
ELSEVIER

Parallel Computing 24 (1998) 2143–2148

PARALLEL COMPUTING

Short communication

An efficient and orderly implementation of bypass queue under bursty traffic

Joe Shang-Chieh Wu, Ying-Dar Lin *

Department of Computer and Information Science, National Chiao Tung University, 1001 Ta Hsuch Rd., Hsinchu, Taiwan

Received 10 February 1998; received in revised form 15 April 1998; accepted 20 August 1998

Abstract

Sharma and Pinnu proposed an implementation of bypass queue by many FIFOs; unfortunately, the detailed procedure paid little attention to maintaining cell sequence, which is an important feature in ATM network. In this paper, we propose an improved architecture which guarantees cell sequence integrity and describe its related operating procedures. © 1998 Elsevier Science B.V. All rights reserved.

Keywords: Bypass queue; Cell sequence integrity; Head of line blocking; Switches

1. Introduction

Sharma and Pinnu [1] described a smart implementation of the bypass queue [2,3]. In Ref. [1], each input port maintains many FIFOs to simulate the operation of the bypass queue. For each input port, when a new cell arrives, its destined output port number is compared with that of the previous incoming cell. If they are the same, the new cell is put in the same FIFO in which the previous one was put. If they are different, the new cell is put in the next FIFO.

Unfortunately, the above procedure paid little attention to maintaining cell sequence integrity [4], an important feature in ATM networks, for each input ATM virtual channel. The following scenario, associated with a sequence of graphs in Fig. 1(a)–(g), demonstrates the condition assuming the number of FIFO queues to be 3 for a given input port.

1. All FIFOs are empty originally, as shown in Fig. 1a.
2. Four cells, for output port A arrives, are put in FIFO 0, as shown in Fig. 1(b).

* Corresponding author. E-mail: ydlin@cis.nctu.edu.tw

PII: S0167-8191(98)00088-X

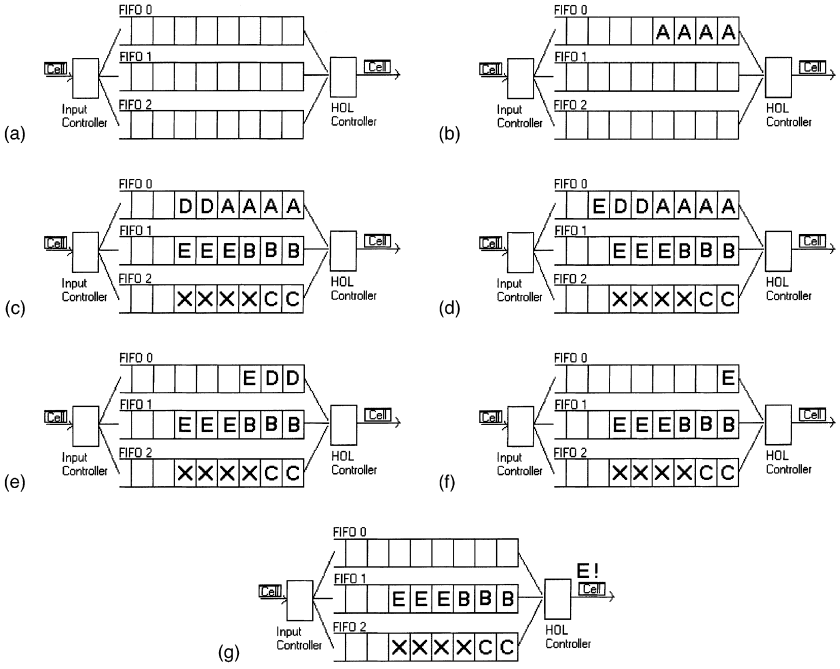

Fig. 1. (a) Each input port, (b) Scenario 1, (c) Scenario 2, (d) Scenario 3, (e) Scenario 4, (f) Scenario 5, (g) Scenario 6.

3. Three cells, for output port B arrives, are put in FIFO 1.
4. Two cells, for output port C arrives, are put in FIFO 2.
5. Two cells, for output port D arrives, are put in FIFO 0.
6. Three cells, for output port E arrives, are put in FIFO 1.
7. Four cells, for output port X arrives, are put in FIFO 2, as shown in Fig. 1(c).
8. One cell, for output port E, is put in FIFO 0 using the round-robin method suggested in Ref. [1], as shown in Fig. 1(d).
9. The input port is informed of the availability of output port A, and four cells in FIFO 0 leave the input port, as shown in Fig. 1(e).
10. The input port is informed of the availability of output port D, and two cells in FIFO 0 leave the input port, as shown in Fig. 1(f).
11. The input port is informed of the availability of output port E, and one cell in FIFO 0 leaves the input port, as shown in Fig. 1(g). But the earlier cells for output port E are still in FIFO 1!

The inability to differentiate the sequence for incoming cells is the primary reason for this condition. To re-sequence cells in the output ports by hardware is a simple solution, but, if possible, an economical approach should be investigated to prevent re-sequencing.

In Section 2, we present our new hardware architecture. We describe the hardware overhead in Section 3, the operating procedures in Section 4, and show an example in Section 5. Conclusion is given in Section 6.

2. Architecture

The architecture for each input queue is composed of three major parts: the Input Controller, FIFO queues and HOL Controller, as shown in Fig. 2(a)–(c), respectively. The major difference from Ref. [1] is the addition of registers and counters. For a given input queue, we add extra counters at Input Controller, and extra registers at HOL Controller. The number of registers and the number of counters are both the same as the number of output ports in this switch. Furthermore, for each entry in these FIFOs, we also need one more field for the stamp value for the corresponding cell.

When each incoming cell enters Input Controller, the current value of the counter corresponding to the cell's output port is stamped on the cell. When each

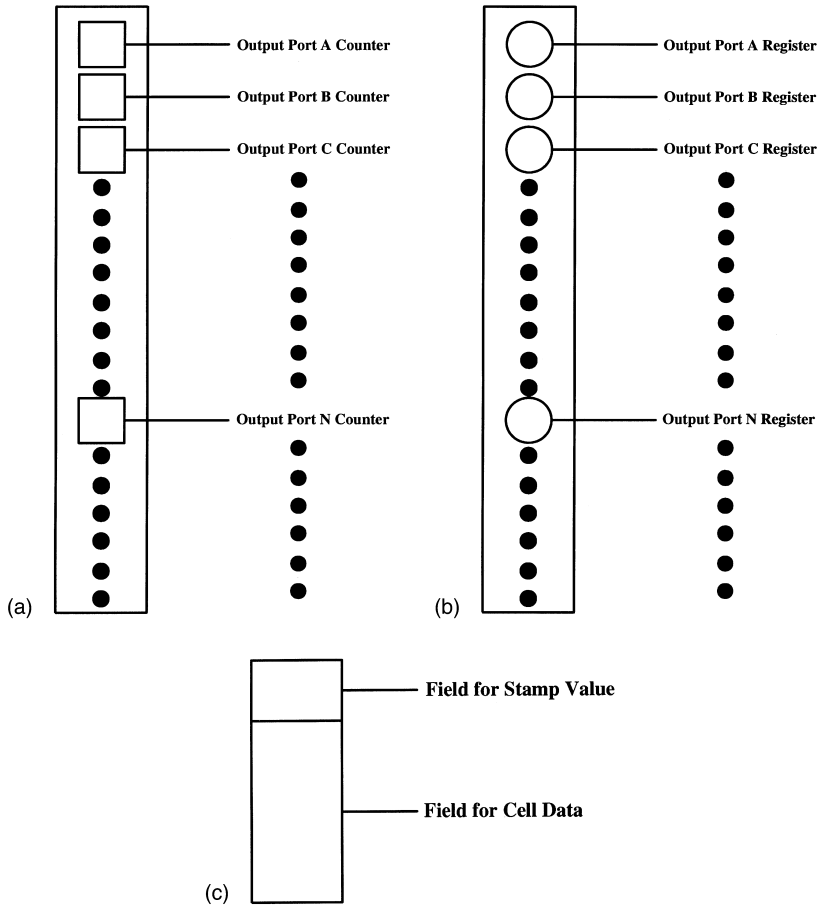

Fig. 2. (a) Input Controller for a given input port, (b) HOL Controller for a given output port, (c) Each entry in FIFO queues.

cell leaves HOL controller, the stamp is removed. The detailed operation is given in Section 3.

3. Hardware overhead

The extra hardware includes counters, registers and modified FIFOs. For an N × N switch, we need totally N^2 counters and N^2 registers. The size for them is $\log_2 W$ bits if the switch has W FIFOs in each input port.

For each entry in all FIFOs, we also need extra space, $\log_2 W$ bits, for stamping. The corresponding overhead, in percentage, is, if the size for each FIFO is L,

$$\frac{N \times W \times L \times \frac{1}{8}(\log_2 W)}{N \times W \times L \times 53} \times 100\%.$$

For an 8 × 8 switch, with 16 FIFOs in each input port and 16 entries for each FIFO, the incurred overhead in FIFOs is 0.94%. For 64 counters and 64 registers, we need just 4 bits for each of them.

4. Procedures

For each incoming cell, the modified sequence of steps, based on Sharma's method [1], taken by the input controller is as follows.

1. Increment the counter, associated with output port of the current incoming cell, in module W fashion.
2. Add a stamp, with the value of corresponding counter, to the cell.
3. Compare the output port of current cell with the output port of previous cell.
4. If both are equal, then go to step 5, else, go to step 6.
5. Check the status of the input queue pointed to by the input pointer. If the buffer is not full, then place the current cell in that queue, else go to step 6.
6. Increment the input pointer in module W fashion.
7. Repeat steps 5 and 6 till an empty input queue is found or the number of queues searched is equal to W.
8. If step 7 fails (i.e. none of the input queues is found empty), decrement the corresponding counter in module W fashion, and discard the cell.

The sequence of steps taken by the HOL controller in each time slot is as follows.

1. Find a candidate cell by examining W cells from the head of line of each queue. If failed, then this input port does nothing in this time slot.
2. Update the corresponding register with the value of time stamp in the candidate cell.
3. Strip the stamp off the candidate cell, and send it out.

The candidate cell must meet the following two requirements.

1. The output port is available.
2. The stamp value must be the same as the increment of the value of corresponding register in module W fashion.

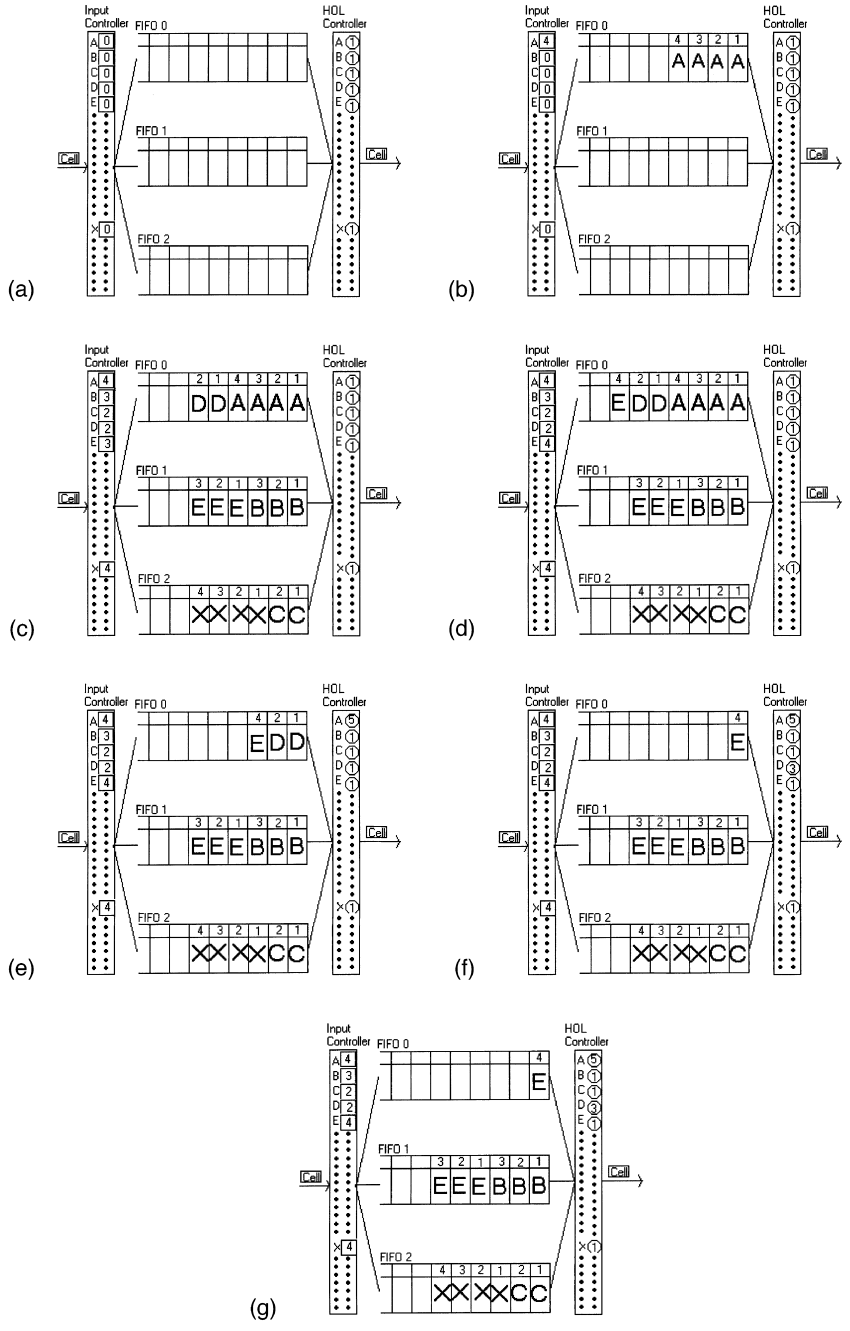

Fig. 3. (a) New Architecture, (b) New Scenario 1, (c) New Scenario 2, (d) New Scenario 3, (e) New Scenario 4, (f) New Scenario 5, (g) New Scenario 6.

5. An example

The following scenario, combined with sequences of graphs in Fig. 3(a)–(g), shows an example for maintaining cell sequence integrity if the number of FIFO queues is 3 for a given input port.

1. Originally, all FIFOs are empty, the values of all registers are reset to 1, and the values of all counters are also reset to 0, as shown in Fig. 3(a).
2. Four cells, for output port A arrives, are put in the FIFO 0, as shown in Fig. 3(b).
3. Three cells, for output port B arrives, are put in FIFO 1.
4. Two cells, for output port C arrives, are put in FIFO 2.
5. Two cells, for output port D arrives, are put in FIFO 0.
6. Three cells, for output port E arrives, are put in FIFO 1.
7. Four cells, for output port X arrives, are put in FIFO 2, as shown in Fig. 3(c).
8. One cell, for output port E, is put in FIFO 0 using the round-robin method suggested in Ref. [1], as shown in Fig. 3(d).
9. The input port is informed of the availability of output port A, and four cells in the FIFO 0 leave the input port, as shown in Fig. 3(e).
10. The input port is informed of the availability of output port D, and two cells in the FIFO 0 leave the input port, as shown in Fig. 3(f).
11. The input port is informed of the availability of output port E, but the cell in the FIFO 0 is blocked due to the incorrect stamp value, as shown in Fig. 3(g).

It is clear that the sequence integrity is maintained by the co-operation of registers and counters.

6. Conclusion

The values of registers and counters show significant information. Each register, whose maximum value is W, shows the stamp value of the next incoming cell at HOL Controller; however, each counter, whose maximum value is also W, shows that of the last leaving cell at Input Controller.

We offer a modified implementation, with FIFO queues, of the bypass queue. The architecture does maintain the cell sequence integrity and the related overhead is small.

References

[1] N.K. Sharma, M.R. Pinnu, An efficient implementation of bypass queue under bursty traffic, Parallel Computing 23 (1997) 777–781.
[2] M. Karol, M. Hluchyj, Input versus output queueing in a space division switch, IEEE Trans. Commun. COM- 35 (1987) 1347–1356.
[3] M. Hluchyj, M. Karol, Queuing in high-performance packet switching, IEEE J. Sel. Areas Commun. 6 (1987) 1587–1596.
[4] R. Händel, M.N. Huber, S. Schröder, ATM Networks Concepts, Protocols, Applications, Addison-Wesley, 1994, p. 21.