BCB: A Buffered CrossBar Switch Fabric Utilizing Shared Memory

George Kornaros
Electronic and Computer Engineering Department (ECE),
Technical University of Crete,
GR 73100 Chania, Greece
kornaros@mhl.tuc.gr

Abstract

Todays' highly complex Systems-on-Chip require high-speed switching interconnects. Advances on signaling technology on the other hand allows the building of high-degree switching fabrics. Buffered crossbars are receiving increasing attention in these areas since they allow efficient high-performance distributed scheduling by decoupling inputs from outputs. However, the requirements in memory storage scale quadratically with the number of ports. An efficient architecture is presented to maximize system performance and memory utilization at the same time. The proposed solution is to integrate a shared memory buffer per row in addition to a minimum-sized dedicated buffer per crosspoint. An efficient novel architecture of an 8x8 crossbar switch is described, that is feasible and provides 100Gbps throughput implemented in a 0.18um CMOS technology.

1 Introduction

Crossbar switches available today are in two forms: arbitrated-unbuffered or synchronous crossbar and buffered or asynchronous crossbar. In an arbitrated crossbar cells are stored on the ingress side until required at the output, when a path is then scheduled through the central crossbar. Most crossbar-based routers today segment packets into fized size cells at the input line cards and reassembles them at the output. However, this segmentation of packets into cells can lead to degraded performance, since in the worst case this can effectively double the required bandwidth. To simplify the complex arbitration required to set up a path from ingress to egress, buffering can be added to the crossbar. In the buffered-crossbar architecture, arbitration is handled independently - *asynchronously* for ingress and egress. Thus, it is promising to achieve good performance without complex implementation. The system cost is significantly reduced since it is not subject to bandwidth fragmentation and no doubling of speedup is required (the crossbar speed usually is larger than the port speed by a factor ≥ 2 when coupling between inputs and outputs exist). The buffered crossbar architecture is suitable even for variable-size packets, when used in the context of IP networks, as introduced in [1] with the *segment-based* crossbars, and in [10] transferring packets in the form of fixeds-size *envelopes*. Moreover they can deliver performance guarantees as Turner proves in [15] if they are equipped with a moderate amount of internal buffer space.

The impact of crosspoint buffer size on performance and cost and the associated buffer utilization are two essential issues in the era of buffered crossbars. The ever increasing requirements for higher performance lead to multiport crossbars with large buffering evenly distributed to all crosspoints, thus making the control easier. However, most of the buffers are usually underutilized, unless there are conditions of heavy loads uniformly distributed. Consider also the waste of space in the case of variable sized packets segmented into cells. Obviously, in this way we consume a lot of silicon area, especially when scaling to many ports. If minimum buffer space is used then large throughput degradation can occur in conditions of unbalanced traffic or large RTTs ([2]). Under the assumption of uniform distribution of traffic we can relax the demand for buffer space, resulting in higher throughput. Kim shows in [3] that four-cell buffers are sufficient to reach good performance for short packets, while for long ones larger crosspoint buffers are required. The crosspoint buffering although beneficial to performance is proportional to uk^2 (k is the number of ports and u the link bandwidth) and dominates silicon area, especially when k scales to large numbers. Kim shows that in high radix routers increase in storage area exceeds wire area that is due to complexity.

Additionally to switching fixed-size cells, the buffer memory at a single crosspoint may even store a packet on the grounds that memory in a multi-million SoC has a low cost ([1]). However, unless segmentation and reassembly

(SAR) is used, the buffer needed per crosspoint, as shown in [1], is at least one maximum size packet plus one RTT of data. The cost is prohibitive even with modern memory technologies, especially when ”jumbo frames” are supported. Large switches, even though using todays’ state-of-the-art VLSI technology, come at cost, as is proved for a 32x32 switch that occupies $420mm^2$ in a 0.18um technology ([11]). To minimize the crosspoint buffer size Dong in [2] proposes sharing the crosspoint buffers among m inputs, such that long RTT can be supported by a combined input and crossbar queued (CICB) switch without decreasing switching performance and with no internal speedup. This solution could be seen as using *fat* channels to aggregate incoming links, or, if multiple inputs are handled together, then the advantages of distributed high-speed arbitration are lost.

Switch fabrics are increasingly employed in modern SoC designs as Networks-on-Chip (NoCs) or as backplanes, whose performance nowadays may scale from 40 to 160 to 320 Gbits/s. Nvidia, for instance, evolves the architecture of NV40/45 graphic processor by adding small and special purpose processing units that use switching interconnects (the G70 uses a 24 to 16 switch) to communicate among the array of quad processors, or to transfer data to and from the accelerators local memory ([12]). The Radeon X1800 uses a so-called Write Crossbar Switch, which is located around the Ring Bus memory controller proper for optimal memory access; it makes sure the requests are distributed evenly ([13]). Besides the graphic accelerator era, the Motorola’s C-5 network processor uses a set of 60Gbps busses to interconnect 16 processing units ([14]). In all these complex systems the buffer space utilization is deemed to be essentially a valuable resource. This direction of integrating high performance crossbar interconnects is arguably significant in future multiprocessor Systems-on-Chip.

The proposed innovative architecture addresses this issue in an efficient way. Both dedicated buffers per crosspoint and shared buffers per incoming port are used. The shared buffer space can be used by any cell with any destination, while when it is full the dedicated buffers can hold only one cell with a specific destination. Each crosspoint has a minimum buffer space in order to economize silicon area and on top of that to protect itself from the malicious behavior of other hungry flows (or bursty traffic patterns) that are absorbed inside the shared buffer space. The dedicated buffers also guarantee that flows destined to an outgoing port will not suffer from starvation. If this dedicated buffer is increased to hold even more cells, then the utilization will drop, unless the crossbar is guaranteed to operate at full rate and the traffic at the incoming and outgoing ports is uniformly distributed. If, on the other hand, the shared buffer is increased then, the complexity increases as well and the VoQs and arbitration of the incoming link cards is essentially performed in the fabric.

This switch uses shared memory in the crosspoints to reduce the total crosspoint buffer size such that better buffer utilization is achieved and flows with high data rates can be handled with smaller amounts of memory than a switch with dedicated buffers. Moreover, it supports in-order delivery of cells despite the shared buffering. This is in favor of high-speed interconnects that need to maintain the sequence of possibly pipelined transfers.

The paper is organized as follows. Section 2 describes the overall system architecture explaining the key features. Section 3 gives performance results studying when shared buffering is beneficial to the system. Finally, Section 4 presents the implementation results.

2 Architecture

The proposed switch architecture is based on a eight-by-eight buffered crossbar, called BCB hereafter. It can be used as a building module to construct multi-plane, multi-stage switches, and moreover, it is capable to provide enough throughput even as a single-stage fabric, and alleviate larger latencies at the same time.

Chao ([9]) and Kim ([3]) use either multi-stage fabrics, or hierarchical organizations to scale to high-performance fabrics and save on area. With small subswitches the control requirements are relaxed, resulting in higher performance. The throughput however still depends on the amount of intermediate buffering. These architectures also incur latency for a cell to traverse all the hops; on the contrary, a cell in BCB faces 2 clock cycles of latency when no other cell is ready for transmission.

Each port of BCB is 64-bit wide, it accepts and forwards fixed-size cells of 8 64-bit words each and uses backpressure to flow-control the incoming traffic. These basic features and its capacity to support packet-based environments will be addressed in the next section. In addition, it may provide extensions that address requirements that modern fabrics face, such as peer-to-peer support, QoS, multicasting, and support for multi-protocol encapsulation.

The BCB system architecture is shown in figure 1. It can be considered as a stack of slices, where each slice consists of the buffers, both dedicated and shared, and the corresponding logic per input port. The input fifos, organized usually as virtual output queues, and the associated arbiter that selects the cells of the flows that traverse the fabric reside on a line-card and are not included in the figure. The figure shows a closer look at one slice, while a round-robin arbiter at each output port is not outlined here. Hence, the BCB fabric consists of a set of slices, one per input port and one arbiter per output port, designed in a modular manner offerring itself for automatic generation and customization to meet the desired features.

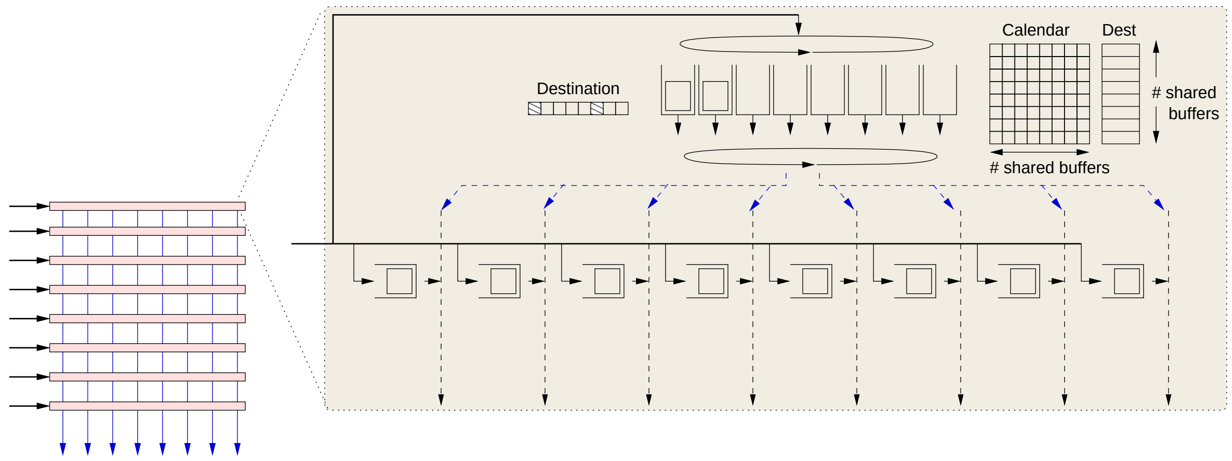

Figure 1. The 8x8 shared buffered crossbar, internal organization of each row.

Each cell that enters an horizontal slice may be buffered either in the dedicated area or in the shared space as figure 1 shows. The horizontal block provides an 8-bit feedback signal as a credit indication, that depicts if there is buffer space to store a cell for an output port. Actually, each bit is a combined piece of information; it depends on the status of the dedicated buffer and on the status of the shared buffer. Thus, the internal memory occupancy status is transparent to the in-port arbiter (not shown in the figure). This line-card arbiter is free to apply any selection policy to the cells waiting to be forwarded to the crossbar. It is flow controlled by an 8-bit signal indicating if the respective output port is free to receive a cell.

The output arbiter at each vertical output link selects the cell to depart following a round-robin discipline. One horizontal slice provides the combined status, empty or full of the internal buffers, both dedicated and shared, to the output scheduler. This occurs as soon as a cell is accepted in either one kind of buffers, immediately after its first word has arrived. Hence, if it is selected by the out arbiter it may depart in a cut-through fashion. For this purpose dual-ported memory elements are employed to support two accesses at the same clock cycle. The latency of a cell passing through the fabric is thus the arbitration delay plus the delay of the other cells that must be served in the vertical direction, if they have arrived earlier.

Whenever a cell is enabled to leave its status must be updated. This is a one cycle operation. However, in the worst case all eight cells that are stored in one horizontal slice structure may become eligible in the same cycle. In the following eight cycles all the status updates will be completed; the dequeue operations are overlapped in the same time interval causing no waste of clock cycles. Consequently, the maximum number of cell positions in the shared memory space should be at most equal to the cell size and to the number of out ports. If the BCB integrates more than eight outgoing ports then, then, in the worst case more than eight updates must be performed in one cell time. If the input port is guaranteed not to provide a new arrival by using backpressure for instance, or increase the cell's size to more than eight words, then even this scenario is feasible.

One challenging task is to maintain cells' sequence, since they reside in a shared memory without fifo support, and they are delivered over multiple paths. This is one fundamental property commonly desired by crossbar fabrics, that is to be *order-preserving*. The arrival time of each cell is recorded and it is taken into consideration when the next eligible cell must be chosen to depart. Each cell that resides in the shared buffer besides its arrival time is also linked with a field that denotes the destination port. The cells with the same destination mask that reside in the shared slots and in the dedicated slot are competing to leave based on their age. A more complex and larger system than this, in terms of respecting the arrival time of a few requests, is detailed in [7], and manages to exhibit high performance based on CAM techniques.

3 Performance

Cycle accurate simulations are used to measure the performance of the switch in terms of latency, throughput and buffer utilization. The cells are injected using a Bernoulli process to create a uniform random traffic. The figures below, 2 and 3, plot the simulation results of the vhdl model of the 8x8 buffered crosspoint switch for two different configurations: a shared buffer of 4 cells per row and, a shared buffer of 8 cells per row. The crossbar faces very low latency at low offered loads, which is due to decoupling of input and output arbitration and cut-through feature of BCB. As the input traffic increases this design achieves a saturation throughput of 100% of capacity. Apparently, using a shared buffer of larger capacity the delay is affected when the input load increases, since a larger number of cells are waiting; in the configuration of 4-cells shared buffer the segments are not allowed in the first place to enter the system and thus this delay is not measured and depicted in the plot.

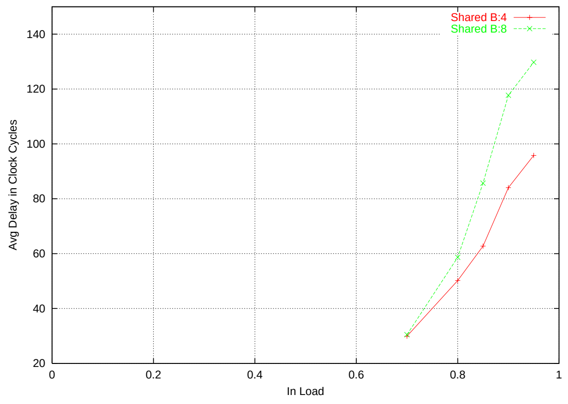

Figure 2. Average delay after simulation for 80K cells evenly distributed over the 8 ports.

The occupancy of the shared buffer is poor at light input loads as one can observe in the second plot. This is expected since the dedicated buffers are used first and when they are full then the shared ones are filled. When the injected traffic exceeds 70% then the buffer utilization increases exponentially.

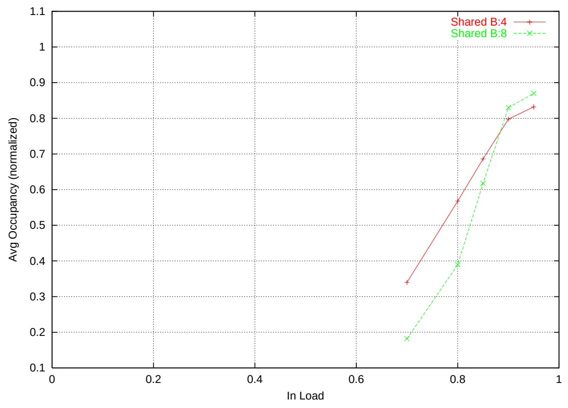

Figure 3. Average occupancy, normalized to the shared buffering degree. Simulation for 80K cells evenly distributed over the 8 ports.

The BCB switch is also tested using unbalanced traffic. The offered load in extensive simulations is based on probability *w*, which is the fraction of input load directed to a single-predetermined output, while the rest of the incoming load is uniformly distributed to all outputs. When the *w* is low (10%), that is, we have uniform traffic then the shared buffer utilization is not very high, even when injecting up to 80% traffic. On the contrary, when *w* increases to 30% then the buffer occupancy rises to 75% at average, and the occupancy variance increases. Figure 4 shows that in this situation it is significant to have more than 4 buffers, to handle unbalanced loads. Finally, when *w* approaches 1 the traffic is completely directional and the occupancy drops to less than 20%.

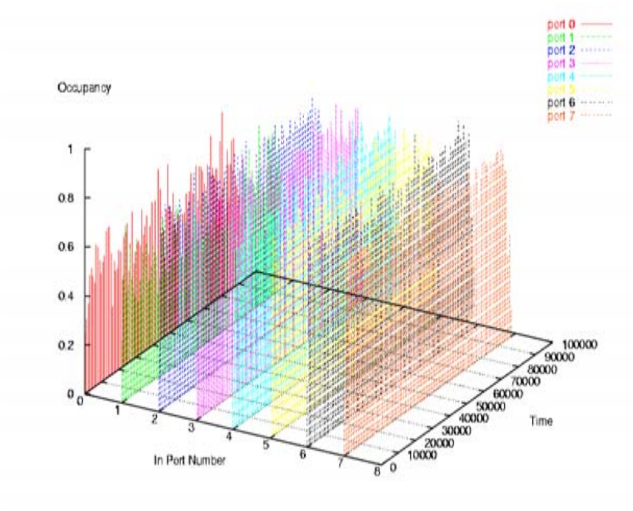

Figure 4. Average occupancy per port under unbalanced traffic. Simulation for 80K cells evenly distributed over the 8 ports.

4 Implementation

The vhdl model of the BCB switch was synthesized and the following table 1 summarizes the results. A standard 0.18 um CMOS technology by UMC is used to obtain indicative results. The first line shows that the timing critical path is dominated by the logic in one horizontal slice and not by the column arbiters, that are not included. This clock cycle is due to low latency which is the main objective in the design. Note that the large amount of memory dominates the cost in terms of area. One memory cut of 8 64-bit words occupies 0.0278 mm^2 of silicon; thus in the first case when a shared buffer for 4 cells is used per row, 49 Kbits occupy 2.67 mm^2, while in the second configuration we need 3.56 mm^2.

The critical path is to find the cell with the longest waiting time in one shared buffer and not the round-robin arbi-

[3]Compiled SRAM: two-port memory blocks

Block	Cycle time(ns)	Area (mm^2)	Mem *Kbits*[1]
row_8x8_s8	4.7	0.85	8.2
bcb_8x8_s4	4.59	4.86	49.15
bcb_8x8_s8	5.1	7.0	65.5

Table 1. Performance and cost of the crossbar block.

tration of the column arbiter. Fast arbiters that are popular have already been used in [5], [6] and recently also appear in combined input and crossbar queued (CICQ) switches in [8].

The bcb switch configured with a 4-cell shared buffer per row was placed and routed to obtain more accurate results. Although the total standard cell area is 4.86 mm^2, achieving a density of 92% is feasible as figure 5 shows. The final total silicon area is 5.25 mm^2 and the clock skew is 120 ps (instead of 300 as set in constraints).

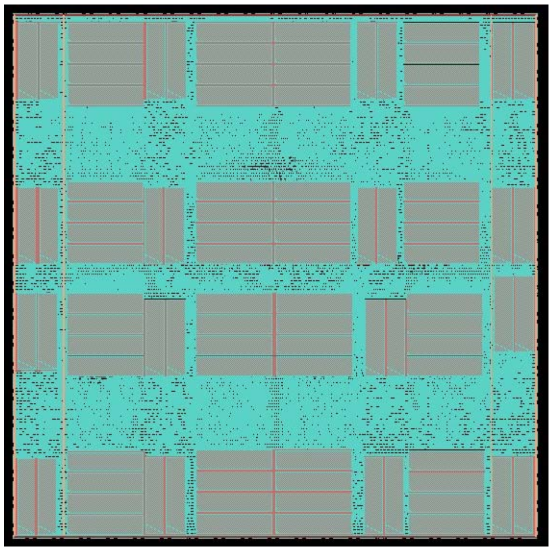

Figure 5. The core layout of the BCB using a shared buffer of 4 cells. Each side is almost 2.3 mm.

The BCB switch is also implemented using reconfigurable technology in a VirtexII-Pro in order to build a real prototype. Since the user pins of xc2vp70-6-ff1704 were not enough for the 8x8 BCB switch, ddr interface techniques are employed. Thus, each port uses a 32-bit data interface running at double core speed and translates the 32 bit datapath to 64 at the input and back to 32 at the outgoing path. The design after placement and routing occupies 21.5K slices (64% of the device) 192 brams (58%) and the core speed is 16ns clock period, which translates to 32Gbps bandwidth.

Future work includes integration of a weighted-round-robin (WRR/WFQ) arbiter, which has already been designed to attach either to one input or to the output port. It features on-chip queuing of cells, configurable to accommodate 8, 16, or 32 8-word cells. Using the device xc2vp70-6-ff1704 (speed grade 6) this small arbiter occupies 745 slices and 6 brams, while achieving maximum frequency of 141.16 MHz. Although it cannot store large amounts of data, it is challenging to integrate such small schedulers with no requirements for off-chip memories, to examine QoS over the BCB architecture. A simulation study exploring the ways to interface and flow control such incoming port managers and the performance guarantees they provide would be of considerable practical value.

Currently, this WRR/WFQ arbiter is unaware of the status of the shared buffer inside each slice. However, additionally to the arbiter's weights, it is easy to keep account of the BCB buffer status, or give priority to serving the cell in the dedicated buffers against the cells in the shared ones if priority must be guaranteed for QoS differentiation.

5 Conclusions

The performance benefits of a buffered crossbar switch come at the expense of a large silicon area especially when it is used as interconnection network of multiprocessing Systems-on-Chip. An efficient architecture is presented to reduce storage requirements, and offer better memory utilization. In addition to minimum-size dedicated buffers per crosspoint, a shared buffer is used per row featuring in-order forwarding of fixed-size cells. In particular, this architecture is possible to provide superior performance in situtations of bursty traffic and in variable size packet-based environments. A feasibility study turned out with an ASIC implementation of the large fabric version, using an 8-cell shared buffer per row, which occupies 7 mm^2 using a standard 0.18 um CMOS technology by UMC and offers an 8x8 switch with almost 100Gbps throughput.

References

[1] M. Katevenis, G. Passas, D. Simos, I. Papaefstathiou, N. Chrysos. ”Variable Packet Size Buffered Crossbar (CICQ) Switches”, *Proc. IEEE International Conference on Communications (ICC 2004)*, Paris, 20-24 Jun 2004, vol. 2, pp. 1090-1096.

[2] Z.Dong & R.Rojas-Cessa. ”Long Round-Trip Time Support with Shared-Memory Crosspoint Buffered Packet Switch”, *IEEE 13th Annual Symposium on High Performance Interconnects*, Aug 2005.

[3] J.Kim, W.Dally, B.Towles & A.Gupta, ”Microarchitecture of a High-Radix Router”, *ISCA'05*

[4] R.Rojas-Cessa, E.Oki, & H.Jonathan Chao, ”CIXOB-k: Combined Input-Crosspoint-Output Buffered Switch”, *Proceedings GLOBECOM’01*, vol.4, pp.2654-2660, Dallas, Nov. 2001.

[5] G.Kornaros et al ”Pipelined Multi-Queue Management in a VLSI ATM Switch Chip with Credit-Based Flow-Control”, *17th Conference on Advanced Research in VLSI*, University of Michigan, Ann Arbor, MI, USA, Sep 15-16, 1997.

[6] G. Kornaros, F. Orphanoudakis, I. Papaefstathiou, ”Active Flow Identifiers for scalable, QoS scheduling in 10-Gbps network processors”, *IEEE International Symposium on Circuits and Systems (ISCAS 2003)*, Bangkok, Thailand, May 25-28, 2003.

[7] G. Kornaros et al ”An Efficient Implementation of Fair load Balancing over Multi-CPU SoC Architectures”, *DSD’2003 Euromicro Symposioum on Digital System Design*, Antalya, Turkey, Sep 3-5, 2003.

[8] K. Yoshigoe, K. Christensen, & A. Jacob, ”The RR/RR CICQ Switch: Hardware Design for 10-Gbps Link Speed”, *Proceedings of the IEEE International Performance, Computing, and Communications Conference*, pp. 481-485, Apr 2003.

[9] H.J.Chao, J.S.Park, S.Artan, S.Jiang & G.Zhang ”True-Way: A Highly Scalable Multi-Plane Multi-Stage Buffered Packet Switch” *IEEE Workshop on High Performance Switching and Routing*, Hong Kong, May 2005.

[10] K.Kar, T.V.Lakshman, S.Stiliadis & L.Tassiulas ”Reduced Complexity Input Buffered Switches” *Proceedings of HOT Interconnects VIII*, Stanford University, Stanford, CA, Aug 2000.

[11] D.Simos, ”Design of a 32x32 Variable-Packet-Size Buffered Crossbar Switch Chip”, Tech.Report FORTH-ICS/TR-339, M.Sc.Thesis, Univ.of Crete, Jul 2004.

[12] A.Medvedev, ”NVIDIA GeForce 7800 GTX 256MB PCI-E - Theory and Architecture” www.digit-life.com, Jun 22, 2005.

[13] A.Stepin, Y.Lyssenko, A.Shilov, ”ATI RADEON X1800 XT and XL Performance: Crushing NVIDIA’s 7800?” www.xbitlabs.com, Oct 6, 2005.

[14] Motorola, C-5 Data-sheet: http://e-www.motorola.com/webapp/sps/site/ /taxonomy.jsp?nodeId=01DFTQ3126q62S

[15] J Turner, ”Strong Performance Guarantees for Asynchronous Crossbar Schedulers” *Proc. IEEE International Conference on Communications (Infocom 2006)*, Barcelona, 23-29 Apr 2006.