ICACT'99 paper

Implementation of a QOS buffering in ASAH-L4 ASIC with Weighted Round-Robin and Maximum Delay Threshold

Chan Kim, SangHo Lee, GangUk Hwang, ByungDo Go, JaeGeun Kim
ETRI , 161 Kajeong-dong, Yusong-gu, Taejon, Korea , ckim@etri.re.kr

Abstract -- This paper describes an implementation of a QOS buffering in a 622Mbps ATM layer processing ASIC called ASAH-L4. It is based on linked list shared buffer using external SRAM and adopts simple weighted round-robin for multi-port, multi-class traffic groups. Then we propose a refinement of this scheme to limit the total queuing delay for some selected sub-queues which is possible by incorporating the next cell's arrival time in the linked list storage unit. This whole mechanism provides bandwidth priority over several classes of traffic while guaranteeing minimum bandwidth for low priority classes and it explicitly controls time delay for some high priority classes.

Keywords – ATM, QOS, shared memory, scheduling

I. BACKGROUND

Shared Memory Buffering using Linked List

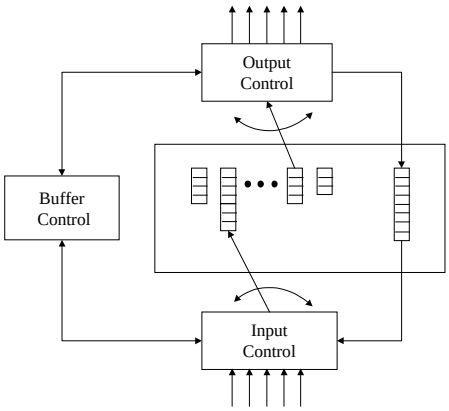

Fig.1 General Shared Buffer QOS Buffering

In ATM switches, cells received from physical lines are stored into a large buffer memory before or after switching. For priority service scheduling for QOS support, the cells are stored into separate queues according to their destined port and QOS class information retrieved from the cell header. These separate buffer queues are generally implemented using common external SRAM memory and many sub-queues are formed using linked list data structure in the SRAM.

When a cell is to be stored at a specific sub-queue, a new storage location is assigned and the cell is stored into that location. The location is then linked to the tail of the corresponding sub-queue by writing the new address pointer to the previous tail location. When a cell is to be read out from a specific sub-queue, the cell is read from the head of the specific sub-queue and the sub-queue's head address is updated to the next head address pointed to by the location being read. These cell locations linked together by pointers form virtual FIFO memories and the queue manager should manage the head and tail address of each sub-queue as well as the queue level each time a cell enters or exits its sub-queue. Fig.2 shows the general form of linked list shared buffer ATM QOS buffering scheme and the same method was used in ASAH-L4.

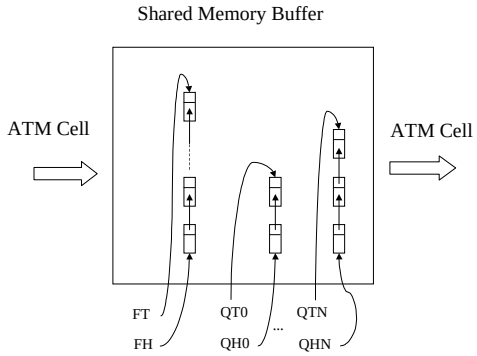

Fig.2 Queue Formation with Linked List

Prior Service Scheduling Methods

In ATM switches, it is required to support various QOS grades for each connection which are negotiated during the call setup. In ATM, the QOS is defined by cell loss ratio, cell delay, and cell delay variation. Some real-time CBR traffic is delay sensitive but not loss-sensitive while other VBR data traffic is loss-sensitive but delay insensitive. These traffics have different QOS requirements. To meet these various requirements, ATM switches, in dealing with the buffering and service scheduling, must exhibit a well-defined priority service scheme. Many kinds of priority algorithms and architectures were devised. Simply putting absolute priority is not enough because when high-priority traffic enters the buffer with a long burst period, the low priority traffic will be delayed too much and high priority traffic will be serviced unnecessarily too fast. In an architecture shown in Chao's paper[1], to handle different levels of loss and delay priorities independently, the cell departure and possible loss sequence is independently scheduled by delay and loss priority sequencer chips and the cell's departure and removal during congestion is governed by the sequence arranged by the sequencers. The sequencer chips inserts and shifts in parallel and distributive manner the priority and address of the cells every time a new cell is arrived. Katevenis [2] presents another architecture where the shared memory is implemented using CAM and priority encoder for

implementing free buffer pool. In his paper, the ATM cell is made into a single word(53x8 bit long) and the scanning memory has the location of the store cell. A state machine scans the scanning memory while servicing and decrementing the weigh values cycling through the scanning memory. In this way, the bandwidth allocation is spread out in a cycle. The most advanced of service algorithms to the author's eyes is Yurie Systems'[3]. Here, basically queues with higher output ranking gets serviced prior to queues with low output ranking. But sub-queue with its aging counter reached at its threshold is serviced with highest priority. And among the same output ranking, sub-queues which has reached its minimum queue level is served first. The minimum level is defined for each classes and this has an effect of negative feedback on the increasing buffer length and is taking the real traffic inflow situation into account. Another good mechanism is purging the queue with lowest loss priority first when buffer resource is detected low.

This paper describes an implementation of a QOS buffering in a 622Mbps ATM layer processing ASIC called ASAH-L4 which basically adopts weighted round-robin service scheduling. Then, this paper presents a refinement of this implementation. This new scheme, with the weighted round robin scheduling, can provide the efficiency and flexibility to provides the means for directly controlling cell delays for some classes as well as assigning bandwidth usage priority for all classes.

II. QOS BUFFERING IN ASAH-L4 ASIC

Architecture of ASAH-L4 ASIC

Fig. 3 shows the architecture of ASAH-L4 ASIC. The QOS buffer control block resides in the Receive Cell Router (QBWB_RX) and Transmit Cell Router (QBWB_TX). The receiver and transmitter function of the Cell Router is the same except for some differences. The QBWB_RX performs basic cell routing functions such as loopback, extraction, buffering, passing (non-buffering mode) and also performs other processing like header replacement, tagging, discarding, RM field replacement, etc. gathering information coming from table lookup, UPC, and OAM blocks. The QBRB pre-pends routing tag in receiver direction and it inserts PM cells in the transmit direction because PM cell insertion is closely related to actual cell transmission.

For QOS buffering, the shared memory is divided into maximum 8 sub-queues according to classes in the receiver direction. For transmit direction, the shared memory is largely divided into maximum 8 port queues and each port queue is divided into maximum 8 sub-queues according to number of ports and classes used. The shared memory is fully shared by all ports and class sub-queues dynamically using the linked list method previously explained. Since the theme of this paper is the QOS buffering of this ASIC, only the interaction of QOS buffer controlling block internal to the QBWB block will be discussed with regard to QOS buffering.

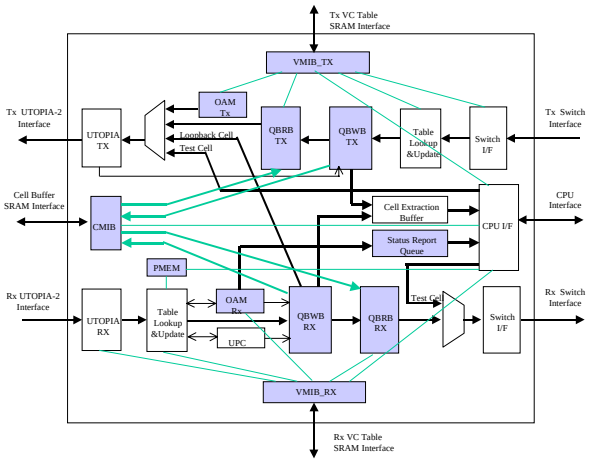

Fig.3 Block Diagram of ASAH-L4 ASIC

When Table Lookup requests a new cell's processing with a cell waiting in UTOPIA FIFO and with all UPC and OAM processing has finished, the QBWB block determines where the cell should go. In QOS mode, when the cell should be stored into a specific sub-queue, the QBWB block assigns a free buffer location and writes the cell into the new address and updates the queue data. At the same time, after scheduling, the QBWB requests a cell's service to QBRB. The QBRB reads the cell and as soon as the service starts, reports the completion to the QBWB with data for updating the sub-queue like new header address.

Actually, the Output Controller in the figure is the QBRB and all others are implemented in QBWB but the block hierarchy can be appropriately determined for synthesis and layout efficiency with human convenience in understanding. The actual implementation is free to designers.

Service Scheduling in ASAH-L4 ASIC

The refined service algorithm of ASAH-L4 is described in this section. It is a merged version of weighted round-robin plus delay threshold. The real implementation did not include the maximum queue delay threshold. But here, rather than separately explaining the actual implementation and the refinement, we chose to describe only the refined version of it, from which the reader can easily guess what was implemented and what is the refined feature of it.

The purpose of this algorithm is to serve the sub-queues in a programmable sequence and at the same time to limit the queuing delay to programmable amounts for some delay-sensitive classes. This can be though of as assigning priority and guaranteeing minimum value on bandwidth usage ratio while limiting actual queuing delay to programmable amount for some classes. The controlled delay value is set on real time unit basis not like those guessed from queue length as in conventional methods. The service scheduler runs in parallel with the actual service processes. It launches a new service and as soon as the service begins, starts the decision process for the next cell because the queue data will be updated right after the start of a service. This scheduling action is fast enough not to cause the system to wait for the decision to finish. From the point of server's view, the time

for the decision in the scheduler is buried into the server's processing time and is invisible.

In this paper, it is assumed that the shared memory is divided into many sub-queues according to the output port and service class. The scheduler decides which sub-queue to service next. It first determines the service port and then the service class. For each port and class, there is a sub-queue associated with it. There is a programmable sequence register for service port and service class. The port sequence defines in what order the non-empty ports will be serviced and the class sequence register defines in what order the non-empty classes will be serviced for a port. Fig. 5 shows the concept of sequenced service. In the actual implementation, since there can be maximum 8 ports and 8 classes for transmitter, and 8 classes for receiver, the port sequence register and class sequence register are 16 entry long. This puts an limitation on the granularity on the priority assignment but we thought it doesn't matter much for intended level of QOS support. The index pointers of class sequence register is independently maintained for each port thus the class sequence is preserved for all ports regardless of other intervening port services.

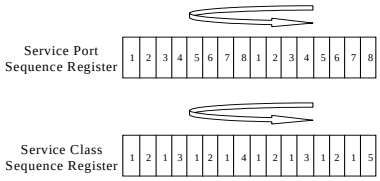

Fig.4 Sequence Register for Ports and Classes

Besides, for each class, there is a maximum queue delay threshold register. Starting the decision, the service scheduler checks if any sub-queue has a cell at its head location which has reached or passed its maximum delay threshold. If such sub-queue exists, that sub-queue is selected. If such sub-queue is not detected, it starts scanning the port sequence register and class register as explained above. This parallel examination of cell delay(current time minus arrival time) is made possible by putting time stamp of next arrival time in the linked list storage unit as shown in Fig. 5.

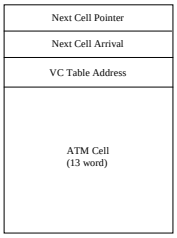

Fig. 5 Unit Storage Format

This storage unit contains ATM cell, the pointer to link the storage units into logical FIFO, and the arrival time of the next cell. This next arrival time is not written at the time the contained cell is written but is written later with the next pointer when a new cell is linked after that location. Fig. 6 shows how the time stamp is written. The unit also contains the VC table's address the cell belongs to. This field is used to retrieve routing tags in the receiver and to retrieve multicast port id and keeping track of statistics data in the transmitter.

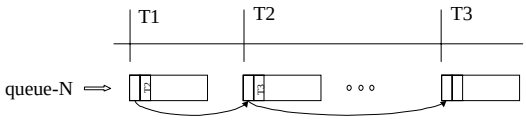

Fig.6 Next Arrival Time Stamping

For simplicity, the processing blocks in QBWB and QBRB related to QOS buffering can be simplified as in Fig. 7. They can be organized into external shared memory, input controller, output controller, queue manager, free buffer manager, and service scheduler.

The external shared memory is segmented into basic storage units as shown in Fig. 5 and the unused storage units are also connected into linked list and this list is called free list. This free-list is managed by the free-list manager. The FLM gives a free address when requested by input controller and appends a used address to the free list when requested by the output controller.

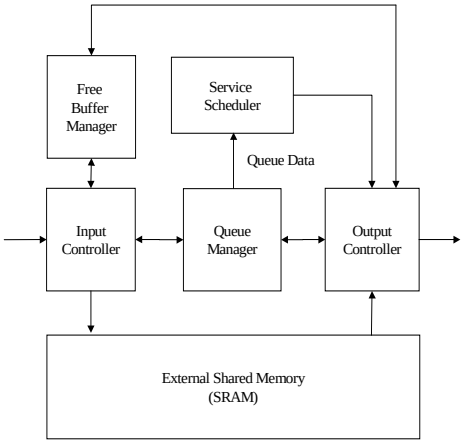

Fig. 7 QOS Buffer Control Blocks

The queue manager updates the head, tail, length of each queue and also updates the next arrival time of the cells located at all sub-queues. The arrival time is used to calculate actual queuing delay of cells at the head of line of each sub-queue.

The input controller, requests a free address when received a cell and writes the cell into that location and links it to the tail of the corresponding sub-queue by writing the new address into the previous tail location. It also write the arrival time to the previous tail location so that the output controller can know the arrival time of the next head cell when reading from the head of a sub-queue. The input controller has optional selective discard mode and buffer level threshold discard function.

The output controller receives the service address from the queue manager and reads cell from that address. It informs the service completion to the queue manager providing next

head address and next arrival time for the serviced sub-queue as soon as this information becomes available. Since these data are read before the actual cell, the queue manager updates the queue data earlier before the actual service completion, the scheduler can start the decision process and finish the decision well before the service completion.

The Queue manager internally manages all the sub-queues' head, tail, length and next arrival time. In ASAH-L4, since the head, tail, and length data is kept in DPRAM, the manager performs read-modify- write for sub-queues when a cell enters or exits any sub-queue. In the mean time, some flags like whether it is empty or not, whether the level is one or not(this is used for some marginal cases in pointer update). If we want to consider the queue level in the scheduling, the level data still can be kept in DPRAM and needed flags like whether the level got greater than some threshold can be updated during the read-modify-write action. Even the arrival time(i.e., waiting time) can also be stored in DPRAM in the same way because the waiting time is derived from arrival time by subtracting it from the current time which incremented regardless of the queue states.

The queue manager returns the tail address and empty flag to the input controller so that the input controller can link the new cell or bypass the linking when empty. It also updates the tail address and head address(when a cell enters an empty sub-queue). It increments the sub-queue's level by one.

On receiving the selected sub-queue index from the scheduler, the queue manager passes the selected sub-queue's head address to the output controller and requests service. It also updates the serviced queue's head and next arrival time when informed the completion of the service. It also decrements the queue's level by one.

The queue manager provides all the queue length (in ASAH-L4, only the flag whether the sub-queue is empty or not), the arrival time of the cell in the head of line of each sub-queue for scheduling.

The housekeeping operations are apparent for entering and exiting events but there are numerous complex cases like "entering into a sub-queue when it is being serviced and not updated yet" etc. In some cases the header replacement after queue service should be disabled when the queue is going empty or a new cell has arrived during the service launch causing the queue level out of empty state. The queue manager takes all these cases into account and keeps the sub-queues information always correct.

The service scheduler determines the sub-queue to be serviced and lets the index out to the queue manager. And begins the new decision process as soon as the queue data is updated right after the cell buffer read and head/level update. As mentioned earlier, this process is quick enough and it skips the entry when no cell is in it. This makes the scheduler always stay in "decide and wait" state and guarantees the performance bottleneck becomes actual cell reading and outing process. This is important for the 622Mbps throughput of the device. On scanning the sequence register, the scheduler examines whether there's any time-overred sub-queue. If it exists, that queue is serviced but if not, regular sequence scanning initiates. The port index stops at the next position when a port is selected and the class index of the port is also stored in the next position when a sub-class of the port is selected. And the stored class index is later used when that port is selected again. This ensures that the class sequencing is preserved for each port. Fig. 8 again illustrates then scheduling algorithm mentioned above.

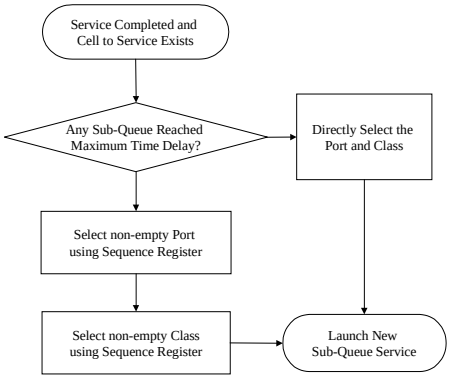

Fig. 8 Service Scheduling Algorithm

Meeting Throughput Requirements

All these functions were(excluding the arrival time related processing) implemented in ASAH-L4 and it operates at 50MHz. But this limitation comes not from internal gate delay but from the fact the whole chip runs at a single clock which is also given out as the UTOPIA master clock and is limited to 50MHz. But if the architecture is slightly modified to accommodate the asynchronous operation between UTOPIA and internal processing, there will be more processing gain achieved. In 622Mbps ATM, the cell arrives every 675 nsec and this is about 34 clocks in 50MHz operation. To handle this amount of traffic with all the table lookup, OAM procedures including PM, and UPC, and writing and reading ATM cells into share memory, it is necessary to use separate memory for VC table and cell memory. So in ASAH-L4 there are separate interfaces for RX VC table and TX VC table and cell buffer memory. Accessing VC table and cell memory is performed at the same time in parallel. Using 32 bit memory bus, it takes 14 clocks to write ATM cell(12 for cell, 1 for pointer, and 1 for VC table address) and also 14 clock for cell reading. Besides, to link the sub-queues and update free-list, we need 2 more clocks. So there is 30 clocks needed in the cell memory bus for every cell passing it. To meet this requirement, the local SRAM interface was very optimized not to loose clock cycles during arbitration and data movement. All details of these techniques will not be discussed here.

III. Conclusion

This paper describes the QOS buffering implementation in a 622Mbps ATM layer ASIC called ASAH-L4 and suggests an enhancement of the scheduling algorithm. The chip uses external synchronous SRAM for cell buffer and dynamically assigns buffer storage for several ports and classes. The QOS

buffering is an independent option for receive and transmit direction. For 622Mbps traffic, the egress traffic will only be QOS buffered and scheduled. This is due to its processing speed limit primarily caused by clocking scheme with regard to UTOPIA interface and partly from gate delay. It adopts a weighted round robin scheme where each port or class is assigned priority in bandwidth usage ratio but guaranteed in minimum usage ratio as well. With a delay threshold mechanism, it will also limit queuing delay of some delay sensitive classes to certain absolute maximums.

The delay threshold mechanism using absolute values can be easily incorporated into the existing implementation developed for ASAH-L4, and this will ensure that some classes will experience lesser delay regardless of the slot assignment. The delay is controllable because of writing time stamp in the linked list chain.

There is an important notion in using existing, un-refined algorithm. Assigning more slot will imply more service rate and this means lower delay in general. But if the actual traffic inflow ratio is larger than the slot ratio, and if the traffic is almost at its maximum, since the processing is at 50MHz, the priority control effect might be weak. But this can be reduced if we put more contrast in the slot assignment ratio than actual traffic inflow ratio. Since the sequence register is programmable and there are sub-queue thresholds and selective discard modes, the loss and delay could be adaptively controlled as required by the real situation.

6. REFERENCES

1. J. Chao and H. Pekcan, "Queue Management with Multiple Delay and Loss Priorities for ATM Switches," *IEEE Int'l Conf. On Comms*., vol.2, pp.1169-1173, May.1994.

2. M. Katevenis et. al, "Weighted Round-Robin Cell Multiplexing in a General-Purpose ATM Switch Chip," *IEEE JSAC*, vol.9,No.8, ,pp.1265-1279, Oct 1991.

3. Kwok-Leung et. al, "Queue Management to Serve Variable and Constant Bit Rate Traffic at Multiple Quality of Service Levels in a ATM Switch", United States Patent 5,757,771. May. 26, 1998/10/28