Accelerated Packet Placement Architecture for Parallel Shared Memory Routers

Brad Matthews[1], Itamar Elhanany[1], and Vahid Tabatabaee[2]

[1] Department of Electrical and Computer Engineering, The University of Tennessee.
[2] Institute for Advanced Computer Studies, The University of Maryland.

Abstract. Parallel shared memory (PSM) routers represent an architectural approach for addressing the high memory bandwidth requirements dictated by output-queued switches. A fundamental related challenge pertains to the design of the high-speed memory management algorithm which is responsible for placing arriving packets into non-conflicting memories. In previous work, we have extended PSM results by introducing the concept of Fabric on a Chip (FoC). The latter advocates the consolidation of core packet switching functions on a single chip. This paper further develops the underlying technology for high-capacity FoC designs by incorporating a speedup factor coupled with a multiple packet placement process. This yields a substantial reduction in the overall memory requirements, paving the way for the implementation of large scale FoCs. We further provide analysis for establishing an upper bound on the sufficient number of memories along with a description of an 80Gbps switch implementation on an Altera Stratix II FPGA.

1 Introduction

Recent years have witnessed unprecedented advances in the design, verification formalism, and deployment of high capacity, high performance packet switching fabrics. Such fabrics are commonly employed as fundamental building blocks in data networking platforms that span a wide variety of application spaces. Local and Metro area network platforms, for example, host fabrics that typically support up to hundreds of gigabits/sec. However, switching fabrics are not limited to Internet transport equipment. Storage area networks (SANs) often necessitate large packet switching engines that enable vast amounts of data to traverse a fabric, whereby data segments flow from users to storage devices, and vice versa.

The switching capacity of an Internet router is often dictated by the memory bandwidth required to buffer arriving packets. With the demand for greater capacity and improved service provisioning, inherent memory bandwidth limitations were encountered rendering input queued (IQ) [1] switches and combined input and output queued (CIOQ) architectures more practical. Output-queued (OQ) switches, on the other hand, offer several highly desirable performance characteristics, including minimal average packet delay, controllable Quality of Service (QoS) provisioning, and work-conservation under any admissible traffic conditions [2]. However, the memory bandwidth of such systems is $O(NR)$, where N denotes the number of ports and R the data rate of each port. Clearly, for high port densities and data rates, this constraint dramatically limits the scalability of the switch.

In relation to standard switching architectures, the Fabric on a Chip (FoC) approach seeks to exploit the recent improvements in the fabrication of VLSI circuitry in order to consolidate many switching functions on a single silicon die. Advances in packaging technology now make it possible for large amounts of information to simultaneously be forwarded to a single chip, which was not possible several years ago. There are several key advantages that are attributed to the concept of FoC. First, it eliminates the need for virtual output queueing (VOQ) [3] as well as some output buffering associated with standard switch architectures. Second, by exploiting the ability to access the multiple on-chip Mbits of dual-port SRAM, packets can be internally stored and switched without the need for external memory devices. The crosspoint switches and scheduler, pivotal components in input-queued switches, are avoided thereby substantially reducing chip count and power consumption. Third, much of the signaling and control information that typically spans multiple chips can be carried out on a single chip. Finally, the switch management and monitoring functions can be centralized since all the information is available at a single location.

In an effort to retain the desirable attributes of output-queued switches, while significantly reducing the memory bandwidth requirements of shared memory architectures, such as the parallel shared memory (PSM) switch/ router, have recently received much attention [4]. PSM utilizes a pool of slow-running memory units operating in parallel. At the core of the PSM architecture is a memory management algorithm that determines, for each arriving packet, the memory unit in which it will be placed. This paper extends previous work by the authors [5] on the design of large-scale PSM switches, from a single-chip realization perspective. By introducing computation and memory speedup components, a more efficient high-speed memory management algorithm is attained, yielding higher system scalability.

The rest of the paper is structured as follows. In Section II an overview of parallel shared memory switch architectures is provided from an FoC standpoint. Section III describes the proposed switch architecture and memory management algorithm. Section IV offers a detailed analysis establishing an upper bound on the sufficient number of parallel memories required. In Section V the hardware architecture and FPGA-based simulation results are described, while in Section VI the conclusions are drawn.

2 Switch Fabric on a Chip

Initial work has indicated that, assuming each of the shared memory units can perform at most one packet-read or -write operation during each time slot, a sufficient number of memories needed for a PSM switch to emulate a FCFS OQ switch is $K = 3N - 1$ [4]. The latter can be proven by using constraint sets analysis (also known as the "pigeon hole" principle), summarized as follows. An arriving packet must always be placed in a memory unit that is currently not being read from by any output port. Since there are N output ports, this first condition dictates at least N memory units are available. In addition, no arriving packet may be placed in a memory unit that contains a packet with the same departure time. This results in additional $N-1$ memory units representing the $N-1$ packets having the same departure time as the arriving packet, that may have already been placed in the memory units. Should this condition not be satisfied, two packets will be required to simultaneously depart from a memory unit that can only produce one packet in each time slot. The third and last condition states that all N arriving packets must be placed in different memory units (since each memory can only perform one write operation). By aggregating these three conditions, it is shown that at least $3N-1$ memory units must exist in order to guarantee FCFS output queueing emulation. Although this limit on the number of memories is sufficient, it has not been shown to be necessary. In fact, a tighter bound was recently found, suggesting that at least $2.25N$ memories are necessary [6]. Regardless of the precise minimal number of memories used, a key challenge relates to the practical realization of the memory management mechanism, *i.e.* the process that determines the memories in which arriving packet are placed. Observably, the above memory-management algorithm requires $O(N)$ iterations to complete.

In [7],[8] Prakash, Sharif, and Aziz proposed the Switch-Memory-Switch (SMS) architecture, which is a variation on the PSM switch, as an abstraction of the M-series Internet core routers from Juniper. The approach consists of statistically matching input ports to memories, based on an iterative algorithm that statistically converges in $O(logN)$ time. However, in this scheme, each iteration comprises multiple operations of selecting a single element from a binary vector. Although the nodes operate concurrently from an implementation perspective, these algorithms are $O(log^2N)$ at best (assuming $O(logN)$ operations are needed for each binary iteration as stated above). Since timing is a critical issue, the computational complexity should directly reflect the intricacy of the digital circuitry involved, as opposed to the high-level algorithmic perspective.

To address the placement complexity issue, in prior work we proposed a pipelined memory management algorithm that reduced the computational complexity of placing a packet in a buffer to $O\left(1\right)$. The subsequent cost associated with reducing the placement complexity is an increase in the number of required parallel memories to $O\left(N^{1.5}\right)$ and a fixed processing latency. The justification resides in the newfound ability to store and switch packets on chip as multiple megabits of dual-port SRAM are now available. Furthermore, it is now plausible to consider that all data packets can arrive at a FoC directly, thus eliminating the need for virtual output queueing [3] as well as some of the output buffering common employed by existing router designs. The elimination of crosspoint switches and scheduler, as found in IQ switches, provides an important reduction in chip count and power consumption. In achieving a greater degree of integration from such consolidation, a substantial reduction in overall resource consumption is expected.

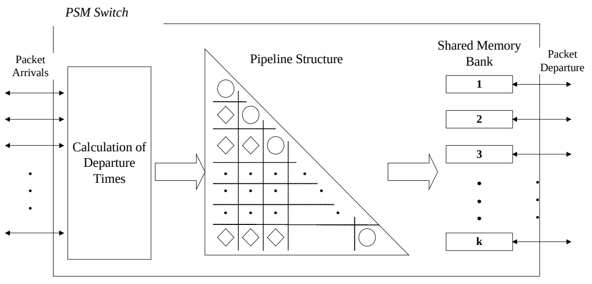

Fig. 1. General architecture of the proposed parallel shared memory (PSM) switch. Arriving packets are placed in a set of $(k > N)$ memory units.

In extending the architecture to allow for speedup and multiple packet placements, we enable more than one packet decision and placement to occur during a single packet time. This helps reduce the number of total required parallel memories, as discussed in the next section.

3 Packet Placement Algorithm

3.1 Switch Architecture

We begin with a detailed description of the proposed PSM switch structure, depicted in Figure 1. The most significant component in the architecture is the pipelined memory-management algorithm. A departure time is calculated for each packet, prior to the insertion of packets into the memory management subsystem. This process is governed by the output scheduling algorithm employed, and is generally very fast. The most straightforward scheduler is first-come-first-serve (FCFS), in which packets are assigned departure times in accordance with their arrival order. To provide delay and rate guarantees, more sophisticated schedulers [2] can be incorporated which is reflected by the departure time assignments. The main contribution of this paper resides in the memory management algorithm that distributes the packet-placement process, at a cost of fixed latency. This is achieved by utilizing a multi-stage pipeline, as illustrated in Figure 2.

The pipeline architecture consists of $\frac{L(L+1)}{2}$ cell buffering units arranged in a triangular structure, where L denotes the number of parallel memory units. Each row is therefore associated with one memory unit. The notion of speedup, s, is introduced with the requirement that the pipeline operate s times faster than the line rate. One desired benefit of operating the pipeline at a higher rate is reduced latency. Moreover, if the incoming packets from the set of N input ports are presented to the pipeline in groups of $\frac{N}{s}$, the number of conflicts from packets with the same arrival time is reduced from N to $\frac{N}{s}$.

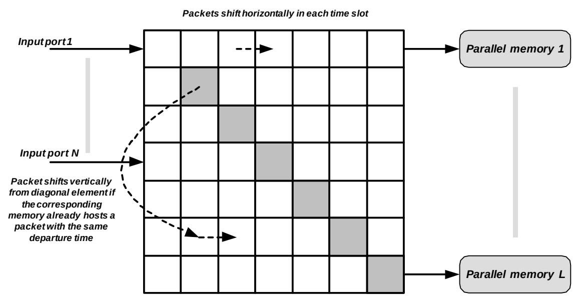
Packets shift horizontally in each time slot
Input port 1
Input port N
Packet shifts vertically from diagonal element if the corresponding memory already hosts a packet with the same departure time
Parallel memory 1
Parallel memory L

Fig. 2. Illustration of the memory management pipeline structure

Incoming packets from input port i are initially inserted into row $i \bmod \left(\frac{N}{s}\right)$. The underlying mechanism is that at every time slot, packets are horizontally shifted one step to the right, with the exception of the diagonal cells. A packet residing in a diagonal cell is either shifted (moved) vertically to another row in the same column or placed in the memory associated with the row in which it resides. Vertical packet shifts occur if the memory associated with the row in which the packet resides contains another packet with the same departure time.

If a vertical shift is to be performed, the diagonal cell must select a row in its column that satisfies the following three conditions: (1) the pipeline cell of the row selected does not already contain a packet; (2) the memory in the row selected must not contain a packet with the same departure time; (3) all of the pipeline cells located in the selected row, regardless of their column, must not contain a packet with the same departure time. Applying these constraints, vertical moves provide a mechanism for resolving memory placement contentions. The goal of the scheme is that once a packet reaches a diagonal cell in the pipeline it has exclusive access to the memory located in its row. If the current row memory is occupied, an attempt is made to place the packet in a row for which there are no existing conflicts.

Placement decisions along the diagonal are made concurrently and independently as means of maximizing the processing speed of the system. As selections are independently made for each packet, it is possible for packets along the diagonal to simultaneously select the same row. In a single packet placement scheme, there exists only one memory location in a row for any given departure time. To reduce the number of conflicts associated with packets simultaneously selecting the same row, the number of memory locations in a row for a given departure time can be increased to $m > 1$. As multiple packet placements to a single memory are now allowed, we must guarantee that m packets can be read from memory during a single packet time. One might speculate that the pipeline speedup, s, and the number of placements allowed, m, which is effectively the memory read rate, must be equal. This is generally not required, since it might be necessary to operate the pipeline at a slower rate as dictated by potentially faster on-chip SRAM resources. In this case, it is still prudent to offer additional placement locations in order to reduce conflict.

In provisioning m packet placement locations for each memory, it would appear that the reduction in row memories is merely an inconsequential outcome of increasing the memory depth. Recognizing that as packets shift vertically from block b to $b+1$, the block size, in terms of physical rows, decreases as s and m increase. This infers that a vertical movement bypasses fewer potentially acceptable rows with each subsequent placement. In subsequent sections, we provide analysis that derives optimal values for these parameters. In order to illustrate the underlying memory-management principal, we refer to the following example.

3.2 Upper Bound on the Sufficient Number of Memories

In this section, we obtain an upper bound on the number of memories sufficient for the pipelined memory management architecture, given a speedup factor, s. Let us view the pipeline rows as arranged in $\mathcal{B}$ sequential blocks. Speedup is introduced into the system through the partition of the N arriving packets into s distinct segments. Packets that arrive at time t to any of the N ports are presented to the first block which consists of $\frac{N}{s}$ rows. Arriving packets are then multiplexed and written to one of the $\frac{N}{s}$ rows in this first block. Once placed in a row, a packet can only be written to one of m memory locations for a given departure time, or shift vertically to another row in block $b+1$.

Lemma 1. *There should be at least* $\left(\frac{s+m}{sm}\right)N - b$ *rows in block* b*, for* $b \in [2, 3, ...\mathcal{B}]$

Proof. Consider a packet moving from block b to $b+1$. For a system with speedup s, it will find at most $\frac{N}{s} - 1$ packets having the same arrival time. Furthermore, there are at most $N - bm - 1$ packets with the same arrival time, since at least, bm packets with the same arrival time are served inthe first b blocks. Therefore, in block $b+1$, there are at most $\frac{N}{s} - 1$ rows occupied with packets with the same arrival time. Since up to m packets with the same departure time can be served with one memory, we need $\frac{N-bm}{m}$ additional rows for packets with the same departure time. Hence, we need $\frac{N}{s} - 1 + \frac{N-bm}{m}$ rows for block $b+1$, or $\left(\frac{s+m}{sm}\right)N - b$ rows for block $b \in [2, 3, ...\mathcal{B}]$.

For a switch with N ports and P blocks, the total number of rows (parallel memories) can be expressed as:

$$\begin{aligned} L(N) &= \frac{N}{s} + \left(\left(\frac{s+m}{sm}\right)N - 2\right) + .. \\ &\quad + \left(\left(\frac{s+m}{sm}\right)N - \mathcal{B}\right) \\ &= \frac{N}{s} + N\left(\frac{s+m}{sm}\right)(\mathcal{B}-1) - \end{aligned} \quad (1)$$

$$(\mathcal{B}+2)(\mathcal{B}-1)/2$$

$$= N\frac{((s+m)(\mathcal{B}-1)+m)}{sm} -$$

$$(\mathcal{B}+2)(\mathcal{B}-1)/2 \tag{2}$$

To compute the total number of rows (or memory units), we must determine the maximum number of $\mathcal{B}(N)$ blocks, or vertical shifts, required to successfully assign all packets to memory.

Lemma 2. *The maximum number of packets with the same departure time in the fourth block is* $P_4 \leq \left(\sqrt{N}-m\right)^2$.

Suppose there are P_1 packets with the same departure time in the first block. Recall that there can be no more than $\frac{N}{s}$ packets with the same arrival time in the first block and no more than N packets with the same departure time in the system, such that $P_1 \leq N$. Packets only move vertically from the first block if a given packet resides in a row that contains m other packets with the same departure time. Let us state that there are P_1 packets residing in R_1 $\left(R_1 \leq \frac{N}{s}\right)$ rows of the first block, then the number of packets that propagate vertically to the second block must equal the number of conflicting packets given by

$$P_3 = P_1 - mR_1 \tag{3}$$

Decisions regarding which row destination for a given packet are made independently such that packets with the same departure time can shift simultaneously to the same row. Note that a maximum of R_1 packets can shift simultaneously such that the resulting number of rows with conflicts in the second block is given by

$$R_2 \geq \left\lfloor \frac{P_1 - mR_1}{R_1} \right\rfloor \tag{4}$$

The value of R_2 represents the number of unique rows that received packets, with the same departure time, from the first block. Applying these same principles, we can further state the maximum number of packets with the same departure time that can shift to the third block block is given by

$$P_3 = P_2 - mR_2 \leq P_1 - mR_1 - m\left(\left\lfloor \frac{P - mR_1}{R_1} \right\rfloor\right) \tag{5}$$

If $P_1 - mR_1$ is divisible by R_1, then

$$P_3 \leq P_1\left(1 - \frac{m}{R_1}\right) - mR_1 + m^2 \tag{6}$$

otherwise, since $P_4 \leq P_3 - 1$, we have

$$P_3 \leq P_1\left(1 - \frac{m}{R_1}\right) - mR_1 + m^2 + 1$$

$$P_3 \leq P_1\left(1 - \frac{m}{R_1}\right) - mR_1 + m^2 \tag{7}$$

The maximum value of (7) is obtained when $R_1 = \sqrt{P_1}$. Substituting $P_1 = N$, yields the following inequality

$$P_4 \leq \left(\sqrt{N} - m\right)^2 \tag{8}$$

Note that if N is a complete square we have,

$$P_3 \leq \left(\sqrt{N} - m\right)^2 . \tag{9}$$

Corollary 1. *A sufficient number of parallel memory blocks required for an NxN switch, employing the proposed architecture, is $O\left(\sqrt{N}\right)$*

Proof. Equation (8) shows that for an N-port switch, the maximum number of conflicting packets with the same departure time in the fourth block is $\left(\sqrt{N} - 1\right)^2$. Let $\mathcal{B}\left(N\right)$ represent the number of stages required for an N-port switch. We can thus express the total number of stages required using the recursive relationship,

$$\begin{aligned} \mathcal{B}\left(N\right) &= 1 \\ &\vdots \\ \mathcal{B}\left(N\right) &= \mathcal{B}\left(N - 2\sqrt{N} + 1\right) + 3 \end{aligned}$$

from which we conclude that $\mathcal{B}\left(N\right) = O\left(\sqrt{N}\right)$.

Theorem 1. *For an $N = k^2$ $k \in \{1, 2, ...\}$ and $s = m$, the number of memories is*

$$\begin{aligned} L\left(N\right) \leq \frac{4k^3}{m^2} + \left(\frac{3}{m} - \frac{6}{m^2}\right)k^2 - \left(\frac{5}{m} - \frac{4}{m^2}\right)k \\ - \left(2 - \frac{5}{m} + \frac{2}{m^2}\right) \end{aligned} \tag{10}$$

with equality if $N = k^2$

Proof. We prove the equality for $N = k^2$, suggesting that the general case trivially follows. We first show by strong induction that the number of required row blocks are

$$\mathcal{B}\left(k^2\right) \leq \frac{2k}{m} + \left(2 - \frac{2}{m}\right) \tag{11}$$

For $k = 1$, the result is trivial. In order to prove it for $k \leq m$, it is sufficient to show $\mathcal{B}\left(m^2\right) = 2$ To that end, for $k = m$, notice the number of rows in the first block is,

$$R_1 = \frac{N}{s} = \frac{k^2}{s} = \frac{m^2}{m} = m \tag{12}$$

Therefore, the maximum number of packets that can move simultaneously to the same row in the second block is m (one packet from each row in the first block). Since each memory can serve up to m packets with same departure time, all packets in the second block rows can be scheduled and there is no need to have third block rows. So far, we have proved the result for $k = 1, \ldots, m$. Next we use, the strong induction step to prove it for $k > m$. We assume it is true for $1, \ldots, k$ $\{k \geq m\}$ and prove it for $k+1$. For $N = (k+1)^2$, using lemma 2 and (9) (given that N is a complete square), we have

$$\begin{aligned} \mathcal{B}\left((k+1)^2\right) &\leq \mathcal{B}\left((k+1-m)^2\right) + 2 \\ &\leq \frac{2\,(k+1-m)}{m} + 2 - \frac{2}{m} + 2 \\ &= \frac{2\,(k+1)}{m} + \left(2 - \frac{2}{m}\right) \end{aligned} \tag{13}$$

Now, we let s=m, then substitute (13) into (1) to obtain,

$$\begin{aligned} L\left(k^2\right) &= k^2 \frac{2\,(\mathcal{B}-1)}{m} - \frac{(\mathcal{B}+2)\,(\mathcal{B}-1)}{2} \\ &\leq k^2 \frac{2\left(2\frac{k}{m} - \frac{2}{m} + 1\right) + 1}{m} \\ &\quad - \frac{\left(\frac{2k}{m} + 4 - \frac{2}{m}\right)\left(\frac{2k}{m} + 1 - \frac{2}{m}\right)}{2} \\ &= \frac{4k^3}{m^2} + \left(\frac{3}{m} - \frac{6}{m^2}\right)k^2 - \left(\frac{5}{m} - \frac{4}{m^2}\right)k^2 \\ &\quad - \left(2 - \frac{5}{m} + \frac{2}{m^2}\right) \end{aligned} \tag{14}$$

4 Hardware Implementation

To establish the viability of the FoC architecture, the proposed memory management algorithm was implemented in hardware targeting an Altera Stratix II EP28S60 FPGA device. The implementation consisted of eight ports, each operating at 10 Gbps, representing a switch with an aggregate capacity of 80 Gbps. The maximum departure time, k, was set to 64. Further, the system was designed with a placement decision speedup (s) of four, requiring packet placement decisions to be performed in approximately $12.5ns$. Additionally, there were four unique locations for each departure time in each row memory, i.e. $m = 4$. The prototype system, with speedup and multiple packet placement, utilized eight physical memories consuming a total of 26.624 kb, including logic mapped to memory. This assumed that only packet headers are processed (as payload is irrelevant to the decision making process). However, if a 64-byte payload is assumed, the aggregate on-chip memory requirements increased to 1.05 Mbit.

Switch Ports (N)	Speedup (s)	Memory Units
8	2	19
8	4	5
16	2	58
16	4	18
32	4	51
64	4	144

Table 1. Number of memories in the proposed PSM switch

While eight physical memories were implemented, principally for symmetry and test purposes, no more than five memories were actually required (as stated in Table 1).

The design required 17,383 adaptive look-up tables (ALUTs), or 35% of the ALUTs available on the target device. Proper evaluation of the switch was established by attaching a packet generator, implemented using an Altera Cyclone EP1C6Q-240C8 device, to apply both Bernoulli i.i.d. as well as bursty traffic to the PSM switch fabric. In varying the traffic load and patterns over a wide range of possible scenarios, the viability of the proposed algorithm in a real-time environment was established. The overall latency contributed by the architecture with respect to a pure output-queued switch was 100 ns (8 stages of 12.5ns each).

5 Conclusions

The notion of designing a packet switching fabric on a chip was introduced and discussed from a theoretical as well as practical perspective. In the context of emulating an output-queued switch, it has been argued that a fundamental challenge pertains to the memory-management algorithm employed. A packet-placement algorithm and related high-speed parallel architecture were described in detail, emphasizing the feasibility attributes. Future work will focus on further reducing memory requirements and the incorporation of quality of service (QoS) provisioning. The switch model and framework presented here can be broadened to further investigate the concept of consolidating multiple switch fabric functions on silicon.

6 Acknowledgements

This work has been partially supported by the Department of Energy (DOE) under research grant DE-FG02-04ER25607, and by Altera, Inc.

References

1. McKeown, N.: The iSLIP scheduling algorithm for input-queued switches. IEEE/ACM Transactions on Networking **7** (1999) 188–201

2. Shreedhar, M., Varghese, G.: Efficient fair queueing using deficit round robin. Proc. of ACM SIGCOMM '95 (1995) 231–242
3. Tamir, Y., Frazier, G.: Higher performance multiqueue buffers for vlsi communication switches. In: 15th Annual Symposium on Computer Architecture. (1988) 343–354
4. Iyer, S., Zhang, R., McKeown, N.: Routers with a single stage of buffering (2002)
5. Matthews, B., Elhanany, I., Tabatabaee, V.: Fabric on a chip: Towards consolidating packet switching functions on silicon. In: Proc. IEEE International Conference on Communications (ICC). (2006)
6. Liu, H., Mosk-Aoyama, D.: Memory management algorithms for dsm switches. Stanford Technical Paper (2004)
7. Aziz, A., Prakash, A., Ramachandra, V.: A near optimal scheduler for switch-memory-switch routers. In: Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures. (2003) 343–352
8. Prakash, A., Aziz, A., Ramachandra, V.: Randomized parallel schedulers for switch-memory-switch routers: Analysis and numerical studies, IEEE INFOCOM 2004 (2004)