Switch Fabric on a Reconfigurable Chip using an Accelerated Packet Placement Architecture

Brad Matthews  
Department of Electrical and Computer Engineering  
The University of Tennessee  
Knoxville, TN 37996

Itamar Elhanany  
Department of Electrical and Computer Engineering  
The University of Tennessee  
Knoxville, TN 37996

Vahid Tabatabaee  
Institute for Advanced Computer Studies  
The University of Maryland  
College Park, MD

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 requirements of such systems is \( O(NR) \), where \( N \) denotes the number of ports and \( R \) the data rate of each port. An intuitive way to illustrate the limitations of OQ switches is via a scenario in which \( N \) packets may arrive from the \( N \) input ports, all destined to the same output port, while at the same time all output ports are transmitting packets that have arrived earlier. In a single shared memory architecture, corresponding to this scenario, the memory bandwidth required is \( 2NR \). Clearly, for high port densities and data rates, this constraint dramatically limits the scalability of the switch.

In an effort to retain the desirable attributes of output-queued switches, while significantly reducing the memory bandwidth requirements, distributed shared memory architectures, such as the parallel shared memory (PSM) switch/router, have recently received much attention [3]. 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 [4] 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.

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) [8] as well as some output buffering associated with standard switch architectures. Second, by exploiting the ability to access the multiple on-chip Mb/s 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. FPGA realizations of a FoC offer unique features, including dynamically configuring packet sizes as well as adding traffic management functions that complement the core packet forwarding capabilities.

The rest of the paper is structured as follows. In Section II an overview of parallel shared memory switch architectures is provided from a 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 $K = 3N - 1$ [3]. 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 [5]. 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 [6],[7] 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 $O(\log N)$ 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 $O(\log^2 N)$ at best (assuming $O(\log N)$ 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 $O(1)$. The subsequent cost associated with reducing the placement complexity is an increase in the number of required parallel memories to $O(N^{1.5})$ and a fixed processing latency. The justification resides in the newfound ability to store and switch packets on chip, in particular large FPGA devices, 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 [8] 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.

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 signifi-
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 \( \frac{N}{s} \).

Incoming packets from input port \( i \) are initially inserted into row \( i \mod \left( \frac{N}{s} \right) \). 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 is 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 an analysis that derives optimal values for these parameters. In order to illustrate the underlying memory-management principal, we refer to the following example.

**Example 1.** We refer to the following traffic scenario pertaining to a 9-port switch with speedup of 1, which is
Each row maintains a map that specifies pre-allocated (or reserved) departure times of packets that have successfully passed the diagonal position. This map, which is essentially a binary mask, is referred to as the row's availability map. The diagonal element refers to this map in order to check if its departure time is available at the corresponding row. If it is available, the cell will horizontally shift to the right and mark the corresponding departure time in the availability map as reserved. If it is already reserved by another cell, it will attempt to shift vertically down to another row where the corresponding element position is available. Following the first vertical jump, a packet may still have contention with packets that are ahead of it with the same departure time but have not reached the diagonal element, or packets having the same departure time that have leaped during the same time slot to the same row. To resolve contention originating from the first scenario mentioned, a second binary availability map, termed the reservation map, is maintained at each row. Upon shifting to a new row, each packet must update that row's reservation map by asserting the bit that corresponds to its departure time. This allows future packets to guarantee that the row they are moving into does not contain packets that have arrived during a previous time step with the same departure time.

For pragmatic reasons, both maps (the availability and reservation) cannot be unbounded in size. In all practical switching systems, once a buffer reaches (or is close to reaching) its limit, flow-control signaling is provided to the sources, indicating the need to either slow down or temporarily halt the flow of packets to a given destination/s. Such a mechanism is always required since instantaneous data traffic congestion may occur at any node in the network. In fact, even if the traffic is said to be statistically admissible, meaning that no input or output is oversubscribed, it may still be the case that for short duration of time a given output port is oversubscribed. To address such scenarios, and in an effort to reduce the probability of dropping any packets, the linecards typically host large memories.

### 3.3 On-Chip SRAM Requirements

In order to obtain an estimate of how much memory is required by the PSM design proposed, we refer to analysis pertaining to a pure output queued switch model. The approach taken is to obtain the expected queue size for each output port, from which a good estimate of the aggregate memory requirements is obtained.

Consider an $N \times N$ output queue in which buffering occurs only at the output ports. During each time slot, the switch must transfer all arriving packets into their respective output ports. Let us first assume that arriving packet traffic is uniformly distributed across the output ports and obeys a (memoryless) Bernoulli i.i.d. process. Consequently, a packet arrives at a given input port with probability $p$ and equal chance of being destined for any of the $N$ output ports, i.e. uniformly and at random. Due to the inherent symmetry of the system, we shall analyze a single output port (e.g. output port 1) from which we will be able to derive the behavior at all other ports. Let $Q_t$ denote the number of packets buffered in the output port buffer at time $t$. Since in each time slot, a maximum of $N$ packets may arrive to a given output port and at most a single packet can de-

---

**Figure 3:** Traffic scenario demonstrating the sufficiency bound on the number of memory units for a 9-port switch.

easily extendable to larger switch sizes when $N$ is a complete square. Consider the settings illustrated in Figure 3. There are nine packets with the same departure time residing in the first block of memory. The first three packets are scheduled, while the others are shifted vertically to the second block. Note that packets 4, 5 and 6 reach the diagonal together and hence move down simultaneously. As shown, they may end up in the same row. A similar pattern of behavior is observed for packets 7, 8 and 9. Packets 6 and 9 are scheduled in the second block, while packets 5, 8 and 4, and 7 move to the third block. Once again packets 5 and 8 move simultaneously to the same row and packets 4 and 7 to a different row. Clearly, after two moves we end up with a set of four conflicting packets, e.g. from $k = 3$ we reach a condition with $k = 2$. Similarly, we need two more shifts to reach a point with only one packet in a block.

### 3.2 Memory Management Architecture

Each row, with the exception of the diagonal elements, can be considered a simple shift register whereby packets are shifted one stage to the right at each time step. In each stage of the pipeline, a single packet assignment is attempted per row. These assignment attempts are only made on the diagonal elements of the pipeline structure, such that each column is associated with at most one assignment as well. The motivation for doing so is to isolate memory assignments, thereby reducing the complexity of the placement logic and associated routing resources.
part, the following Markov chain can be used to accurately portray the evolution of the output buffer:

$$Q_{t+1} = Q_t + A_{t+1} - D_{t+1}$$

where $$A_t \in [0, N]$$ denotes the number arrivals during time slot $$t$$ and $$D_t \in [0, 1]$$ is an indicator function representing a departure event. Provided the queue size is not allowed to take negative values, then $$D_t = 1$$ if and only if $$Q_t > 0$$. $$A_t$$ is clearly i.i.d. (as defined above), and its distribution is given by

$$P(A_t = k) = \left( \frac{N}{k} \right) \left( \frac{p}{N} \right)^k \left( 1 - \frac{p}{N} \right)^{N-k},$$

which converges to

$$\lim_{N \to \infty} P(A_t = k) = e^{-p} \frac{p^k}{k!}$$

Moreover, since the arrival process is independent of the departure process, $$A_t$$ is independent of $$Q_t$$, yielding the following result [9]

$$E[Q] = \frac{p^2}{2(1-p)}.$$

By multiplying the above term by $$N$$, we obtain the total average memory consumed by a pure output queued switch under uniform Bernoulli i.i.d. traffic. This provides us with a reasonable estimate of the amount of memory the PSM switch is required to host on chip. As an example, for a load of 90% (i.e. $$p = 0.9$$) the average memory requirement for a large switch is merely 4.95 packets per-port.

Similar analysis may be applied to obtain the average queue size when bursty traffic is considered [9]. It is widely acknowledged that real-life traffic tends to be correlated, or bursty, on many levels [10],[11]. Accordingly, we consider a discrete-time, two-state Markov chain generating arrivals modeled by an ON/OFF source that alternates between the ON and OFF states. Let the parameters $$p$$ and $$q$$ denote the probabilities that the Markov chain remains in states ON and OFF, respectively. An arrival is generated for each ON and OFF, respectively. An arrival is generated for each

$$E[\Lambda]$$

can be directly obtained, from which we assert that the average queue size is

$$E[Q_{(ON/OFF)}] \approx \frac{Bp}{(1-p)}.$$

Note that in this case the average memory size is linearly proportional to the mean burst size.

4. 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 $$B$$ 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

![Figure 4: Example illustrating the proposed memory-management algorithm for a 4-port switch. The state of the pipeline structure is depicted for 4 consecutive time slots.](image)

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 $$b \in [2, 3, .. B]$$.

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

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

$$L(N) = \frac{N}{s} + \left( \left( \frac{s+m}{sm} \right) N - 2 \right) + \left( \left( \frac{s+m}{sm} \right) N - 3 \right) + .. + \left( \left( \frac{s+m}{sm} \right) N - B \right)$$

$$= \frac{N}{s} + N \left( \frac{s+m}{sm} \right) (B-1) - \frac{(B+2)(B-1)/2}{(s+m)}$$

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

(5)

To compute the total number of rows (or memory units), we must determine the maximum number of $$B(N)$$ 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 \( P_4 \leq P_1 \left(1 - \frac{m}{R_1}\right) - mR_1 + m^2. \)

**Proof.** Suppose there are \( P_1 \) packets with the same departure time in the first block. Recall that there can be no more than \( \frac{N}{mR_1} \) 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 \( R_1 \) 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

\[
P_4 = P_1 - mR_1.
\]

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

\[
R_2 \geq \left[ \frac{P_1 - mR_1}{R_1} \right].
\]  

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 is given by

\[
P_3 = P_2 - mR_2 \leq P_1 - mR_1 - m \left[ \frac{P_1 - mR_1}{R_1} \right].
\]

If \( P_1 - mR_1 \) is divisible by \( R_1 \), then

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

otherwise, since \( P_4 \leq P_3 - 1 \), we have

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

\[
P_4 \leq P_1 \left(1 - \frac{m}{R_1}\right) - mR_1 + m^2
\]

The maximum value of 9 is obtained when \( R_1 = \sqrt{N} \). Substituting \( P_1 = N \), yields the following inequality

\[
P_4 \leq (\sqrt{N} - m)^2
\]

Note that if \( N \) is a complete square we have,

\[
P_4 \leq (\sqrt{N} - m)^2
\]  

**Corollary 3.** A sufficient number of parallel memory blocks required for an \( N \times N \) switch, employing the proposed architecture, is \( O(\sqrt{N}) \).

**Proof.** Equation 10 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 \( B(N) \) 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,

\[
B(1) = 1
\]

\[
B(N) = B(N-2\sqrt{N}+1)+3
\]

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

**Theorem 4.** For an \( N \)-port switch, where \( N \leq k^2 \), \( k \in \{1, 2, \ldots\} \) and \( s = m \), the number of memories is

\[
L(N) \leq \frac{4k^3}{m^2} + \left(\frac{3}{m} - \frac{6}{ms}\right)k^2 - \frac{5}{m} - \frac{4}{m^2}k - (2 - \frac{5}{m} + \frac{2}{m^2})
\]

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

\[
B(k^2) \leq \frac{2k}{m} + (2 - \frac{2}{m})
\]

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

\[
R(1) = N/s = k^2/s = m^2/m = m.
\]

Therefore, 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 \( 1, \ldots, k-1 \geq m \) and prove it for \( k+1 \). For \( N = (k+1)^2 \), using lemma 2 and (11) (given that \( N \) is a complete square), we have

\[
B((k+1)^2) \leq B((k+1)^2 - m^2) + 2
\]

\[
\leq \frac{2(k+1-m)^2}{m} + 2 - \frac{2}{m} + 2
\]

\[
= \frac{2(k+1)}{m} + (2 - \frac{2}{m})
\]

Now, we substitute (15) and \( s = m \) in (5) to obtain,
### Table 1: Number of memories in the proposed PSM switch

<table>
<thead>
<tr>
<th>Switch Ports (N)</th>
<th>Speedup (s)</th>
<th>Memory Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>2</td>
<td>19</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>58</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>18</td>
</tr>
<tr>
<td>32</td>
<td>4</td>
<td>51</td>
</tr>
<tr>
<td>64</td>
<td>4</td>
<td>144</td>
</tr>
</tbody>
</table>

\[ L(k^2) = \frac{k^2 2(B-1)}{m} - \frac{(B+2)(B-1)}{2} \]
\[ \leq k^2 \frac{2(2k/m - 2(m+1) + 1)}{m} - \frac{(2k/m + 4 - \frac{2}{m})}{2} \]
\[ = \frac{4k^3}{m^2} + \frac{3 - \frac{4}{m^2}}{m} \]
\[ = \frac{4k^3}{m^2} + \frac{3 - \frac{4}{m^2}}{m} \frac{k^2}{m} - \frac{2k^2}{m^2} \]
\[ = \frac{4k^3}{m^2} + \frac{3 - \frac{4}{m^2}}{m} \frac{k^2}{m} - \frac{2k^2}{m^2} \]
\[ \leq \frac{4k^3}{m^2} + \frac{3 - \frac{4}{m^2}}{m} \frac{k^2}{m} - \frac{2k^2}{m^2} \]

While the availability map provides information pertaining to the memories contents, it does not provide any information regarding the location of the \( \frac{N}{2} - 1 \) packets that potentially arrived during the same time slot as \( p_i \). Having stated that a vector of length \( \frac{N}{2} - 1 \) is sufficient to locate an available row, an occupancy vector must be created to determine the locations of \( \frac{N}{2} - 1 \) packets within the \( \frac{N}{2} - 1 \) subset, reflecting potential adequate rows. Hence, the bitwise-OR of the occupancy vector and availability vector provides a single binary decision vector representing viable rows for packet placement. The decision vector defines rows which (1) are not associated with memories that contain a packet with departure time \( d_i \), and (2) do not contain a packet with the same arrival time as \( p_i \).

The memory-management algorithm is based on selecting an available memory from one of \( \frac{N}{2} - 1 \) memories, such that all \( N \) packets are placed at the end of a at most \( O(\sqrt{N}) \) phases. To accomplish this, we first determine whether the packet in the decision block is blocked by a packet with the same departure time as \( p_i \), as indicated by the availability vector. If \( p_i \) is blocked, the decision vector is then used to select an available row from one of \( \frac{N}{2} - 1 \) rows. The result of this decision process is that \( p_i \) is either written to the memory associated with the row in which it resides or placed in a row that is free of conflict.

While placement is certainly the more difficult task, some consideration must be given to retrieving packets from a shared memory structure. One practical solution is to present a single packet bus to each shared memory unit. As packets are read from memory, each output port will have at most one packet destined for it at any given time. Correspondingly, we can present a shared packet bus of size \( N \times \text{packet\_size} \) to each memory unit such that bus slice \( j \), having width corresponding to a packet size, represents a packet destined for output \( j \). Each shared memory unit determines the destination \( j \) of the packet it is transmitting to an egress port and only drives the \( j^{th} \) bus slice of the packet bus. As a result, each egress port will only be responsible for transmitting the packet located at the \( j^{th} \) bus slice, corresponding to the port enumeration of the packet bus. Implementation of this scheme requires the need for an \( N \times m \times 1 \) multiplexer located in each memory cell.

### 5. HARDWARE IMPLEMENTATION

#### 5.1 Design Approach

The critical path in the design is the memory assignment process that takes place at the diagonal elements, or decision cells, of the pipeline structure. A decision cell residing on row \( r_i \) \((i \in [1, 2, \ldots, L])\) must determine whether a packet, \( p_i \), can be placed at the memory associated with row \( i \). The memory is considered available for packet placement if it contains less than \( m \) packets with the same departure time, \( d_i \), as the packet residing in the decision cell. If the memory is occupied with \( m \) packets, each with the same departure time \( d_i \), then the packet is shifted to a new row \( r_n \).

In light of these placement rules, an availability vector of length \( k \) is created to indicate locations available for the placement of a packet at time \( d_i \). In a distributed shared memory switch, each shared memory will contain \( km \) cells representing \( k \) consecutive departure times. A decision cell need not know the precise number of locations available for a given departure time, only that at the number is less than \( m \). Thus, the size of the availability vector presented to the decision cell is simply \( k \). In order to perform the appropriate placement selection for packet \( p_i \) with departure time \( d_i \), the column of bits pertaining to position \( d \) in the availability map is examined. In accordance with lemma 1, we can state that since a packet always shifts into the next block, the maximal number of rows that is to be examined by each diagonal cell is \( \frac{N}{2} - 1 \). This inherently bounds the critical path of the design, as it defines an upper bound on the search space that must be considered at each step of the memory management process.

#### 5.2 FPGA Implementation Results

To establish the viability of the FoC architecture, the proposed memory management algorithm was implemented in hardware targeting an Altera Stratix II EP28S80 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 64-byte payload is assumed, the aggregate on-chip memory requirements increased to 1.05
Mbit. 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 12.5 ns each).

6. CONCLUSIONS
The notion of designing a packet switching fabric on a chip was introduced and discussed from a theoretical as well as practical perspective. It has been argued that in the context of emulating an output-queued switch, a core 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.

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

8. REFERENCES