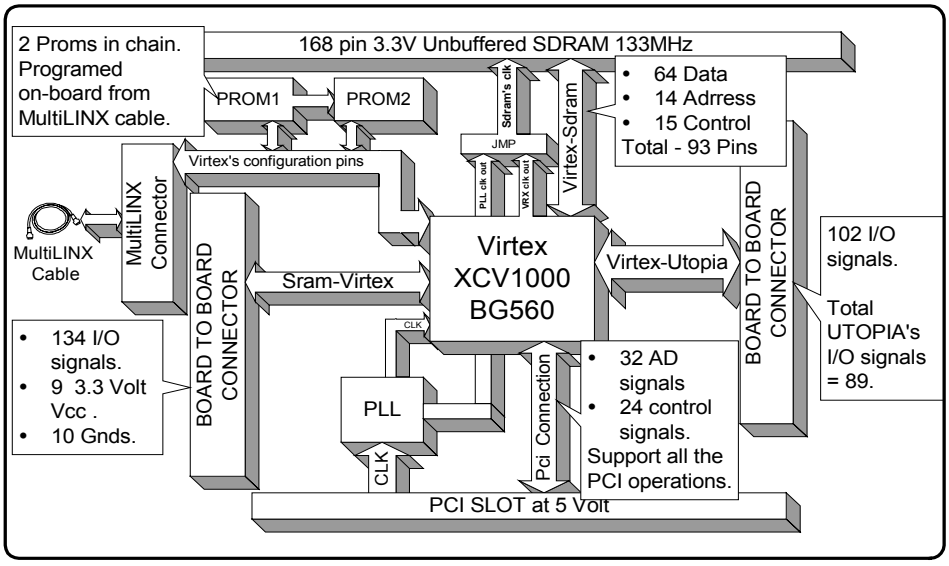

Figure 1. Architecture and Block Diagram of PLATO

The PLATO platform has a large FPGA, which in addition to the clock generation circuit, programming ports, etc. has four main ports:

- A UTOPIA level 2 port, to provide the physical connection to daughterboards with copper or optical fiber outputs – the daughterboard also has the UTOPIA level 2 - ATM framing circuitry
- A 256 MB 133MHz SDRAM port for buffer space
- An auxiliary port which will be used in another project [20] for SRAM look up tables
- A PCI bus port for communication with the host.

Although the general topology resembles several existing products, the need for new hardware was largely mandated by the numbers of pins needed for each port, as well as the voltages in each case (e.g. PCI at 5V, SDRAM at 3.3V, etc.). A notable difference with ANN is that PLATO does not have a general-purpose processor embedded in it. Although we recognize the usefulness of such a processor, the goal of PLATO was to experiment with the performance that the reconfigurable resources offer in several applications, and, the development of VHDL design libraries for operations such as cell disassembly and reassembly, payload extraction, etc.

4 Implementation and testing of the PLATO V.X1.0 System

The PLATO System, as shown in Figure 1, comprises of a PCI board with a large FPGA and ports for the SDRAM, the UTOPIA level 2 interface (through connectors) and SRAM interface (also through connectors). The architecture has been implemented in two versions, that have a different FPGA each, but share the same daughterboards. The version PLATO V.X1.0 is based on the Xilinx XCV-1000 Virtex device with a 560-pin BGA footprint, whereas the version V.A1.0 is based on the Altera APEX 20K400E device with a 652-pin BGA footprint. The large number of pins necessitated a correspondingly complex multi-layered printed circuit board (PCB), which comprises of eight and ten layers for the Xilinx and Altera FPGAs respectively. Both boards, as well as all daughterboards were manufactured at INTRAKOM SA, using the most aggressive technology that the company had available (in terms of conductor pitch and spacing). Both boards were assembled at the Institute of Computer Science-FORTH, and partially at the Microprocessor and Hardware Laboratory of the Technical University of Crete. Both boards are operational, but in this paper we will present the testing of the V.X1.0 system.

Following the manufacturing, the PLATO V.X1.0 system underwent functionality tests. These tests were largely download capability tests, connectivity tests, and speed tests. Specifically, all download modes of the Virtex part were tested successfully. Following these tests, simple counters to divide the clock and drive LED displays were implemented and tested successfully. In terms of connectivity, a ring counter to link all user I/O pins was implemented, and tested successfully, while checking each pin in an oscilloscope for double pulses that would indicate short circuits. This test passed as well, and indeed the speed of the system proved to be 172MHz with an 86MHz external clock source. The above tests were unrelated to the functionality of the PLATO system as an active network node, and were aimed at checks for design, manufacturing, and assembly flaws (none were found). While V.X1.0 was in design and manufacturing, the active 4x4 ATM switch was designed and prototyped using the PCI Pamette, as reported in [30]. The Pamette design was much slower than the corresponding PLATO design, but included all required functionality. Furthermore, a traffic load generator was implemented on a personal computer, and through the PCI bus cells of varying distributions were generated and processed in hardware by the Pamette.

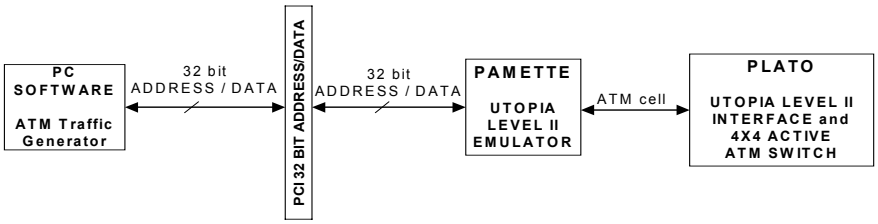

Figure 2. Experimental Setup to Test PLATO V.X1.0

Once the PLATO system was ready, the design migrated from the Pamette to PLATO, as shown in Figure 2. The migration was accomplished over two days (a period much shorter than the general tests described above). Using the previous experience, the Pamette was used as I/O for PLATO, so that the entire suit of tests could be verified on PLATO (albeit at a smaller speed). The test has the following steps: The traffic generator produces in software ATM cells that are send via the host's PCI bus to the PCI Pamette, which emulates the UTOPIA level 2 port and sends the cells to PLATO. The emulator, after receiving a sufficient number of cells, commences the transmission of the cells to the UTOPIA interface of the ATM switch. The cells are received by the UTOPIA interface of PLATO, and are placed in a FIFO. The ATM switch dequeues cells from the FIFO, extracts the VPI and VCI values, and based on these fields, accesses the connection table. This lookup produces the new VPI and VCI values, as well as the correct output link for this cell. The cell is then enqueued in the outgoing link's FIFO for later transmission. The emulator contains 8 FIFOs (4 receive FIFOs filled by the ATM traffic generator and 4 transmit FIFOs that return the switched cells to the traffic generation for verification and statistics). Therefore, to simulate the simultaneous arrival of 4 ATM cells to the switch we pipeline the transfer of the 4 ATM cells to the 4 input FIFOs, i.e. we transfer one word per cycle to one FIFO, the next word to the next FIFO and so on. The cell is transmitted from PLATO to the Pamette UTOPIA emulator and is later read via the PCI bus by the software. It is then possible to verify the correctness of the switching function, and collect statistics of its operation.

We tested the system using two input scenarios, which we will call the single input port scenario and the multiple port scenarios. In the single input port case we performed a systematic test of all paths through the switch. Using the software traffic generator, we supplied cells from each input port, destined to each of the output ports. Running the 16 experiments corresponds to transmission from any of the four input links to any of the four output links. The rate with which the traffic generator can generate cells is smaller than the service rate of the switch. This is a result of the fact that the traffic generator runs in software, whereas the switch is implemented in hardware. Therefore, the use of the PCI Pamette to inject the cells in the switch limits the input rate seen by the switch. Taking into account the latencies introduced by the various parts of the system (the PCI interface latency, the UTOPIA emulator latency etc), as well as the latency of the switch itself, we found that while a cell is been given to the input port of the switch, the switching of the previous cell is already complete. We have verified that all the combinations of input and output ports work correctly, and that no cells are lost, as expected, since there is no contention.

The multiple port scenario was performed, in order to test the switch under congestion situations. In our experiments we transmit 50 cells each time, distributed in a round-robin fashion to each input port. In this setting, if we vary the destination of the cells we can

achieve different congestion situations and levels. To investigate the behavior of the switch under different congestion situations, we performed two experiments. First we set the connections so that cells from input ports 0 and 1 are switched to output port 0, while cells from input ports 2 and 3 are switched to output ports 2 and 3 respectively. This scenario induces congestion on the output port 0 and we expect that at steady state, only 75% of the incoming cells will be able to be transmitted to the output. Half of the cells coming from ports 0 and 1 are expected to be dropped. Our experiments verified that the switch behaved correctly under this congestion, and that the other cells were indeed unaffected by the congestion as expected.

5 Priority Enforcement for TCP/IP over ATM Traffic

This application demonstrates some of the more sophisticated capabilities of PLATO V.X1.0, as it allows differentiated treatment of TCP/IP packets over ATM networks, depending on the priorities of the TCP/IP packets themselves. We have designed, fully simulated (post place and route, with timing analysis), and mapped onto the PLATO V.X1.0 hardware an application which, after disassembling ATM cells (as described in [29]) to extract their payload, checks the header of IPv4 packets. In these packets, we can check the *precedence* field of the *Type of Service* byte, which is contained in the first three bits of the second IP header byte. Each different value of precedence is treated differently in terms of routing precedence, leading to different rates of IP flows within ATM networks. The higher the precedence value is, the faster service the corresponding packet will receive.

One IP flow comprises of several ATM cells. We consider that these cells are switched in sequence, and through the same path. Although this assumption is not correct in theory, in practice routing tables get changed infrequently with respect to individual TCP/IP packets and thus the assumption is realistic. Our application would normally run at an intermediate node of the network, thus handling the full traffic, or it would act as a filter near the network edges. The application is based on the knowledge that the first packet of an IP flow contains the priority status of the entire flow. This information is extracted in the manner described in the previous paragraph, it is placed in a table within PLATO, and then it is used on every TCP/IP packet of the same IP flow that is transmitted over the ATM network. Priority queues are implemented in hardware, with grouping of same-priority IP flows.

Although in principle the idea is simple, its correct implementation entails substantial complexity, arising from the fact that each ATM cell as well as each TCP/IP payload of the ATM cell are processed at the full network speeds. In case of congestion, lower priority traffic is not merely rejected, but rather it is buffered and routed when the resources are available. The required speed for simultaneous operation of four 155Mbps ATM links has been achieved for PLATO. More specifically, PLATO can run this application at 53MHz. With the 16 bit datapath of the UTOPIA 2 interface to the ATM network, and four simultaneously operating links, the minimum target speed would be 155Mbps x 4 links / 16 bits in the UTOPIA level 2 datapath = 38.75 MHz (these results are from post place and route simulations, and thus they are very conservative). The operations of receiving a cell from an input (or a queue), writing a cell to a queue, or sending a cell to an output can be done in parallel, so that up to three cells can be simultaneously processed. Resource-wise, this application occupies 10% of the slices (logic) and 50% of the blockram (memory) of the Virtex XCV 1000 chip. From previous results [29][30] we have reported that the full 4x4 ATM switch (without the application), together with the SDRAM controller and the PCI interface

occupy less than 30% of the Virtex part, which means that we are well within the capabilities of PLATO V.X1.0 for the entire system.

The system has been simulated for traces of multiple cells, in all types of combinations and priorities. We will describe one such experiment, stating however, that the remaining experiments yielded correct results. In this experiment, we have a priority 5 cell coming from link 1, to be output to link 0. As soon as we have four bytes of this cell we check whether it corresponds to an IP flow. As it turns out that this is indeed the case, we place it in the queue with priority 5. When the entire packet is received it is transmitted to the UTOPIA level 2 port for output to the appropriate link. This process requires 68 cycles, of which 54 cycles are for the UTOPIA port (27 cycles to receive and 27 cycles to transmit a cell). While the above operation takes place, and while the first cell is written in the queue, another cell enters the switch from incoming link 2, for output at the link 3, with priority 0. This cell will be output as soon as the first cell transmission is completed, and this cell will require 83 cycles total (of which 54 cycles are for the UTOPIA). The difference between the 68 and the 83 cycles is due to the internal datapath. Specifically, the VCI, VPI and incoming link information (a total of 30 bits) are placed in a content addressable memory (CAM). Subsequently, the CAM is used to determine the address of the SRAM where the priorities and information about IP flows are kept. The response of the CAM is dependent on how the data is stored within. As the experiment continues, we receive from link 0 a cell, which is not part of an IP flow. This cell is processed normally and without the priority scheme. Following the reception of the third cell (and prior to its transmission) we receive a cell from incoming link 3 for outgoing link 2 with priority 7. This cell takes 62 cycles to be processed and exit the switch (including the 54 UTOPIA cycles).

The description of the above experiment shows how the PLATO system can implement routing schemes that even a high-end processor cannot process in high-speed networks.

6 Conclusions

In conclusion, the first version of PLATO, an experimental active network platform has been implemented and works properly for non-trivial active network applications. Such applications demonstrate that active networks are a promising direction in payload-aware processing of computer networks. In future work, the UTOPIA level 2 daugherboard will be tested and the system will be evaluated with actual loads. In addition, more active network applications will be developed for this platform.

Acknowledgements

This work was supported by the Greek Secretariat for Research and Technology (GSRT) and the European Social Fund through the PENED 99 program under contract 99EΔ 408. We thank the Xilinx Corporation and the ALTERA Corporation for significant donations to our institutions. We are indebted to Dr. Laurent Moll and Dr. Mark Shand of Compaq Labs for the loan of a PCI Pamette. Lastly, we thank Mr. George Kalokairinos and Mr. Michalis Ligerakis for their invaluable assistance in the design and assembly of the printed circuit board.

References

1. L. Moll, M. Shand, “Systems Performance Measurement on PCI Pamette”, In Proceedings of FCCM `97, pp. 125-133, April, 1997.
2. T. McDermott, P. Ryan, M. Shand, et al., A Wireless LAN Demodulator in a PAMETTE: Design and Experiences, In Proceedings of FCCM `97, pp. 40-45, April, 1997.
3. J. McHenry, P.Dowd, T. Carrozzi, et al., “An FPGA-Based Coprocessor for ATM Firewalls”, In Proceedings of FCCM `97, pp. 30-39, April 1997.
4. D. Wetherall, U. Legedza, and J. Guttag, “Introducing New Internet Services: Why and How”, IEEE Network Magazine, July/August 1998.
5. L. Lehman, S. Garland and D. Tennenhouse, “Active Reliable Multicast”, In IEEE INFOCOM'98, San Francisco, CA, March 1998.
6. B. Schwartz, A. Jackson, W. Strayer, W Zhou, D. Rockwell, and C. Partridge, “Smart Packets for Active Networks”, BBN technologies, In Proceedings of the Second IEEE Conference on Open Architectures and Network Programming (OPENARCH’99), March 1999.
7. D. Alexander, M Shaw, S. Nettles, and J. Smith, “Active Bridging”, Proceedings of the ACM SIGCOMM'97 Conference, Cannes, France, September 1997.
8. M. Hicks, P. Kakkar, J. Moore, C. Gunter and S. Nettles, “PLAN: A Packet Language for Active Networks”, In Proceedings of the International Conference on Functional Programming (ICFP '98), 1998.
9. S. Merugu, S. Bhattacharjee, E. Zegura and K. Calvert, “Bowman: A Node OS for Active Networks”, In Proceedings of IEEE Infocom 2000, March 2000.
10. S. Bhattacharjee, K. Calvert and E. Zegura, “Self-Organizing Wide-Area Network Caches”, In Proceedings of IEEE Infocom'98, San Francisco, CA, March 1998.
11. A. Campbell, H. De Meer, M. Kounavis, K. Miki, J. Vicente, and D. Villela, "A Survey of Programmable Networks", ACM Computer Communications Review, April 1999.
12. A. Lazar, K. Lim, and F. Marconcini, "Realizing a Foundation for Programmability of ATM Networks with the Binding Architecture", IEEE Journal on Selected Areas in Communications (JSAC), Special Issue on Distributed Multimedia Systems, Vol. 14, No. 7, September 1996, pp. 1214-1247.
13. S. Vinoski, “CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments”, IEEE Communications Magazine, Vol. 14, No. 2, Feb 1997.
14. A. Campbell, M. Kounavis, D. Villela, J. Vicente, K. Miki, H. De Meer, and K. Kalaichelvan, "Spawning Networks", IEEE Network Magazine July/August 1999**.**
15. R. Keller, S. Choi, M. Dasen, D. Decasper, G. Fankhauser, B. Plattner, "An Active Router Architecture for Multicast Video Distribution", In IEEE INFOCOM 2000, March 26 - 30, 2000.
16. S. Choi, D. Decasper, J. Dehart, R. Keller J. Lockwood, J. Turner and T. Wolf, "Design of a Flexible Open Platform for High Performance Active Networks", Allerton Conference, September 1999.
17. D. Decasper, Z. Dittia, G. Parulkar, B. Plattner, “Router Plugins - A Software Architecture for Next Generation Routers”, In Proceedings of SIGCOMM’98, September 1998.
18. I.Hadzic, J.M.Smith, W.S. Marcus, "On-the-fly Programmable Hardware for Networks", In Proceedings of Globecom, 1998.
19. W. Marcus, I. Hadzic, A. McAuley, J. Smith, "Protocol Boosters: Applying Programmability to Network Infrastructures", IEEE Communications Magazine, vol. 36, no. 10, pp. 79-83, October 1998
20. P. Stogiannos, A. Dollas, V. Digalakis, “A Configurable Logic Based Architecture for Real-Time Continuous Speech Recognition using Hidden Markov Models”, Journal of VLSI Signal Processing, Kluwer Academic Publishers, vol. 24/(2/3), pp.223-240, March, 2000.
21. P. Newman, G. Minshall, T. Lyon: “IP Switching: ATM Under IP”, IEEE/ACM Transactions on Networking, vol. 6, no. 2, April 1998, pp. 117-129.
22. M. Katevenis, I. Mavroidis, G. Sapountzis, E. Kalyvianaki, I. Mavroidis, and G. Glykopoulos: “Wormhole IP over (Connectionless) ATM”, To appear at IEEE/ACM Transactions on Networking, October 2001.

23. P. Gupta, S. Lin, N. McKeown: “Routing Lookups in Hardware at Memory Access Speeds”, Proceedings of IEEE INFOCOM’98, San Francisco, CA USA, April 1998.
24. A. Dollas, K. Papademetriou, C. Mathioudakis, E. Markatos, M. Katevenis. “Experimental ATM Network Intrerface Performance Evaluation”. Technical Report FORTH-ICS/TR-244, February 1999.
25. T. von Eicken, A. Basu and V. Buch. “Low-Latency Communication Over ATM Networks Using Active Messages”, IEEE Micro, Feb. 1995, pages 46-53.
26. M. Welsh, A. Basu, and T. von Eicken. “ATM and Fast Ethernet Network Interfaces for User-Level Communication”, In Proc. of Third IEEE Symposium on High-Performance Computer Architecture (HPCA-3), San Antonio, February 1997.
27. E. Nygren, S. Garland, and M. Kaashoek. PAN: A high-Performance active network supporting multiple mobile code systems. In Proceedings of the Second IEEE Conference on Open Architectures and Network Programming (OPENARCH’99), March 1999.
28. J. Turner, G. Parulkar, D. Dehart, S. Choi. T. Wolf, B. Platter, R. Keller. Design of a High Performance Active Router. In http://www.arl.wustl.edu/arl/projects/ann/
29. A. Dollas, D. Pnevmatikatos, N. Aslanides, et al., Architecture and Applications of PLATO – E Reconfigurable Active Network Platform, In Proceedings of the 2001 IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM), Rohnert Park, CA, May, 2001.
30. A. Dollas, D. Pnevmatikatos, N. Aslanides, et al., Rapid Prototyping of a Reusable 4x4 Active ATM Switch Core with the PCI Pamette, In Proceedings of the 2001 IEEE International Workshop on Rapid System Prototyping (RSP), pp. 17-23 Monterey, CA, June, 2001.