

Design of a High-Speed Overlapped Round Robin (ORR) Arbiter

Kenji Yoshigoe and Kenneth J. Christensen
 Computer Science and Engineering
 University of South Florida
 Tampa, FL 33620
 {kyoshigo, christen}@csee.usf.edu

Allen Roginsky
 IBM Corporation
 Research Triangle Park, NC 27709
 roginsky@us.ibm.com

Abstract

Round robin (RR) arbitration is commonly used for scheduling of cells in high-speed packet switches. In this paper, we present an overlapped RR (ORR) arbiter design that fully overlaps RR polling and cell scheduling. The ORR arbiter achieves 100% throughput even when a cell transfer time is less than a worst case polling, or scheduling, cycle. This is done by scheduling blocks of cells during a cell transfer time.

1. Introduction

Many switch architectures, including the well-known iSLIP and DRR, use round robin (RR) arbiters as part of their switch matrix scheduling. The RR/RR Combined Input Crossbar Queued (CICQ) switch [2] uses two levels of RR arbitration. Cells buffered in N queues at the input ports need to be scheduled for forwarding to an output link. A good switch scheduler should enable 100% throughput for all schedulable offered loads and be feasible to implement. Such a scheduler must be work conserving. Fast schedulers are needed to support ever-increasing switch size (port count) and link speed.

Most existing RR arbiters are based on a two-stage approach where scheduling is done simultaneously with cell transmission. That is, the cell forwarded in time slot t_i was scheduled in time slot t_{i-1} . In existing two-stage designs, 100% throughput can only be achieved if $N \cdot T_p < T_c$ for N number of queues, time T_p to poll a queue, and time T_c to service a non-empty queue (to forward a cell). Pipelined arbiters are not of a two-stage design. A pipeline-based concurrent round-robin dispatching scheme using multiple sub-schedulers was proposed in [1]. Each sub-scheduler provides a dispatching result in every P scheduling cycles where P is the number of sub-schedulers. We are interested in the case of $N \cdot T_p \geq T_c$. In this paper, we present the Overlapped Round Robin (ORR) arbiter that can achieve 100% throughput for this case.

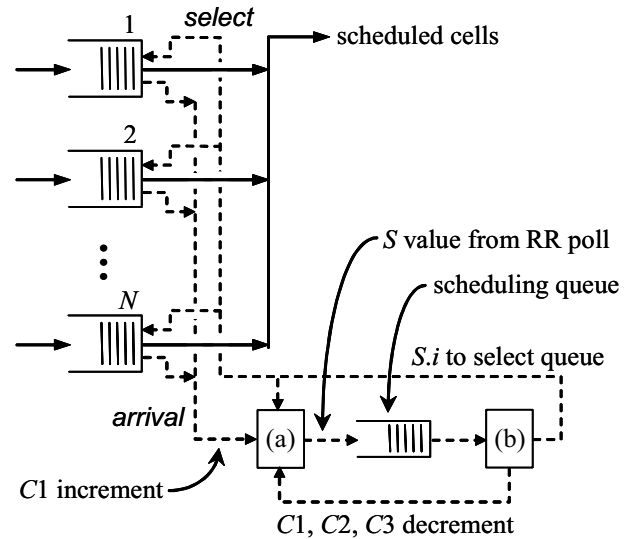


Figure 1. Cell queues and scheduling queue

2. Design of the Overlapped RR Arbiter

In a system of N queues, each queue has one control input (*select*) and one control output (*arrival*). Figure 1 shows the system of N queues, Q_i ($i = 1, 2, \dots, N$), with a (a) RR polling unit and (b) cell scheduling unit. The polling algorithm is shown in Figure 2 and the scheduling algorithm in Figure 3. A counter $C1_i$ is incremented on cell arrivals to Q_i and decremented on scheduled cell departures. The *arrival* output causes the increment of $C1_i$. The decrement of $C1_i$ is caused by the scheduling algorithm of Figure 3. The counter $C1_i$ represents the number of cells currently queued in Q_i . A counter $C2_i$ is decremented on cell departures from Q_i and is increased in the polling algorithm shown in Figure 2. The counter $C2_i$ represents the number of cells in a queue “marked” for forwarding. At all times, $C1_i - C2_i \geq 0$. The input *select* is used to select a queue for forwarding cells. Only one *select* line can be active on any given cell slot. A single counter $C3$ is also maintained representing the number of cells permitted to

This material is based upon work supported by the National Science Foundation under Grant No. 9875177 (Christensen and Yoshigoe).

1. do forever
2. $i = \text{mod}(i, N) + 1$
3. while $(C3 > K)$ wait
4. $\text{mark} = \min(K, C1_i - C2_i)$
5. if $(\text{mark} > 0)$
6. $C2_i = C2_i + \text{mark}$
7. $C3 = C3 + \text{mark}$
8. $S.i = i$
9. $S.m = \text{mark}$
10. queue S to the scheduling queue

Figure 2. Polling algorithm

1. do forever
2. if (the scheduling queue is non-empty)
3. $S = \text{dequeue}$ from scheduling queue
4. set *select* for queue $S.i$
5. do for $j = 1$ to $S.m$
6. wait for a cell to finish forwarding
7. decrement $C1_{S.i}$, $C2_{S.i}$, and $C3$
8. reset *select* for queue number $S.i$

Figure 3. Scheduling algorithm

be forwarded in the scheduling queue. All counters are stored in the polling unit. A constant value, K , is used in the polling unit. The counter $C3$ and the setting of K are described later..

The polling algorithm (Figure 2) “visits” each queue by testing if $C1_i - C2_i > 0$. If this holds, then there are unmarked cells in the queue. When a queue is visited and the *mark* value is non-zero, the counters $C2_i$ and $C3$ are updated and a scheduling value, S , comprising the queue index, i , concatenated with the number of cells marked in this visit, m , ($1 \leq m \leq K$) is queued in a special scheduling queue. The polling time T_p is incurred in lines 2 to 10 of the polling algorithm and in the time to increment $C1_i$. We use the notation $S.i$ and $S.m$ to mean the index value and marked cell count, respectively, for a given value of S . The value S is of size $\log_2(N) + \log_2(K)$ bits. Line 3 in the polling algorithm stops the polling if the value of $C3$ exceeds K . The counter $C3$ contains the sum of $S.m$ currently queued in the service queue. The stopping of polling in line 3 is essential for short-term fairness.

The scheduling algorithm (Figure 3) dequeues S from the scheduling queue when all currently scheduled cells have been forwarded. For example, if the currently dequeued scheduling value has $S.m$ equal to 3, then after 3 cell forwarding times the next queued scheduling value will be dequeued. The index $S.i$ is the queue to be issued a *select* for forwarding of $S.m$ cells. The polling and scheduling algorithms run concurrently.

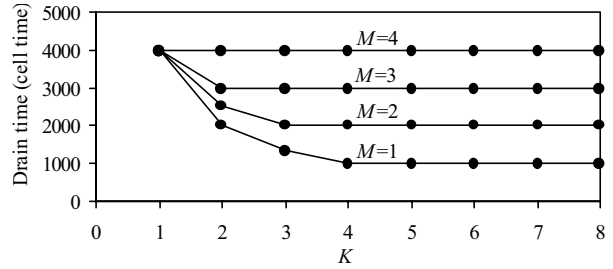


Figure 4. Work conservation results

Lemma. The smallest integer K needed for the ORR scheduling to achieve work conservation for all possible cases of queued cells in the N queues can be derived as, $K = \text{ceil}(N(T_p/T_c))$ where $\text{ceil}(\cdot)$ is the ceiling function.

Proof. Omitted due to space reasons.

Theorem. A new HOL cell at any queue of the ORR arbiter can be forwarded in less than $K \cdot (N - 1) + 2K + 1$ cell forwarding time.

Proof. Omitted due to space reasons.

3. Evaluation of the ORR Arbiter

Using simulation, we evaluated the ORR arbiter for fairness by measuring drain time from saturated queues. For $N = 4$, M queues were saturated with 1000 cells each. The value of M ranged from 1 to 4 and K ranged from 1 to 8. Figure 5 shows the results. All queues were entirely drained in $1000 \cdot M$ cell times for $K \geq 4$ for all M measured. This supports the K from the lemma. For an ideal RR (with $T_p = 0$), all queues are also entirely drained in $1000 \cdot M$ cell times.

4. Summary and Future Work

We have proposed the Overlapped Round Robin (ORR) arbiter that fully overlaps polling and cell scheduling. The ORR arbiter can be used to implement the RR arbitration in the RR/RR CICQ switch [2]. Future work includes the implementation of the ORR arbiter using an FPGA technology such as Xilinx.

References

- [1] E. Oki, R. Rojas-Cessa, and J. Chao, PCRRD: A Pipeline-Based Concurrent Round-Robin Dispatching Scheme for Clos-Network Switches,” *Proceedings of IEEE International Conference on Communications*, pp. 2121-2125, April 2002.
- [2] K. Yoshigoe and K. Christensen, “A Parallel-Polled Virtual Output Queued Switch with a Buffered Crossbar,” *Proceedings of IEEE Workshop on High Performance Switching and Routing*, pp. 271-275, May 2001.