FORTH-ICS / TR - 258 June 1999

Hardware Implementation of a Routing Filter to support Wormhole IP over ATM

Iakovos Mavroidis

ABSTRACT

Increasing internet traffic needs require fast routing which implies using hardware instead of software. Wormhole Routing was used in multiprocessor computers to provide fast communication among CPU's. In order to implement Wormhole IP over ATM we need *Routing Filters* placed between ATM subnetworks, as described in [1]. These Filters will receive IP packets segmented into ATM cells (according to AAL5) from one ATM subnetwork, and forward them after a VP/VC translation to the next subnetwork. We have designed and implemented such a Filter. In this report we describe its organization and the datapath that we simulated in Verilog. For a thorough description of its functionality please refer to [1].

Hardware Implementation of a Routing Filter to support Wormhole IP over ATM

Iakovos Mavroidis [1]

Computer Architecture and VLSI Systems Division (CARV)
Institute of Computer Science (ICS)
Foundation for Research & Technology - Hellas (FORTH)
P.O. Box 1385, Science and Technology Park of Crete,
Heraklion, Crete, GR-711-10 GREECE
Tel: +30 81 391662, Fax: +30 81 391661
email: jacob@ics.forth.gr

Technical Report FORTH-ICS/TR-258 June 1999

ABSTRACT

Increasing internet traffic needs require fast routing which implies using hardware instead of software. Wormhole Routing was used in multiprocessor computers to provide fast communication among CPU's. In order to implement Wormhole IP over ATM we need *Routing Filters* placed between ATM subnetworks, as described in [1]. These Filters will receive IP packets segmented into ATM cells (according to AAL5) from one ATM subnetwork, and forward them after a VP/VC translation to the next subnetwork. We have designed and implemented such a Filter. In this report we describe its organization and the datapath that we simulated in Verilog. For a thorough description of its functionality please refer to [1].

This text is available in Postscript form, by anonymous ftp, from server "ftp.ics.forth.gr".
Directory: "tech-reports/1999".
Files: "1999.TR258.RoutingFilterCore.README",
"1999.TR258.RoutingFilterCore.ps.gz"

[1] The author is also with the Computer Science Department, University of Crete, Greece.

Contents

1 Introduction

In *Wormhole Routing* information is split into small parts, called "flits" which are then transmitted in order. However, routing decisions are made only for the first flit, which carries the destination address of the whole group, and all the rest follow along the same route. "Wormhole IP over ATM" uses this same idea to route IP packets over ATM subnetworks. Routing Filters are placed between the ATM subnetworks to take routing decisions by making VP/VC translations. This process is best described in [1]. We have designed and simulated in Verilog such a Filter.

We believe that the design of the Filter is useful because it will provide a cheap and quick router, since it is implemented in hardware while major parts of other routers are implemented in software. We exploit today's fast hardware which permits us to examine and route every cell of the IP packets.

The organization of the rest of this report is as follows.

In section 2 we give a brief overview of the functionality and components of the Filter.

In section 3 we describe the organization of the data structures used by the Filter.

In section 4 we describe the time scheduling of the memory accesses of the Filter.

In section 5 we describe the cell buffers of the Filter and their overflow probability.

In section 6 we describe the datapath simulated in Verilog.

In section 7 we propose some potential future improvements to the Filter.

In section 8 we conclude.

2 Filter Overview

A Routing Filter is put between two conventional ATM subnetworks to route IP traffic. It receives from its input link native ATM cells or IP cells coming from other Routing Filters or source hosts of the attached subnetwork. These two kinds of traffic can be distinguished, since IP traffic is the one received through the permanent VP's that connect the Routing Filters and through their VC's that are dedicated to IP. The Filter is transparent to ATM traffic which is forwarded directly to the output link, while the IP traffic is first processed. The Filter should be able to distinguish IP packets, take IP routing decisions to decide the next Filter or destined host that each packet should be routed to, and forward each packet to its output link using the corresponding permanent VP for the next-hop Filter or destined host of the attached subnetwork. To achieve bidirectional functionality for the Filter its ports have to be bidirectional and be able to act as both input and output links. The Filter has two bidirectional ports and it transmits every cell received from one port to the other and vice versa.

2.1 Filter Operations

Semipermanent VP/VC connections are used to transmit IP traffic. Certain input and output VP/VC's are dedicated solely to IP traffic. Hence, ATM and IP incoming cells are distinguished by their VP/VC identifiers. The first cell of an IP packet is routed creating a new VP/VC connection, while the rest are sent through this connection. The last cell of an IP packet is recognized by a 2-bit field of its ATM header, called PTI. The first incoming cell having the same input VP/VC as the last cell of an incoming IP packet, has to be the first cell of the next IP packet of this VP/VC. We use VP switching for the IP traffic, hence the VP of the outgoing cell determines its route. On every output VP of the IP traffic there are 32 consecutive VC's reserved for IP traffic.

Certain VP/VC's of incoming traffic are dedicated to IP traffic. For each VP of the incoming traffic the maximum number of VC's for IP traffic is taken to be 32 of the 64K available VC's, since this number is small enough not to affect the ATM traffic and large enough to support the IP traffic. In the Connection Table we store 32 entries for each VP, which correspond to the possible active connections established by the IP packets for this VP. Every IP packet creates a VP/VC connection in the Connection Table of the Filter. The Connection Table also holds the VP/VC's of the incoming cells that belong to IP traffic. Hence we first examine the Connection Table and the VP/VC of the incoming cell in order to decide whether the cell belongs to IP traffic or not. Specifically, we read the corresponding entry of the VP of the incoming cell by indexing with the 5 LS bits of the incoming cell's VC. The entry includes a 11-bit field, called *VCin*, and a 2-bit field, called *State*. The *State* can be either *ATM*, *Drop*, *Active IP*, or *Inactive IP*. If the 11 MS bits of the VC of the incoming cell match the *VCin* field, and the *State* is not *ATM* the cell belongs to IP traffic, otherwise it belongs to ATM traffic. If the cell belongs to IP traffic and the *State* is *Inactive IP* the cell is the first cell of an IP packet. The first cell of an IP packet is routed to the output VP that the Routing Table specifies when indexed with the cell's IP destination. The aforementioned two fields of the Connection Table entries and the Routing Table can be modified by the Manager, in order to create the semi-permanent VP/VC's for IP traffic. The VC of the output VP is determined by the Free List which stores the unoccupied VC's of every output VP belonging to IP traffic.

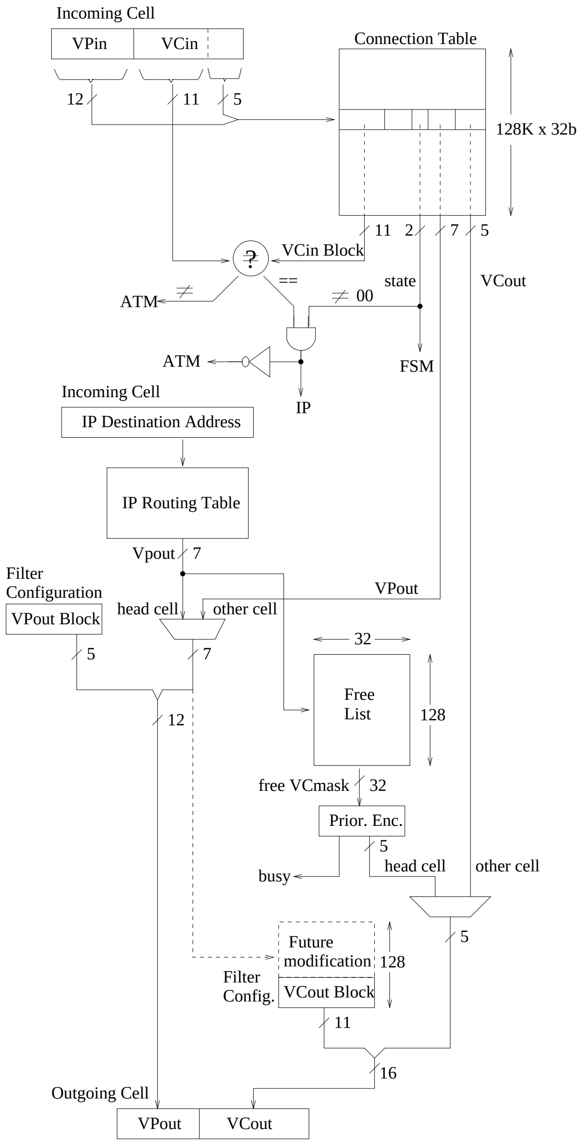

Figure 1: VP/VC translation of an incoming cell.

The Filter operations can be distinguished according to the type of the incoming cell. Specifically, if the input cell is the first cell of an IP packet, then the major operations include reception of the input cell, selection of the output VP from the Routing Table, selection of a free VC for the output VP from the Free List, modification of the entry in the Connection Table, and transmission of the cell. If all the VC's of the output VP are occupied then the *State* of the connection entry in the Connection Table is set to *Drop* and all the cells of the packet are dropped. If the input cell is not the first cell of an IP packet, the Connection Table provides the output VP/VC to use for the cell transmission. Finally, if the cell belongs to ATM traffic, it is sent unmodified (i.e. using the same VP/VC value) to the output port.

A block diagram that depicts the VP/VC translation of an incoming cell can be seen in Fig. 1. For simplicity, the Routing Table determines only the 7 LS bits of the output VP, where the other 5 bits are statically determined by the Manager, and the Free List determines only the 5 LS bits of the output VC where the other 11 MS bits of the VC are also determined by the Manager. Hence, the Free List holds 32 VC's for each of the 128 potential output VP's that may be used for IP traffic. The Routing Table and the Connection Table are held in 2 SIMMs, while the Free List is kept in the internal SRAM of the FPGA, since it is small enough to fit.

2.2 Filter Components

We have created the PCB, and implemented on FPGA the core functions of the Filter. The main components of the board are :

- **Two Optical Transceiver chips** which convert the optical signal to electrical signal and vice versa.
- **Two TNETA chips** which transform Sonet to Utopia and vice versa.
- **Two DRAM SIMMs** of 32 Mbytes each, which hold the Connection Table and the Routing Table, as described below.
- **The FPGA** which implements the core functions of our Filter.
- **Two clocks** at 20 and 40 Mhz frequency, output by a PLL. The former is used by every chip in the PCB while the latter is used only by the FPGA.

The communication between these components is depicted in Figure 2.

The board is attached to a PC, communicating through its parallel port. The major parts of the FPGA are the following :

- **The TNETA interfaces** that receive cells at, and transmit cells from the Filter's internal buffers.
- **The Routing Table Lookups** which provide the output VP that corresponds to the destination of an IP packet.
- **The Free List** that holds the output VC's dedicated to IP traffic and not yet occupied by any IP packet, for every output VP.
- **The Manipulation of the Connection Table** that holds the VP/VC's of the incoming cells that belong to IP traffic, and the connections established by IP packets.
- **The Manager** which configures and reads the data structures of the Filter.

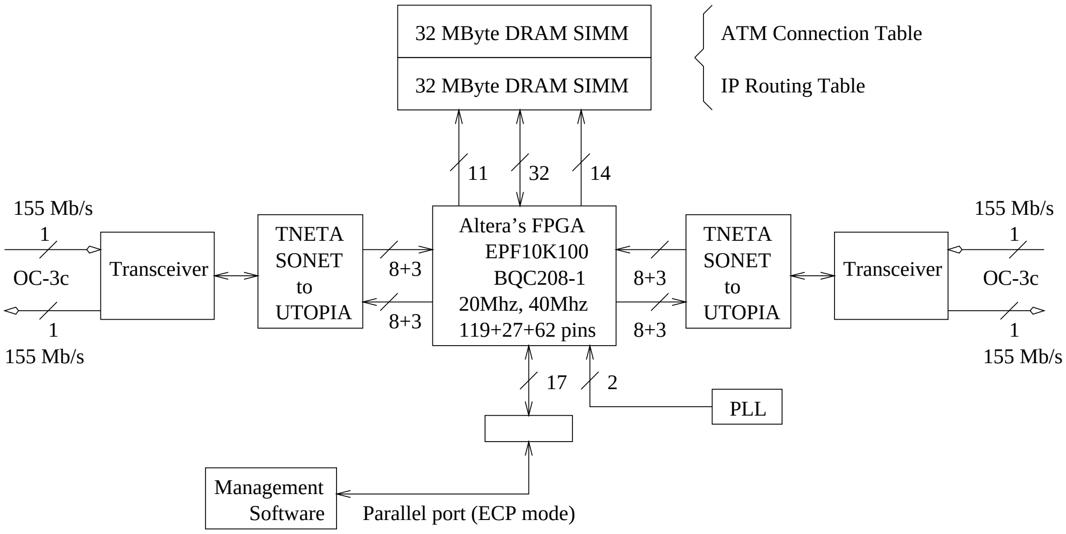

Figure 2: Block diagram of hardware prototype.

3 Data Structures Organization

The Routing decisions depend on the Routing Table content. The Connection Table holds the active connections of the IP packets. The output VC's dedicated to IP traffic are held in the Free List. We briefly describe each one of these data structures.

3.1 Routing Table

The Routing Table is organized in two levels as described in [2]. The first level is held in one SIMM and the second level is held in the first half of the other SIMM.

3.2 Connection Table

The Connection table is in the second half of one memory SIMM, and has size 4Mbits (4K VP's of 32 VC's each with 32 bits per VC). There are two Connection Tables (one for each direction) with 32 32-bit entries for each of the 4K VP's. Fig. 3 shows an entry of the Connection Table, and how its *State* field is encoded.

The Connection Table is indexed using a 17-bit index consisting of the 12-bit VP and the 5 LS bits of the VC of the incoming cell. If the 11 MS bits of the VC of the incoming cell match the *VCin* field, the entry is valid, otherwise the cell belongs to ATM traffic. If the entry is valid the *State* field is set to :

- **ATM** if the cell belongs to ATM traffic.
- **Inactive IP** if the cell is the first of an IP packet and hence no connection has been established yet.

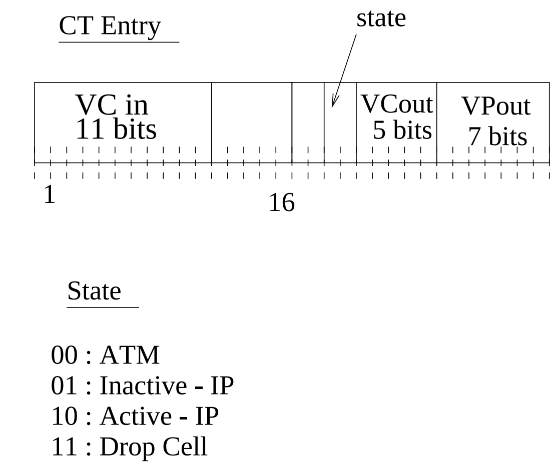

Figure 3: Connection Table Entry.

- **Active IP** if the cell belongs to an IP packet that has established an ATM connection. Its output VP/VC identifier is recorded in the *VCout* and *VPout* fields of the Connection Table entry.
- **Drop** if the cell is dropped.

The *State* field is modified according to the FSM in Fig. 4, if the incoming cell belongs to an IP packet.

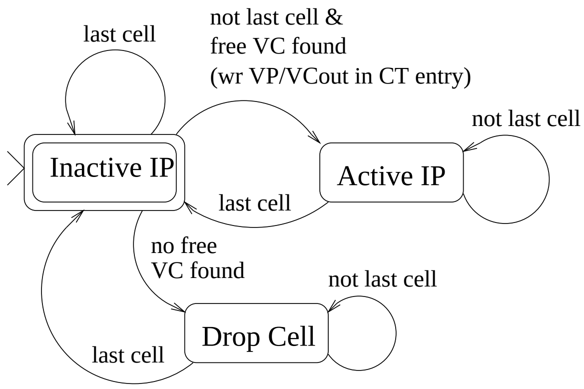

Figure 4: State field FSM.

3.3 Free List

The Free List holds one bit for each output VP/VC identifier dedicated to IP traffic. This bit specifies whether the output VP/VC is currently occupied by an IP packet or not. Since the supported maximum number of output VP's for IP traffic is 128 and every VP can hold 32 VC's for IP traffic, we need a 128x32 bitmap. When the first cell of an IP packet is transmitted, we search the Free List to find a free VC of the VP determined by the Routing Table. If a free VC is found, it is occupied, otherwise the packet is dropped.

When the last cell of an IP packet is transmitted, its output VP/VC is freed and returned to the Free List.

The Manager can occupy or free a VP/VC of the Free List. Thus, the Manager can deactivate or activate the VC's of an output VP of IP traffic.

4 Time scheduling of Memory Accesses

A cell time is the minimum time required to receive or transmit a cell, which is 53 cycles of the slow clock (20 Mhz). During this period many shared resources are accessed, for instance the external DRAM, and the internal SRAM or logic cells of the FPGA. Determining statically the time scheduling of the DRAM accesses, leads implicitly to a scheduler for every shared resource. During each cell time, the FPGA reads twice (once for each flow of data since the Filter is bidirectional) both levels of the Routing Table, and also reads and writes twice (using a read-modify-write operation supported by the DRAM) to the Connection Table. The remaining cell time is used by the Manager for reading or writing and the DRAM for refreshing. These operations sum up to 7 memory accesses per cell time.

The DRAM has a 32-bit data bus for input and output. The time needed for a read operation of the DRAM is equal to 4.5 clock cycles, and the time needed to write the 32-bit data back to the same address of the DRAM (write part of a read-modify-write) is equal to 2.5 clock cycles. In order to implement the algorithm for Fast Routing Lookups, as described in [2], we need to perform 2 read operations (one for each level of the Routing Table) as well as a read-modify-write operation (for the Connection Table) during a cell time. The time scheduling of these memory accesses is depicted in Figure 5.

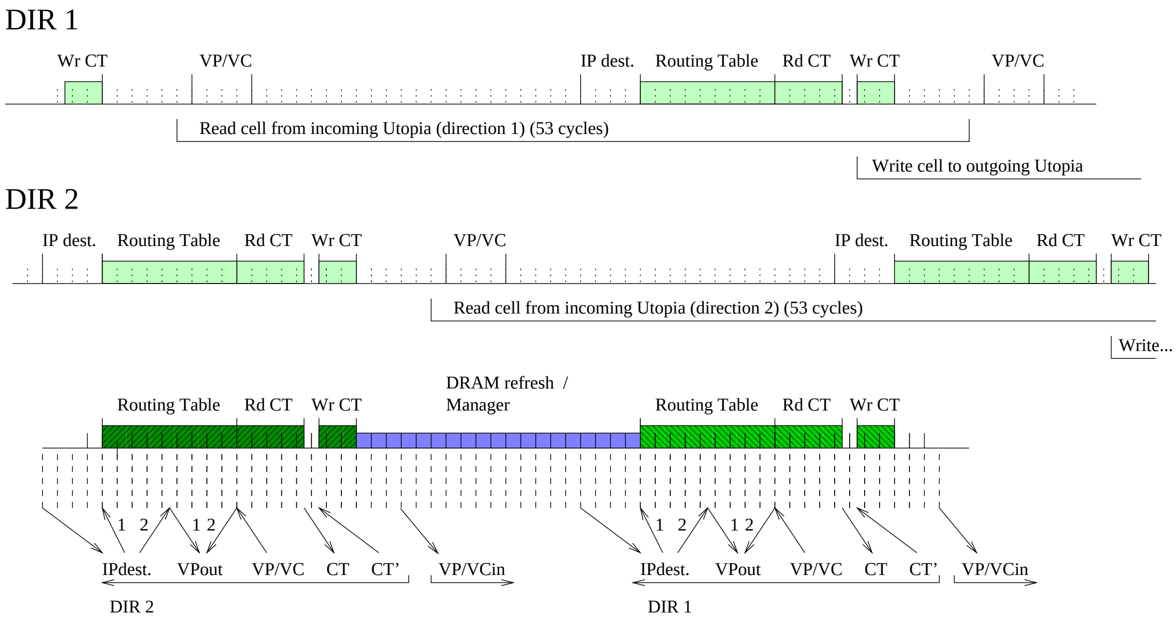

Figure 5: Time Scheduling of Memory Accesses.

In the upper part of the above figure we see the operations performed during a cell time for one of the two directions of data flow through our Filter. First, the VP/VC of the incoming cell is received. Then the IP destination is extracted from the 4 bytes 27 through 30 of the incoming cell (assuming that this is the first cell of an IP packet) and is used for the lookup to the Routing Table. Finally, the read-modify-write operation to the Connection Table occurs. The cycles when the DRAM is accessed are shaded in the figure.

In the middle part we see the time scheduling of the operations during a cell time for the other direction of data flow through the Filter. As we see the DRAM accesses performed for the two directions do not overlap. In the bottom part we see the DRAM accesses performed by either direction, as well as the cycles reserved for the refresh and the Manager operations. The small arrows show the cycles that the FPGA accesses the memory or receives data from it. The numbers on the arrows refer to the level of the Routing Table that is accessed. Some arrows are in the middle of clock cycles because the FPGA has a clock at 40Mhz and hence these are positive edges of this clock. The edges of this clock are not depicted in the Figure for simplicity.

To simplify the control logic, the memory operations that we just mentioned, are scheduled to occur during *every* cell time, even though some times their results may be useless (e.g. when a cell from pure ATM traffic is received, or when the cell is not the first one of the cells comprising an IP packet). Of course, special care has to be taken to prevent harmful writes to the Connection Table.

To further simplify the control logic, the cell times start and end at specified points in time. This specified timing of the cell times by the FPGA, contradicts with the asynchronous nature of cell arrivals. The TNETA interface transmits cells to the FPGA in an asynchronous fashion, as they are received. Thus, when an incoming cell arrives in the middle of an FPGA cell time, it is buffered and its processing is deferred to the next cell time. In order to avoid overflow of this buffer the TNETA interface stops receiving incoming cells until the FPGA asserts a signal to inform it that it has started processing the current cell.

5 Cell buffers of the FPGA and TNETA chip

A TNETA chip receives cells from or sends cells to each of the two bidirectional ports of our Filter, as can be seen in Figure 2. This chip has two 2-cell buffers, one for the incoming cells from the Transceiver and the other for the outgoing cells of the other direction received from the FPGA. The TNETA chip can not always receive cells at the same speed it transmits them. That is because the content of the Sonet frames, that determine the output throughput of the TNETA chip, is not solely determined by the number of the cells. Hence, while the FPGA transmits cells to the TNETA chip at the speed it receives them, when there is heavy traffic the TNETA's cell buffer that holds the cells it receives from the FPGA sometimes may overflow. The TNETA chip activates the signal TXAF when the cell buffer has less than 6 bytes of free space left. Thus, in order to prevent overflow of the TNETA's transmit buffer the FPGA stops sending bytes to the TNETA when the TXAF is activated. This implies that if the TXAF signal is active long enough, the internal 2-cell buffer of the FPGA may overflow since the incoming cells are not transmitted to the TNETA chip. Also if the TXAF signal is activated, when the current incoming cell is ready for transmission the previous outgoing cell may not have been transmitted yet and the FPGA will have to hold on the transmission of the incoming cell. Thus when the TXAF is activated the start of the transmission of the incoming cell does not happen in a specified cycle in the above time scheduling of Figure 5. In order to prevent the overflow of the internal cell buffer of the FPGA, the FPGA activates the RRE signal of the receive interface of the other TNETA chip which disables the reception of the cells from the TNETA chip. When this signal is activated the TNETA chip does not transmit to the FPGA the cells it receives from the Transceiver. This may result to the overflow of its 2-cell buffer and hence to the loss of incoming cells. In this case and when AAL5 is used the whole corrupted AAL5 frame and included IP packets will be considered lost, since the error checking of its CRC will result to its drop. Figure 6 depicts the percentage of the AAL5 frame loss when raw ATM traffic is passed through the Filter.

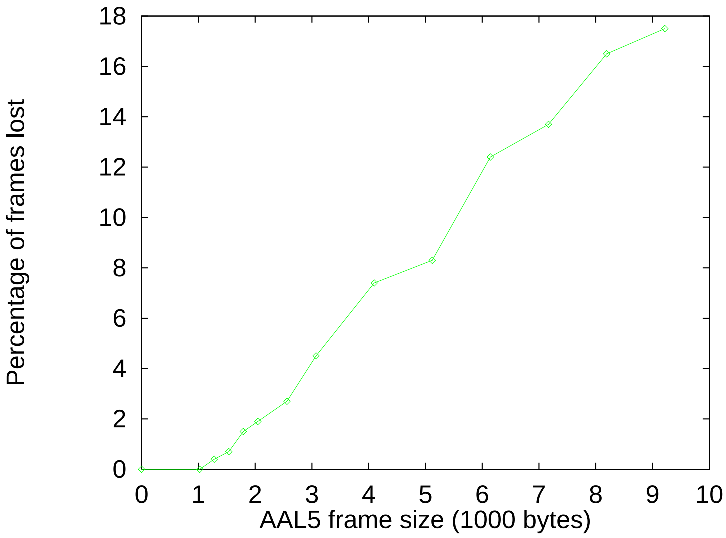

Figure 6: AAL5 frame loss probability.

6 Simulation in Verilog

For our on-FPGA Filter implementation we used ALTERA's tools and FPGAs. The FPGA model that we used is the FLEX10K100BQC208-1 since this is large enough to fit our design and potential future work. This FPGA has 208 pins, 4992 logic cells, and 12 EABs, where an EAB consists of 256x8 bits. We used 148 pins, 6 EABs and about 40 percent of the available logic. Our implementation can sustain a slow clock at 60 Mhz and a quick clock at 65 Mhz, exceeding by far the frequency of the two input clocks.

6.1 Datapath

The Datapath of the FPGA can be separated in 4 parts, namely the Routing Table Lookups, the Manager with Parallel port interface, the DRAM interface, and the Filter core including the aforementioned Filter operations. The implementation of the Routing Table Lookups is described thoroughly in [3]. In this section we will describe the datapath of the Filter core. A simplified datapath is illustrated in Fig. 7. Notice that the four major parts it consists of, are the Input datapath, the Free List datapath, the DRAM/Manager communication datapath, and the Output datapath, which are described below.

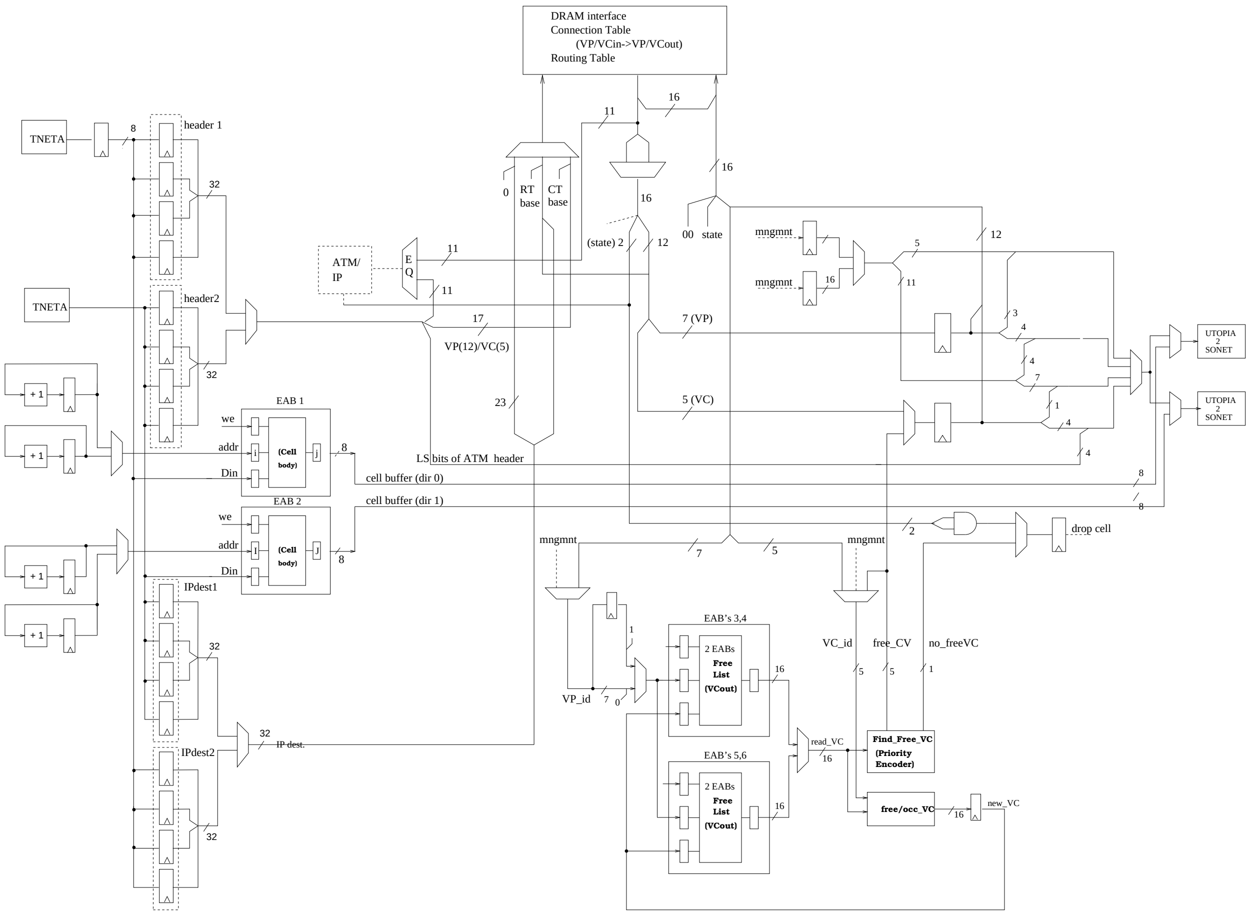

Figure 7: Filter core datapath.

6.1.1 Input Datapath

This part of the datapath includes the receiving interface of the Filter with the TNETA chips, a buffer for the incoming cell, and registers that hold the VP/VC and the IP destination of the incoming cell if it is the first cell of an IP packet.

The TNETA interface includes a 8-bit bus for receiving data, a 8-bit bus for transmitting data and 6 signals for controlling communication. A full datasheet of the TNETA chip is included in [4]. The first 4 bytes of the incoming cell contain its VP/VC, and the four bytes 27 through 30 contain the IP destination, if this is the first cell of an IP packet. The incoming cells are buffered in two EABs, one for each direction. Each EAB holds 2 cells.

Fig. 8 shows the datapath of the Filter input.

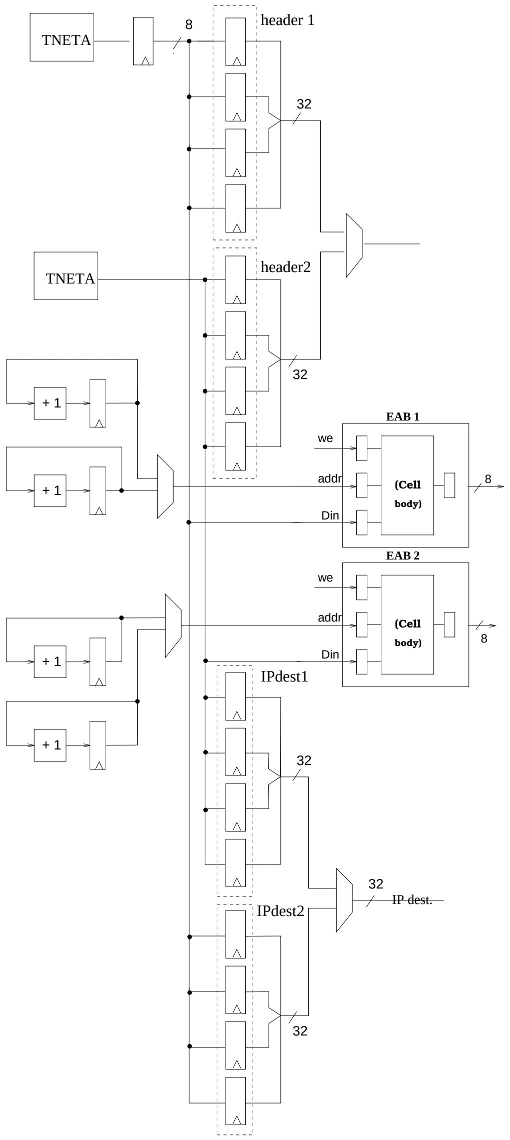

Figure 8: Filter Input Datapath.

Registers *header1* and *header2* hold the ATM headers of the incoming cells (one for each direction), which contain the VP and VC identifiers. The 2 EABs store the incoming cells. Registers *IPdest1* and *IPdest2* hold the IP destinations of the incoming cells for the two directions, which are valid only if the corresponding cell is the first of an IP packet. Using the quick clock and the 2 EABs we have created a 2-port internal memory for each direction at the frequency of the slow clock. Hence, the incoming cell can be buffered while another cell is read for transmission.

6.1.2 Free List Datapath

The Free List defines whether a VC of an output VP, belonging to IP traffic, is free or occupied. The operations supported by this datapath are finding out a free VC and occupying or setting free a VC of an output VP. Fig. 9 shows this datapath.

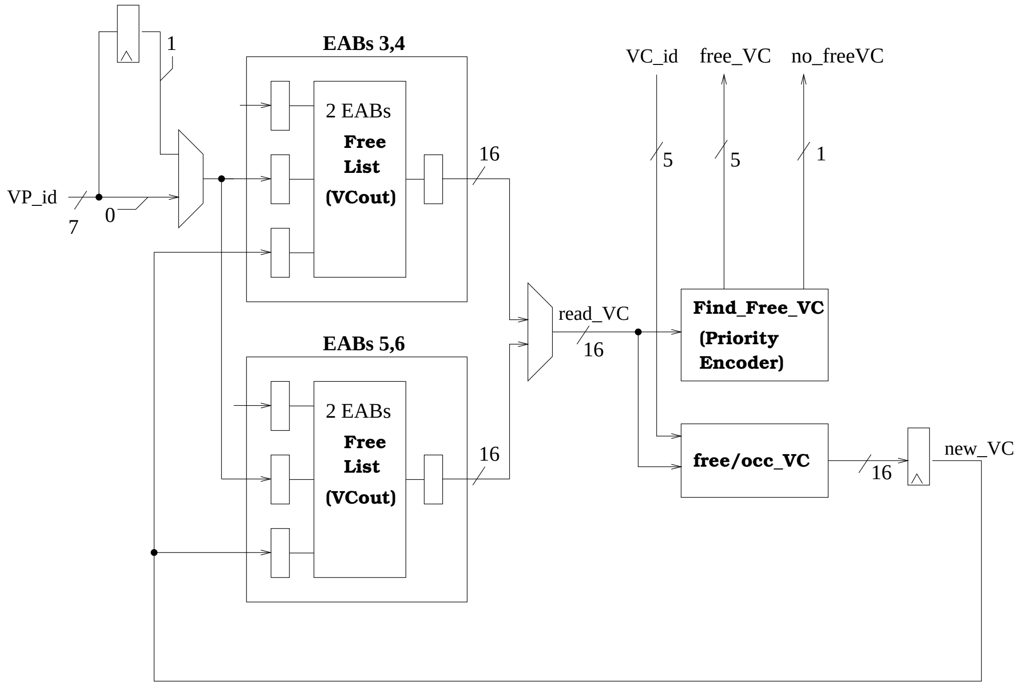

Figure 9: Free List Datapath.

Each direction holds its Free List in two EABs. The memory organization of two EABs, combined together, is 256x16 bits. Hence, two reads are needed in order to read the 32 bits corresponding to the 32 VC's of an output VP. The first 16 bits are held in the first half of the two EABs and the rest in the second half.

All three Free List operations start by reading the 32 bits out of the bitmap that correspond to the 32 VC's of a VP, specified by the *VP_id* input bus. These bits appear on the bus *read_VC* in two consecutive cycles. The combination logic named *Find_Free_VC*, finds the first free VC, which is the first zero bit of the *read_VC* bus, by using a priority encoder. Only if the first 16 bits of the 32 VC's are all occupied the circuit will use the next 16 bits. The outputs of this circuit are the free VC found, and a bit specifying whether a free VC was found or not.

The *free/occ_VC* combinational circuit frees or occupies the VC specified by the *read_VC* input bus using a decoder. The 32-bit vector that specifies the state of the 32 VC's is then updated and written back to the EABs, with two write transactions on the 16-bit bus *new_VC*.

When the *reset* input signal of the FPGA is activated by pressing the reset button of the board the content of the Free List is set to zero. Both directions use the same combinational circuits, because the time scheduling of the memory guarantees no overlap between the Free List operations.

6.1.3 DRAM/Manager communication datapath

The time scheduling in Fig. 5 is implemented in hardware by using a counter which helps to create the control signals. The DRAM communication includes the Routing Table reads, and the Connection Table modifications, whose order is specified in Fig. 5. Fig. 10 shows the datapath of the DRAM and Manager interfaces.

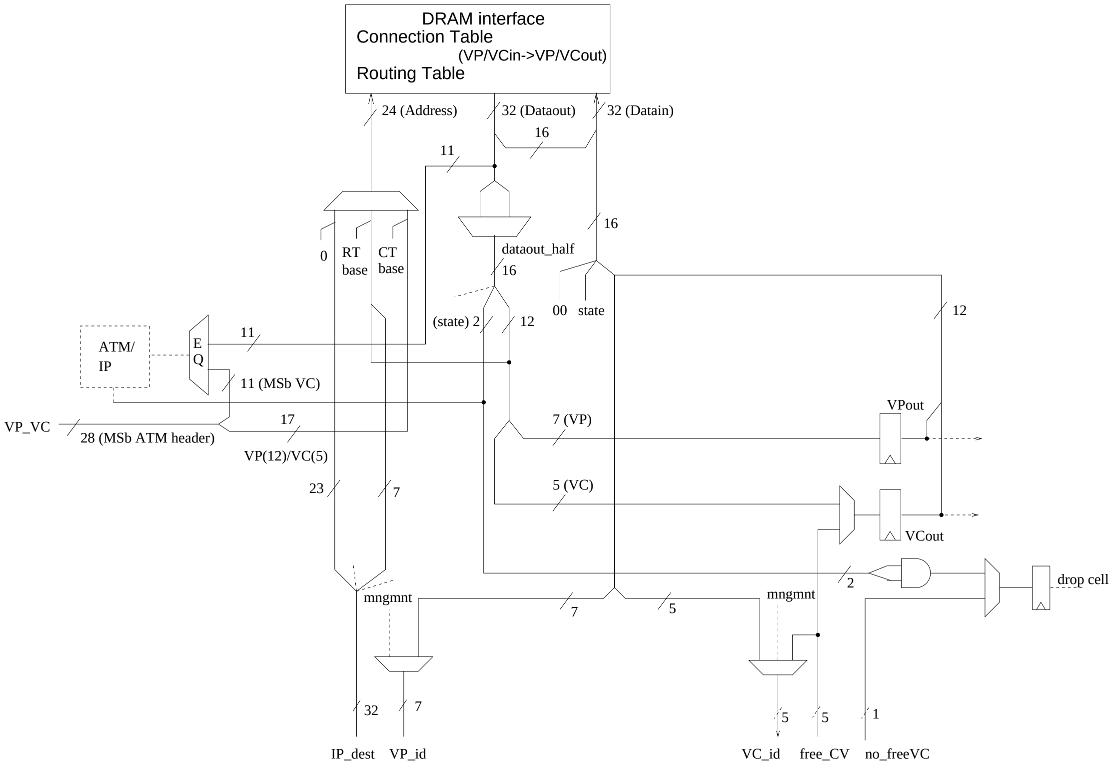

Figure 10: DRAM/Manager communication Datapath.

The DRAM address and data buses are the *Address Datain*, and *Dataout* buses of the Figure. The VP/VC identifiers and the IP destination of the incoming cell are transmitted on the *VP_VC*, and *IP_dest* buses. Registers *VPout* and *VCout* hold the translated VP/VC identifier to use for the outgoing cell. According to [2], the Routing Table is kept in a two-level array. The following steps are implemented by the FSM of the datapath, where the first three are according to [2].

- The first level of the Routing Table, kept in the first SIMM, is indexed with the 24 MS bits of the *IP_dest*. The indexing is implemented by writing the 23 MS bits of the *IP_dest* on the address bus and selecting the MS half or the LS half (*dataout_half*) of the 32-bit *dataout* according to the 24th bit of the *IP_dest*. The selected half contains the output VP of the cell, and a valid bit which defines if the information is valid.

- In the same way, the second-level array is indexed with the 12 LS bits of the *dataout_half* and the 8 LS bits of the *IP_dest* bus. The new data read on the *dataout_half* are valid, if the previous data of the first step were invalid.
- The output VP read from the Routing Table is written to the *VPout* register.
- The circuit reads the Connection Table entry corresponding to the *VP_VC* bus, while in parallel it finds a free VC of the output VP (*VPout* register) in the Free List.
- If the *State* field of the Connection Table entry is *Inactive IP*, then the VC read from the Free List is written in the *VCout* register, and it becomes occupied only if the incoming cell is not the last of an IP packet. If the *State* field is *Active IP*, the *VPout* and *VCout* fields of the Connection Table entry are written to the *VPout* and *VCout* registers.
- The Connection Table entry is modified according to Fig. 4, if the cell belongs to IP traffic.

The Manager can read and write the Free List, and write the registers that hold the MS bits of the output VP/VC of every outgoing cell. If a VC of the Free List is occupied then no packet can use it until it is freed. Hence, the Manager can activate and deactivate output VC's for IP traffic.

6.1.4 Output Datapath

This part of the Filter datapath includes the two interfaces with the TNETA chips, for transmitting the outgoing cells in both directions. Fig. 11 shows this datapath.

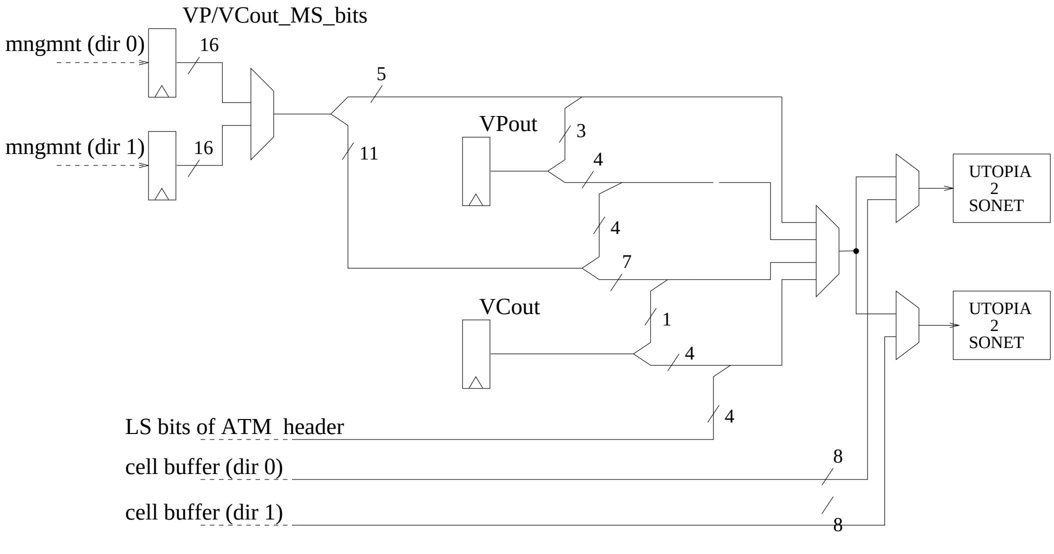

Figure 11: Filter Output datapath.

Registers *VP/VCout_MS_bits* are set by the Manager of the Filter, and define the 5 MS bits of the VP and the 11 MS bits of the VC for every outgoing cell that belongs to IP traffic. Registers *VP_out* and *VC_out* hold the LS bits of the VP and VC of the outgoing cells. These four registers are used only if the outgoing cell belongs to IP traffic, otherwise the header of the cell is read from the corresponding *cell buffer* bus. All the rest bytes of the cell are also read from the *cell buffer* buses.

7 Future Work

Several improvements can be proposed in order to increase the functionality and performance of the Filter.

First the 11 MS bits of the VC of the outgoing cell can be determined by indexing an array of 128 VC's x 11 bits with the 7 LS bits of its VP. This modification is depicted in Fig. 1. Hence every output VP can use a different set of 32 consecutive VC's for its outgoing IP traffic.

Also the Filter's 2-cell internal buffer, which holds the incoming cells, can be increased in size in order to decrease the percentage of frames lost shown in Fig. 6. The 2 EABs used to accommodate the cell buffer can hold up to 9 cells, hence we can have a 9-cell buffer instead of the 2-cell buffer.

Another modification can be proposed concerning the timing of the Free List operations. A VC of the Free List is freed when the last cell of the IP packet holding the VC is transmitted by the FPGA. If this cell is lost the last cell of the next packet with the same IP destination will free the VC. Hence, no packet can use this VC until the arrival of the the aforementioned packet, which may be delayed enough. In order to prevent such an undesired long time deactivation of a VC, every VC can be freed after a specified time has passed since the last cell with this output VC was transmitted.

Finally in [1] another Filter is proposed where incoming cells are buffered in DRAM and hence they are not dropped when the Free VC List is empty.

8 Conclusion

This report described the hardware design of the Filter's core functions, its simulation in verilog and its on-FPGA implementation. Wormhole IP routing Filters in combination with ATM subnetworks can replace today's expensive and slow routers. We exploit today's high speed hardware to process and translate the VP/VC identifier of each cell belonging to IP traffic in a timely fashion. The routing is implemented as soon as the first cell of an IP packet has arrived, contrary to other IP routers where the routing is implemented on the whole IP packet. The hardware design is simple and small enough, hence it is certainly an attractive and cheap solution to IP over ATM.

9 Acknowledgments

I would like to thank ICS-FORTH for funding my work, and professor Manolis Katevenis for his overall guidance and remarks.

References

[1] Manolis Katevenis, Iakovos Mavroidis, Ioannis Mavroidis, and George Glykopoulos: *"Wormhole IP over (Connectionless) ATM"*, http://archvlsi.ics.forth.gr/wormholeIP/arch98/wIP_98.html

[2] P. Gupta, S. Lin, N. McKeown: *"Routing Lookups in Hardware at Memory Access Speeds"*, Proceedings of IEEE Infocom, San Francisco, CA USA, April 1998.

[3] G. Sapountzis: *"IP Routing Table Organization and Management in the Wormhole IP Routing Filter"*, Technical Report TR-257, ICS-FORTH, June 1999, ftp.ics.forth.gr/tech-reports/1999/1999.TR257.RTOrGMng.ps.gz

[4] Texas Instruments Inc. : *"TNETA1500, 155.52-Mbit/s SONET/SDH ATM Receiver Transmitter"*, http://www.ti.com/sc/docs/products/network/tneta1500.html#Datasheets

[5] A. Brodnik, S. Carlsson, M. Degermark, and S. Pink: *"Small Forwarding Tables for Fast Routing Lookups"*.

[6] R. Barnett (rbarn@westell.com): *"Connectionless ATM"*, IEEE Electronics & Communications Engineering Journal, Great Britain, October 1997, pp. 221-230.

[7] Peter Newman, Greg Minshall, and Tom Lyon: *"IP Switching: ATM Under IP"*, Ipsilon Networks, November 1997.

[8] Antony Alles: *"ATM Internetworking"*, Cisco Systems, Inc. May 1995.