고속 입력큐 스위치 패브릭을 위한 3차원 라운드로빈 스케줄러

정갑중* · 이범철**
*경주대학교 · **한국전자통신연구원

THREE-DIMENSIONAL ROUND-ROBIN SCHEDULER FOR ADVANCED INPUT QUEUING SWITCHES

Gab Joong Jeong* · Bhum-Cheol Lee**
*Gyeongju University · **ETRI
E-mail : gjjeong@kyongju.ac.kr

요 약

본 논문은 고성능 공통 입력버퍼를 가지는 패킷 스위치 패브릭을 위한 새로운 3차원 라운드로빈 스케줄러에 관한 연구이다. 본 논문에서 제안된 스케줄러는 각 입력 버퍼에서 각 출력 큐를 독립적으로 관리하는 분산형 공통 버퍼를 가지는 스위치 패브릭구조에서 고속으로 동작하는 스케줄러이다. 제안된 스케줄러는 M 개의 각 공통 입력 버퍼가 K개의 입력 및 출력 포트를 가지고 N 개의 가상 출력 큐를 관리하고 K≥M (K≤M)일 때 매 K(M) 사이클 마다 MxK 개의 가상 출력 큐들이 목적 출력 포트로 패킷을 전송할 수 있도록 스케줄링한다. 대량 병렬 스케줄링 구조를 이용하여 대용량의 스위칭 포트를 가지는 고성능 스위치 패브릭에의 고속 응용을 지원한다.

ABSTRACT

This paper presents a new, three-dimensional round-robin scheduler that provides high throughput and fair access in an advanced input-queued packet switch using shared input buffers. We consider an architecture in which each input port group shares a common buffer and maintains a separate queue for each output, which is called the distributed common input buffer switch. In an NxN switch, our scheduler determines which queue in the total MxN input queues is served during each time slot where M is the number of common buffers. We suppose that each common buffer has K input ports and K output ports, and manages N output queues. The 3DRR scheduler determines MxK queues in every K(M) cycle when K≥M (K≤M), and provides massively parallel processing for the applications of high-speed switches with a large number of ports. The 3-DRR scheduler can be implemented using duplicated simple logic components allowing very high-speed implementation.

keyword
distributed common input buffer, parallel round-robin scheduler, switch fabric

I . Introduction

There has recently been a merging of ATM switches, LAN switches, and IP routers [1]. The architectures of high speed switches and IP routers are built around a cross-point switch that is configured to use a centralized scheduler, and each uses a fixed size cell as a transfer unit. There is a popular perception that input-queued switches suffer from inherently low performance due to head of line (HOL) blocking. HOL blocking arises when the input buffer is arranged as a single FIFO queue.

Rather than maintain a single FIFO queue for all cells, each input maintains a separate queue for each output. This scheme is called "virtual

output queuing" (VOQ) and was first introduced by Tamir et al. in [2]. HOL blocking is eliminated because cells only queue behind cells that are destined to the same output. No cell can be held up by a cell ahead of it that is destined to a different output. When VOQs are used, the possibility of increasing the throughput of an input-queued switch from 58% to 100% for both uniform and non-uniform traffic has been demonstrated [3]. So, cross-point switches that use VOQs have been employed in a number of studies. However, several cells in an input module can be timed to be sent simultaneously since scheduling is independently executed for each output. This is called "buffer blocking" and is a primary factor in throughput saturation. The optimum architecture for input queuing switches introduced by Obara in [4],[5] is the sharing of a common input buffer between several input ports. This architecture enhances the throughput of an input-queued switch.

We consider here the input queuing switch architecture in which several input ports share a common buffer that employs VOQs. When we use a cross-point switch, we require a scheduling algorithm that configures the fabric during each cell time, and decides which inputs will be connected to which outputs. In this paper, we restrict our focus to the efficient and fast scheduling method of our switch architecture.

II. Advanced Input Queuing Switch Fabric

Let us consider the input queuing switch architecture shown in Fig. 1, called the distributed common input buffer switch, where several input ports share a common queue. Let K be the group size of the input ports. There is no change in an input controller which uses VOQs, a space-division switch, and a contention control. The input controllers in an input port group can be replaced with a KxK shared buffer output-queued switch that manages N output queues internally. In this architecture, the K cells are allowed to be sent simultaneously, i.e. all of the K cells to be set at one time are allowed to originate from the same input port, and each of the K cells are destined to different output ports.

In an NxN space-division cross-point switch, there are M (= N/K) common input buffer controllers, and a common input buffer can use K shared communication links simultaneously. When all of M common input buffers request N shared communication links, a centralized scheduler has to determine the non-conflict N input queues among MxN requests during one cell time. Here, each of the K input queue groups has to be selected by different output ports, evading conflicts in an NxN space-division cross-point switch fabric. In the selection of non-conflict input and output combination sets, the centralized scheduler has to consider fair service for all input queues without any starvation ports and maximum input and output combination sets to which to transfer cells of the input queues simultaneously without buffer blocking. Therefore, there is need of a new scheduling algorithm and method for complex scheduling that takes into consideration both fairness and buffer blocking.

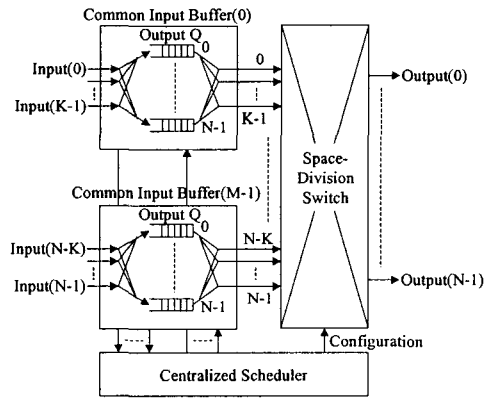

Fig. 1. Distributed common input buffer switch.

III. Three-Dimensional Round-Robin Scheduler

The three-dimensional round-robin (3DRR) scheduling scheme is a variation of a simple basic 2DRR scheduling scheme. The architecture of the proposed 3DRR scheduler consists of an address round rover and two steps of projections as shown in Fig. 2. The address round rover can be implemented by using simple modulo-N counters as in the basic 2DRR scheduler. The concept of 3DRR scheduling is completed through the two steps of the projection operations. In the first projection, there are a request matrix and a parallel arbitration processor. The first projector considers all the heads of the input queues, and projects those input queues which have a request to a virtual screen according to the proposed rotating operation. In the second projection, the KxM (= N) input queues are selected from among all the MxN input queues

which are projected by the first projection.

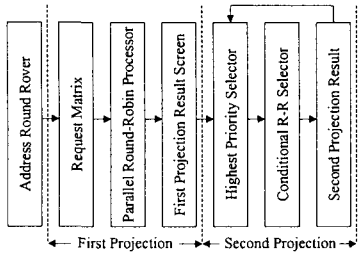

Fig. 2. architecture of parallel scheduler.

In our distributed common input buffer switch, each common input buffer can use K multiple output ports simultaneously. To serve all common input buffers fairly, let us consider the request matrix for the 3DRR scheduling as shown in Fig. 3. Let us configure the request matrix so that all the requests from a common buffer are placed in one column. Then, the request matrix is configured by M columns and N rows. Let us take K copies of the configured MxN request matrix. After that, let us separate each MxN request matrix into K atomic request matrices which are MxM (or MxK) request matrices. As an atomic request, it can be an MxK request matrix in the case of the asymmetric atomic request matrix when MxK = N and M≠ K. Hereafter, we will describe an atomic request as only an MxM request matrix for clear illustration. All the copied and separated request matrices can be virtually implemented by using bus or multiple fan-outs of the original request data.

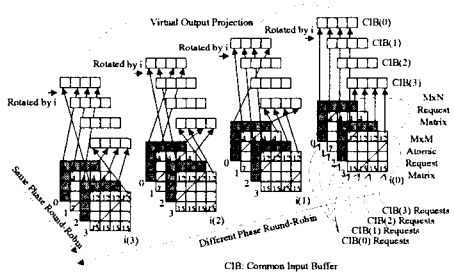

Fig. 3. The request matrix configuration and the first projection structure for a 16x16 switch example using *four 4x4 common input buffers.*

In the first parallel projection, we describe two types of parallel round-robin scheduling, same phase round-robin scheduling and different phase round-robin scheduling. At first, let us consider same phase parallel round-robin scheduling. An MxN request matrix is separated into the K atomic request matrices above. The K atomic matrices are scheduled in parallel by the basic 2DRR scheduling with the same round-robin phase. From the same round-robin start address, the atomic matrices are scheduled during M cycles. In each scheduling cycle, M diagonal requests in each of K atomic request matrices are scheduled during M cycles by M parallel processors. The M diagonal requests in an atomic request matrix are independent of each other for the parallel processing of each *input and output combination set as is usual* with basic 2DRR scheduling. Independent requests can be selected by simple modulo-N counter. If any requests among the M diagonal requests request non-selected input and output ports, the requested input and output ports are selected in parallel at the round-robin cycle. These M parallel selections are performed in K parallel. Therefore, the input and output port sets selected by MxK parallel processors are projected to the K multiple virtual output ports of the individual common input buffer to which each set of input and output ports belongs. The projection location of each set among its K multiple virtual output ports is determined by the order of the separated atomic request matrices. Each selected set of the atomic request matrices of the first MxN request matrix is projected to the K multiple virtual output ports of its common input buffer in ascending order. Hereafter, we call these atomic schedulers an MxN parallel round-robin scheduler.

Here, let us take K copies of the previous MxN parallel round-robin scheduler, which starts scheduling at the same round-robin start signal, for the K atomic request matrices of the MxN request matrix. The K copies of the MxN parallel round-robin scheduler schedule in parallel their MxN request matrices in different phase individually. Every MxN parallel round-robin scheduler starts scheduling from different round-robin start addresses, and their selected input and output sets are projected to the K multiple virtual output ports of their common input buffer after a rotation operation. The K sets of the scheduling result of the i'th MxN parallel round-robin scheduler which starts scheduling at the i'th round-robin cycle, where $0 \le i \le (K-1)$, are rotated by i and projected to their K multiple virtual output ports.

In the second parallel projection, there are two steps for the parallel selection of N input queues among MxN input queues that are projected to

the virtual output ports by the first projection. At the first step, we need to select one input queue which has the highest priority and maximum cost from among the K input queues originated from the same common input buffer at each round-robin cycle. When we find the highest priority input queue, the selected input and output ports in previous round-robin cycles are excluded. K selections in parallel are needed for one common input buffer since one common input buffer has K physical output ports. An input and output set for a physical output port of a common input buffer cannot be selected more than once in the same common input buffer at the same time slot, since every physical port of K considers different request sets each other at every round-robin cycle.

At the second step, a conditional round-robin selector determines the final results of the selected input and output ports to which to transfer the queued input cells in all common input buffers by using the N shared output ports of M common input buffers. It grants K input queues which do not conflict with the output ports of the other common input buffers, among N input queues in a common input buffer at the same time slot. N conditional round-robin selectors determine N input queues at one time slot in parallel. If there is an input queue projected to two or more output ports in multiple, the conditional round-robin selector selects only one input queue of a common input buffer which has the highest priority in the round-robin cycle. Every conditional selector uses the same round-robin sequence for the K sets of the input and output ports. At each round-robin cycle, multiple projections can occur only in M output ports of the space-division cross-point switch fabric, i.e. every one port among M common input buffers. The output port locations in which the same set can be projected more than once at each round-robin cycle are fixed. So, the conditional round-robin selector can be implemented by simple logic gates because it is an M to 1 selector, not an N to 1 selector. The simple selector can support high-speed operations.

IV. Conclusion

We have proposed a new 3DRR scheduler for an advanced input queuing switch fabric which can be applied to both the high-speed ATM switch and the IP router. It takes into consideration complex parallel scheduling to eliminate buffer blocking effect that degrades the throughput of an input-queued switch. Our approach can be easily implemented in a single ASIC chip to control all of the distributed common input buffers and to configure the space-division cross-point switch. If the number of internal queues in a common input buffer is three times larger than the number of output ports, the performance of the advanced input queuing switch is comparable to the performance of an output queuing switch. Switch throughput reaches above 90% of an output queuing switch.

References

[1] G. Parulkar, D. Schmidt, and J. Turner, "AITPM: A Strategy for Integrating IP with ATM," in Proc. INFOCOM '96, March 1996.

[2] Y. Tamir and G. Frazier, "High performance multi-queue buffers for VLSI communication switches," in Proc. 15th Ann. Symp. on Comp. Arch., pp.343-354, June 1988.

[3] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch," in Proc. INFOCOM '96, March 1996.

[4] H. Obara, "Optimum architecture for input queuing ATM switches," IEE Electronic Letters, pp. 555-557, March 1991.

[5] H. Kondoh, H. Notani, H. Yamanaka, K. Higashitani, H. Saito, I. Hayashi, S. Kohama, Y. Matsuda, K. Oshima, and M. Nakaya, "A 622-Mb/s 8x8 ATM Switch Chip Set with Shared Multibuffer Architecture," IEEE J. Solid-State Circuits, vol. 28, no. 7, pp. 808-815, July 1993.