Rapid Prototyping of a Reusable 4x4 Active ATM Switch Core with the PCI Pamette

Apostolos Dollas, Dionisios Pnevmatikatos*, Nikolaos Aslanides,
Stamatios Kavvadias, Euripides Sotiriades, Kyprianos Papademetriou

Department of Electronic and Computer Engineering
Technical University of Crete
Chania, 73100
Greece

dollas@mhl.tuc.gr

Abstract

PLATO, a new, reconfigurable platform for experimentation with active networks is under development. Due to the large number of factors affecting the final validation of the prototype, we have used the PCI Pamette as a rapid prototyping platform. A 4x4 active ATM switch has been prototyped, together with all the circuits that disassemble, route, and reassemble ATM cells. Several experiments have been conducted with this prototype, substantially speeding up the design process, leading to working subsystems before the final platform is fully debugged, and providing significant insight into the operation of the final system. The design itself was moved from the Pamette to its final operating platform (called PLATO) in two days.

KEYWORDS: Active networks, reconfigurable computing, design core, ATM switch, rapid prototyping

1. Introduction

Active networks[1][2], as opposed to conventional ones, process the header as well as the payload of cells transmitted through them, leading to improved network flexibility, adaptability, security and functionality. The typical flexibility offered by active networks is automatic, rapid protocol deployment, making networks programmable. This in turn, leads to the necessity of processing power on active network nodes. Conventional networks provide routing of packets based on information found in the header of each packet. In addition to the above functions, active networks process the payload of the packet, either for routing purposes (e.g. for dynamic balancing of loads before server systems, detection of Denial of Service attacks), or for improved functionality (e.g. for data extraction).

With very few exceptions, almost all active network projects are software based, due to the (correct) assumption that networks of the future will contain processing power comparable to high-end processors of today. Whereas this approach is excellent for the exploitation of new ideas, it leaves out interesting areas such as real-time payload processing of packets, which may be needed for important applications. Such applications can be the detection of Denial-of-Service (DoS) attacks, real-time load balancing for e-commerce servers, real-time network based speech recognition servers for v-commerce, protocol boosting, etc. To experiment with such applications we have developed PLATO, a reconfigurable platform for ATM networks. PLATO has been implemented and is undergoing tests, whereas the PCI Pamette[3] has been used as a rapid prototyping tool, to test several subsystems and applications. Although network applications with FPGA's have been implemented in the past [4][5], the vast range of real time active network applications that can be developed with last generation, large FPGA's makes this project promising.

The purpose of this paper is to present how a reusable 4x4 active ATM switch design was rapidly prototyped on actual hardware, prior to its operation on PLATO. The

* also at the Institute of Computer Science –FORTH
Irakleio – Crete, Greece

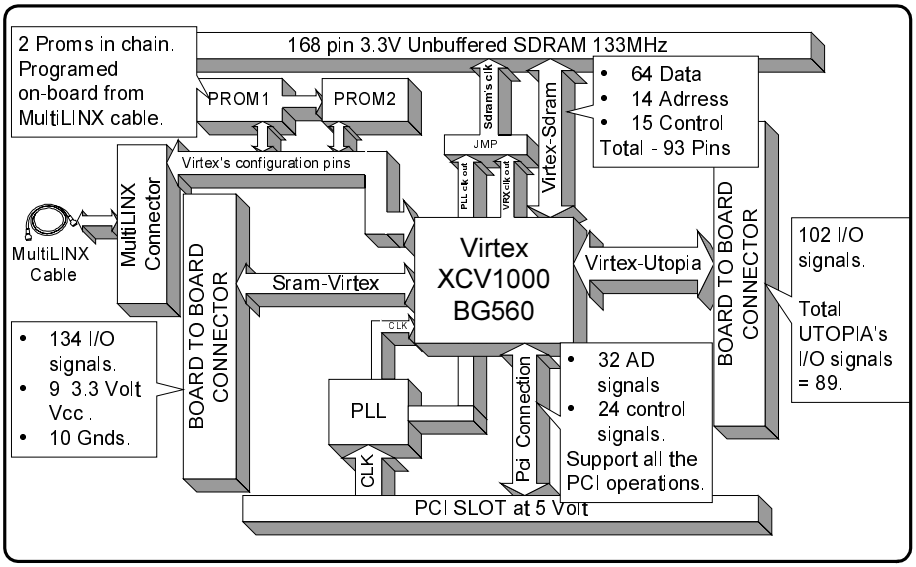

Figure 1. Architecture and Block Diagram of PLATO

design was tested on the reconfigurable platform PCI Pamette. These tests have provided us with a good estimate about the performance of the switch, as well as with well tested, reusable design cores. This paper presents the testing methodology encompassing CAD tool-based simulations, actual runs on the PCI Pamette, and a software-based custom traffic and workload generator, quantifying how each aspect either saved design time or made the testing more complete. The know-how from working with actual hardware in addition to the simulators is also invaluable. Section 2 presents the PLATO architecture. Section 3 presents the prototyping of the 4x4 active ATM switch core with the PCI Pamette. Section 4 presents experimental results, followed by a section on the present status of the project and conclusions.

2. PLATO Architecture and Applications

The architecture, as well as a block diagram of PLATO are shown together, in Figure 1. This is deliberate, because the platform is mainly aimed to be a tool for experimentation and for 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, as well as 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 expansion.

The PLATO platform has a large FPGA, which in addition to the clock generation circuit and programming ports, 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 for SRAM look up tables, which will be used in a real-time speech recognition project [6]
- A PCI bus port for communication with the host.

Although the general topology resembles several existing products, the need for a new hardware design 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.).

Two versions of PLATO are under development, one with Xilinx Virtex XCV 1000 and one with ALTERA 20K400E FPGA's. This way we hope to get results and gain insight on how the architecture of the FPGA's and

the corresponding CAD tools affect the system level performance, as well as have experimental results on the reusability of our cores. As the Xilinx version of the architecture is at a more advanced stage, the remainder of this paper will refer only to this version.

There are several applications for PLATO, but the ones for which design is in advanced stages are:

- Active 4x4 ATM Switch[9]
- Protocol boosting for TCP/IP over ATM[7][8][9], and,
- Content-based routing and rejection of packets, for detection of DoS attacks and server load balancing[9].

In the core of all these applications lies an active 4x4 ATM switch, which (unlike typical ATM switches) does *not* route ATM cells with header-only information, but rather, it disassembles the cells, extracts the payload, interfaces with the application (if any), performs the switching, reassembles, and outputs the cells. The application interface is in hardware, and comprises of two 48-signal buses. This switch does not have the low latency of commercial ATM switches, since it completely disassembles and reassembles ATM packets, but does allow for easy interfacing to the active network applications. Therefore, a reusable and highly robust design core of this switch is needed, and this core was implemented and tested on the PCI Pamette.

3. Prototyping a 4x4 Active ATM Switch Core with the PCI Pamette

3.1 The PCI Pamette

The Compaq Labs' PCI Pamette is a PCI bus compatible board with five interconnected Xilinx FPGA's, DRAM, and expansion ports. One FPGA, which at boot time is loaded from ROM, handles the communication through the PCI bus, and four more FPGA's (in our system XC4044XL), can be used for user designs (User Logic Cell Arrays - ULCAs). The card is memory mapped and allows PCI transactions through software libraries. The general organization of the PCI Pamette is shown in Figure 2.

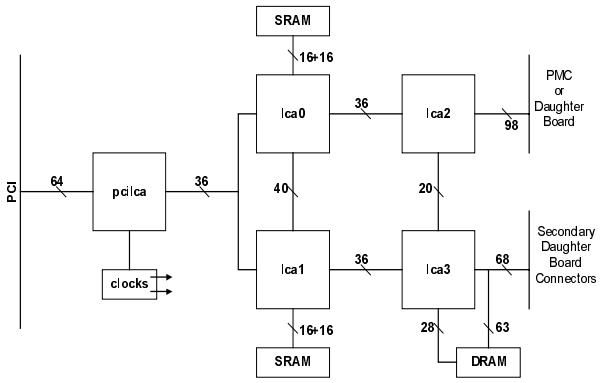

Figure 2 The PCI Pamette

3.2 Architecture of the Switch

The main operation of the switch is to receive ATM cells from a UTOPIA level 2 interface, disassemble them, switch the cells based on their VPI and VCI values and incoming link (PHY), and output the cells at the UTOPIA level 2 interface. There are four incoming and four outgoing links, all operating at 155Mbps. UTOPIA is a standard protocol from the ATM forum that implements the service access point (SAP) between physical and ATM layer, either at UNI or NNI. In the present version, the only buffering available is one cell per incoming link and one cell per outgoing link, whereas the internal datapath is 48 bits wide. The routing algorithm is simply to give higher priority to a lower numbered link. The reason is that the purpose of the switch is mainly to perform processing on the payloads, rather than develop an efficient 4x4 ATM switch, which exists in optimized products for many years.

The switch is either passing the cells to the output or drops them if the output link is occupied. The algorithm of the switch is simple but can give us the percentage of cells per link that are dropped. Based on this figure we can optimize the design in terms of bandwidth and latency. Because audio and video applications can have real time requirements, we wanted to see how the switch is running under bursty traffic. We developed software (described in section 3.5) which supplies ATM cells to the switch based on the Pareto distribution [11], which successfully models bursty interarrival times, observed in TCP Internet traffic. The Pareto distribution is based on two parameters, on the values of which it can model finite or infinite mean and finite or infinite variance. The software also emulates a part of the functionality of the physical interface.

3.3 System Decomposition

The hardware design has three main components. The first is the emulator of the UTOPIA bus (physical layer part). This emulator gets cells from the software via the PCI bus and delivers them to the interface of the switch (ATM layer part of the UTOPIA), which is the second main component. The emulator and the switch interface follow standard rules of the UTOPIA level 2 protocol. The third main component is the switch itself. It disassembles the cells from a specific incoming link, reads their VPI and VCI values, checks a lookup table and calculates the new VPI, VCI and outgoing link values, then reassembles the cells and delivers them to the switch interface. Because we must test the hardware on the PCI Pamette, there is a subsystem that implements the interface of the PCI with our design. The system design is shown in Figure 3.

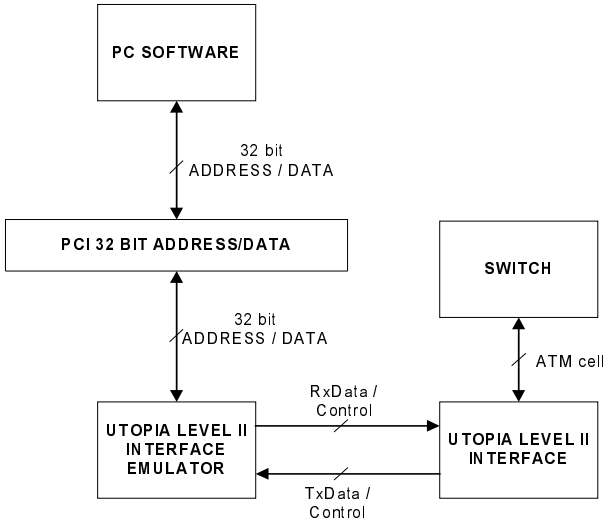

Figure 3 System design

We will now present the operation of the switch. The PC sends through the PCI to the PCI Pamette one ATM cell (53 bytes). The UTOPIA emulator FPGA receives a minimum of one entire cell before setting the proper control signals according to the UTOPIA protocol. The switch interface reads the cell from the emulator FIFO's. When the interface reads a complete cell, it sends the cell to the switch, which starts to process its header immediately. Processing entails the extraction of the VPI and VCI fields, a table look up to determine the new VPI and VCI, and composition of the new header. In the full-scale version of the system, this step will entail processing of the payload as well, which is extracted at the same time as the VPI and VCI fields. After the header processing is done, the switch starts to send the newly composed cell via the interface to the FIFO's of the emulator (these are different FIFO's from the ones at the receiving end). Then the software reads the new cells from the FIFO's and compares the cell data (header and payload) against the C model of the switch, to verify correct operation.

3.4 System partitioning

One design problem we encountered was that the PCI interface should use the PCI clock and the 4x4 switch should have another, slower clock. The need for the slower clock comes from the fact that the design cannot fit in only one of the four FPGAs of the PCI Pamette, and we had to partition it. Because the FPGA's have a limited number of pins to interface to each other (see Figure 2), we use two FPGA's (lca2 and lca3) as connecting lines, and these lines add a lot of delay. Although the design could run in one FPGA with 30 nsec clock period, the delay from lca2 and lca3 is about 20 nanoseconds, so the clock should be no faster than 50 nsec. The PCI Pamette does offer a programmable clock, but this is not synchronized with the PCI clock, so we developed our own clock signal from division of the PCI clock.

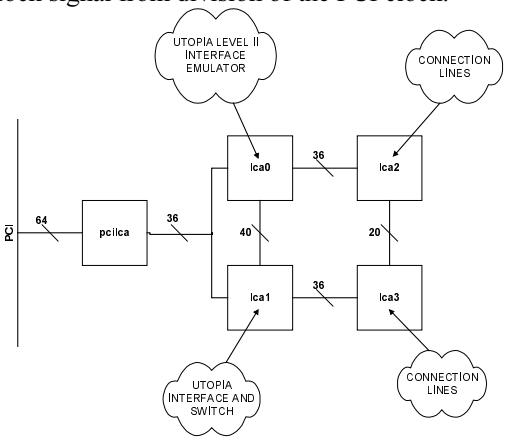

Figure 4 ATM Switch Mapping on the PCI Pamette Resources

Figure 4 shows the final partitioning. We put the emulator in the FPGA called lca0 in the schematic, because it should have an explicit connection with the PCI lca. The UTOPIA level 2 interface and the switch were mapped to the FPGA lca1 because it has the most signals to connect with lca0. This connection has the shortest possible delay. Also, the switch and the UTOPIA interface need more signals than are available between FPGA's. Therefore, they are in the same FPGA, so that they can interface with internal wires. One part of the design that is in lca0 runs at the PCI clock and the rest runs at half the frequency.

3.5 Software and I/O Considerations

In the initial stages of the project we developed a simulator for the switch in C. A traffic generator was also developed in C++, in order to supply ATM cells with varying VPI/VCI and predefined distributions to the simulator. This last software framework was easily integrated with our code to handle PCI Pamette transactions, providing us with a very powerful testing tool for the 4x4 ATM switch, as well as for active applications running alongside. The traffic generator allows us to model incoming data flows with selected distributions. If the distributions are parameterized on the maximum or average bandwidth, we can partition the total bandwidth of each physical link at will or randomly.

In particular, each flow (data coming in a specific incoming link with specific VPI/VCI values) is checked for cell arrivals in the current software cycle. The cell arrivals are based on a selected distribution and the software accumulates per link cells that have arrived in the last cycle. One cycle corresponds to the time of a subsequent cell's appearance on a link, which depends on link's bandwidth. Those cells are sent to the PCI Pamette, one per link, in round robin fashion. After one cell has been sent to all links, the cells transmitted by the switch must be gathered, because the transmit FIFOs of the UTOPIA emulator are only 32 16-bit words deep. After the transmitted cells are gathered, they are written to a (per link) file. When all cells accumulated on the four links are sent, received or deleted, the software increments the program's notion of time (the current cycle) and starts all over again.

We chose to simplify the hardware by doing all the control in software. The hardware only supplies a set of addresses for reading/writing cells in/from the UTOPIA emulator FIFOs and reading the FIFO associated counters. Thus we operate the PCI Pamette as the target of transactions initiated by software. This mode of operation, needs on average 6 cycles for a write and 12 cycles for a read operation (on PCI clock), because of the way the PCI interface handles transactions. Assuming 50% reads and 50% writes, the total attainable PCI bandwidth in this mode is ~118.5 Mbps (~59 Mbps in each direction), which can saturate a single incoming link[1].

Even though the transmit direction FIFOs have only 32 16-bit positions (one ATM cell needs 27), we can test conflict handling on the switch. If the second cell is not dropped, its 5 first 16-bit words will be collected by software and the cell can be thus identified.

[1] The reason is that the switch and its UTOPIA interface is run with half the PCI clock frequency on Pamette, while in the actual platform we can run it with a clock up to 50 MHz - which is three times more. Thus, if the switch can handle 59 Mbps from a single incoming link on Pamette, it will be able to handle up to 3*59=177 Mbps in PLATO.

4. Experimental Results

4.1 Design Cycle Speedup with Rapid Prototyping

One of the main reasons to develop a prototype is the better, and in many cases exhaustive, testing of the design vs. simulation alone. Although the experimental 4x4 ATM switch only performs basic operations on a number of VCI and VPI's, which is much smaller than that of a commercial switch, the total number of states to be tested is quite large. Running a simulation on a 550MHz Pentium III processor with 256Mbytes of RAM takes 7 minutes of CPU time per ATM cell simulation. The above time does *not* include the setup of the simulation, which is comparable if not larger in this case (preparation of vector inputs, correct outputs, etc.). The simulation setup time could be reduced with specialized CAD tools such as ModelSim, which, however, we do not have available. In actual simulations, the total time to setup and run simulations for the processing of 5 ATM cells took roughly 5 hours. This time is an upper bound and could be reduced, however, it is indicative of the CAD tools-only approach. The complete elimination of CAD tool-based simulation is not possible or even desirable, of course, because the design which is prototyped should be in fairly stable form and because internal signals are not easily accessible with prototyping.

The experiments which needed to be performed, even on our small prototype, to exhaustively test it are:

1. Transmission of an ATM cell for every possible VPI,VCI and PHY (1024 cases)
2. Transmission of an ATM cell for every input and output combination (16 cases)
3. Multiple cell transmission (chosen to be 10 cells) from every input to every output (16 cases).
4. Collisions of packets in all possible combinations every 2, 3 and 4 inputs (44 cases)
5. Experiments for different distributions (e.g. Gaussian, Pareto) for experiment 3.

With the traffic generator and the other tools that we developed, the PCI Pamette can run the ATM packet switching experiment in 5 sec (yielding a design process speedup of 84). Even more impressive is the case for experiments 3 and 5 (above), where the speedup of the design process is superlinear. This speedup is readily explainable as follows: each experiment on the PCI Pamette takes a (fixed) amount of time to configure the system and a variable *but small* amount of time to run the experiment. Whereas in the case of simulation a train of 10 ATM cells will take ten times more CPU time to run, in the case of the PCI Pamette it will take ten times more execution cycles, which in this case is a few hundred cycles (still, considerably less than the number of cycles to set up the experiment). The above experiments involve a total of 2044 ATM cells, which at 7 minutes per cell would require 14,308 minutes of the CPU, not counting experiment setup and result evaluation time, resulting in over one man-month level of effort (these are not long simulations that can run overnight). By comparison, the 1024 cases of experiment #1 (which would require 15 man-days) were run in one day.

The combination of the traffic generator and the experiment execution on the PCI Pamette greatly reduces the setup time for each experiment, and allows for the testing of many more cases than simulation alone would allow for. The time to develop the traffic generator as well as all the experiments on the PCI Pamette is roughly 2 man-months, and in addition to running all the experiments faster than simulation would, resulted in a highly flexible, reusable environment for future experimentation.

4.2 Experience Gained from the Pamette

In addition to the design process benefits, described in Section 4.1, major implementation experience was also gathered. Prototyping reintroduced functional errors that were previously found and corrected during the VHDL Modeling - Synthesis – Simulation iterative process. In particular, during simulation it was observed that, ATM cells directed towards outgoing link 0 where switched correctly, while cells directed to other links where erroneously rejected. The problem was reintroduced due to incorrect timing of internal control signals in the implemented design. Additional experience was gained from the Pamette mapping of the design, which forced the prototype to use two clocks. In interfacing the two parts using different clocks, we used FIFOs clocked with the PCI clock, and synchronized slow control signals with intermediate flip-flops. We should also underline the fact that without the use of implementation constraints, the design would not meet the timing requirements for correct operation at all times. This was the most time consuming part of the prototyping process because several full implementations of the design were required. In addition, placement constrains were used to achieve the target clock frequency of the PCI part of the design. All these particular problems underline the differences among functional and timing simulations, vs. hardware implemented on an actual platform. On the other hand, the design was verified on actual hardware and considerable debugging experience was gained, which aided its port to the PLATO system.

The design of the emulator, the 4x4 ATM switch and the UTOPIA level 2 interface took three personmonths, and the port to the PCI Pamette took twenty persondays.

4.3 Transfer of the Design to the PLATO System

Following the prototype of the switch with the PCI Pamette, we proceeded to transfer the design to the PLATO system, which in the mean time was fabricated. The Virtex-based PLATO platform has been operating at the Technical University of Crete (TUC). The 560 pin BGA package necessitated an 8-layer PCB design, which was performed at TUC. The board was fabricated by INTRACOM SA, and assembly of the board took place at the Institute of Computer Science (ICS)–FORTH. A PCI interface has been designed at the Technological Educational Institute (TEI).

The Virtex-based PLATO board underwent tests for two weeks to confirm proper download capability and operation of the FPGA, operation at several speeds (up to 70 MHz in tests), and proper I/O (e.g., no short circuits). Following the testing of the board, the 4x4 ATM switch design was transferred to PLATO. *The trasfer of the 4x4 ATM switch design, together with the UTOPIA interfaces and the UCF file to determine the pinout of the PLATO Virtex was successfully done in two days, by two engineers (four person-days*). This rapid transfer demonstrates how well the Pamette allowed for prototyping of the 4x4 ATM switch while the PLATO was still designed. Within the same two days, the Pamette was used for another purpose too: as the PCI interface of PLATO was not operational, the Pamette was used for I/O with the PLATO. In other words, the 4x4 switch moved from the Pamette to PLATO, whereas the Pamette itself became the front end system to test successfully the PLATO board. It is noteworthy that in these tests, the UTOPIA level 2 interface of the PLATO system was running on the Virtex, whereas the Pamette took data from the PCI bus, and reformatted it to the UTOPIA level 2 standard for transfer to the Virtex. The design was conservatively run at 16.5 MHz (half of the PCI clock) to

verify correct operation, whereas tests at full speed have not been performed yet. The resource allocation in this test are 1692 Virtex slices for the 4x4 switch, including the UTOPIA level 2 interface (13% utilization), and, 909 CLB's of a single Xilinx XC4044XL on the Pamette for the modeling of the UTOPIA level 2 interface. The 13% resource utilization of the Virtex is very encouraging, because substantial circuitry can be added for payload processing.

In terms of design modifications for the 4x4 switch to fit on the Virtex part, the only two modifications that proved necessary were the pin assignment changes (UCF files) and the remapping of the FIFO's from XC4000XL series technology to Virtex technology. The effort for these modifications is included in the two days (four persondays) to port the design on the Virtex part.

5. Present Status and Conclusions

An additional design, based on the ALTERA 20K400 is in the advanced design stages at TUC and ICS. The extensive testing of the reusable ATM switch core with the PCI Pamette has allowed for separate debugging of the PLATO board itself vs. the applications running on it. The software interface and traffic generator we have developed are useful in the PLATO system as well, and as such will be fully utilized.

The next step is to have the PLATO system fully working and at full speed, and begin experimentation on actual ATM packets for applications such as Wormhole IP over ATM protocol boosting, detection of Denial of Service attacks, and content-based routing.

In conclusion, despite the overhead to partition the design onto the PCI Pamette, the approach proved very successful and in the long run it saved considerable development time.

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 99ED 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, and for generous allocation of their time to transfer their PCI Pamette know-how to us. Lastly, we thank Mr. Sotirios Zogopoulos and Mr. George Kalokairinos and Mr. Michalis Ligerakis for their invaluable assistance in the design of the printed circuit board.

References

[1] 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.

[2] D. Wetherall, U. Legedza, and J. Guttag, "Introducing New Internet Services: Why and How", *IEEE Network Magazine*, July/August 1998.

[3] L. Moll, M. Shand, "*Systems Performance Measurement on PCI Pamette"*, In *Proceedings of FCCM `97*, pp. 125-133, April, 1997.

[4] T. McDermott, P. Ryan, Mshand, *et al.,* A Wireless LAN Demodulator in a PAMETTE: Design and Experiences, In *Proceedings of FCCM `97*, pp. 40-45, April, 1997.

[5] J. McHenry, P.Dowd, T. Carrozzi, *et al*., "An FPGA-Based Coprocessor for ATM Firewalls", Proceedings, FCCM `97, Napa, California, pp. 30-39, April, 1997.

[6] 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.

[7] M. Katevenis, I. Mavroidis, I. Mavroidis, and G. Glykopoulos, "Wormhole IP over (Connectionless) ATM", Institute of Computer Science (ICS), FORTH. http://archvlsi.ics.forth.gr/wormholeIP

[8] 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.

[9] A. Dollas, D. Pnevmatikatos, *et al*., Architecture and Applications of PLATO, a Reconfigurable Active Network Platform, IEEE FCCM 2001 Symposium, (accepted, to appear).

[10] Ohtomo, Y; Yasuda, S; Nogawa*, et al.,* A 40-Gb/s 8x8 ATM switch LSI using 0.25-mu m CMOS/SIMOX, IEICE TRANSACTIONS ON ELECTRONICS Vol. E81C, No. 5, pgs. 737-745, May 1998.

[11] B. Arnold, Pareto Distribution, International Cooperative Publishing House, Baltimore, MD, 1983.