Table-lookup based Crossbar Arbitration for Minimal-Routed, 2D Mesh and Torus Networks

DaeHo Seo Mithuna Thottethodi

School of Electrical and Computer Engineering
Purdue University, West Lafayette.
{seod,mithuna}@purdue.edu

Abstract

Crossbar arbitration—which determines the allocation of output ports to packets in the input queues—is a performance-critical stage in the overall performance of routers for input-queued networks. The overall performance of crossbar arbitration depends on two metrics: (a) matching power *– the ability of the arbiter to maximize the number of matches between requesting inputs and free outputs and (b) arbitration throughput – the number of such matches per unit time. Ideally, crossbar arbitration should maximize both metrics. Unfortunately, implementing high performance matching schemes compromises arbitration throughput. Similarly, simpler arbitration mechanisms that deliver high arbitration throughput offer lower matching power.*

The major contribution of this paper is the design of a table-lookup based crossbar arbitration mechanism—`TabArb`*—that delivers superior matching and high arbitration throughput for minimal-routed, two dimensional mesh and torus networks. The two key innovations of* `TabArb` *are: (a) it forwards multiple requests from each input port to multiple output ports to expose adequate matching potential and (b) it employs precomputed tables that store maximum cardinality matches for all possible request combinations. Our technique improves the saturation throughput of adaptive routed mesh network by 14.8%. It offers little improvement for the DOR router due to limited opportunity.*

1 Introduction

Crossbar arbitration[1] is a widely studied problem in interconnection networks architecture with applications ranging from IP routers, ATM routers, multiprocessor interconnection networks and other input-queued networks [1, 4, 5, 7, 9, 10, 11, 14, 18]. Crossbar arbitration is required in input-queued routers because the packets at the input queues demand access to various output ports. The arbitration logic considers the requests from each input port and grants output ports to a subset of requestors. An output port may be granted to at most one input port and each input port may accept at most one grant since multiple packets from an input port cannot traverse the switch simultaneously.

The overall performance of crossbar arbitration depends on two metrics: (a) *Matching Power* – the ability of the arbiter to maximize the number of matches between requesting inputs and free outputs and (b) arbitration throughput – the number of such matches per unit time[2].

Ideally, crossbar arbitration should maximize both metrics. Unfortunately, achieving high matching power compromises arbitration throughput because: (a) Maximizing the match in logic is costly. Most sophisticated matching algorithms incur long latencies to discover maximal matches and (b) Requests for subsequent arbitrations depend on the outcome of the immediately preceding (long latency) arbitration. On the other hand, simpler arbitration schemes that sacrifice matching power but more than compensate for the loss of matching power by enabling higher arbitration throughput [11] have also been proposed. This paper addresses the challenge of designing crossbar arbitration mechanism that delivers the best of both worlds – superior matching power *and* high arbitration throughput – in the context of minimal-routed, two dimensional mesh/torus networks.

There are two key insights that our technique exploits: First, arbitration is typically implemented as a two stage process. The first stage selects a single request at an input port which is then forwarded to the second stage. At the second stage, the actual assignment of output ports to input ports is determined. This approach wastes available matching flexibility since output assignment is performed with-

1-4244-0910-1/07/$20.00 ©2007 IEEE.

[1] Alternately referred to as “crossbar scheduling”, “switch allocation”, “switch scheduling”, “switch arbitration”.

[2] Fairness is an equally important third metric for crossbar arbitration. However, our primary focus is on performance-related metrics because that is the primary concern in the domain of our interest (multiprocessor interconnection networks and onchip networks). We do include anti-starvation guarantees as a failsafe measure.

out knowledge of the entire request set at each input port. Our approach forwards multiple (or all) requests to expose matching flexibility which is then successfully harnessed.

Second, we recast the arbitration problem as a table-lookup problem with the requests of all input ports forming the index into the table and the contents of the table forming the output grants. The table-lookup based approach shifts the problem of discovering the optimal match offline. Instead, only the table lookup time is on the critical path. The table-lookup is conceptual and can be implemented as hard-wired combinational logic.

A naive implementation of the table-lookup approach is expensive in terms of area overheads. We propose two implementation optimizations that make TabArb feasible for 2D networks. Our first optimization exploits the observation that several input request combinations never occur due to routing restrictions such as minimal routing and dimension-ordered routing. Our second optimization exploits the insight that most of the performance gains of TabArb can be captured by forwarding a partial set of requests instead of full request forwarding (FuRF). Arbiters that use traditional 2-stage arbitration with single request forwarding can be seen as extreme form of *partial request forwarding* (PaRF). We demonstrate that there exist other design points in the PaRF design space which capture the benefits of PaRF without sacrificing too much matching power by forwarding the appropriate number of requests. A key advantage of PaRF is that, unlike FuRF, it permits pipelining of the arbitration process, thus increasing arbitration throughput with a small decrease in matching power. Certain points in the PaRF design space reduce the implementation overhead of TabArb as well as the lookup latency.

This paper demonstrates that minimal-routed, two dimensional mesh and torus networks are ideal candidates for table-lookup based arbitration because: (a) Sophisticated matching logic, which rely on iterative [1, 10] or wave-propagation [18] approaches, are slow. For example, Mukherjee *et.al.* estimate that arbitration mechanisms with high matching power consume 4 cycles [11] for the 2D torus network of Alpha 21364 [12]. (b) TabArb implementation costs are small enough to allow a fast lookup and area overheads are negligible. Even with an aggressive 10 FO4[3] clock cycle, our mechanism achieves single cycle arbitration for dimension ordered routers and 2-cycle, pipelined arbitration for minimal, adaptive routers.

Simulations reveal that TabArb improves the saturation bandwidth for minimal, adaptive routers by as much as 14.8% for some traffic patterns. TabArb incurs a small performance penalty for other traffic patterns in which arbitration is not the primary bottleneck.

TabArb—does have a limitation in that it does not scale to high radix routers [8] due to exponential growth of area required to perform the table-lookup. However, high radix routers are rarely implemented with an internal high radix crossbar due to complexity. Kim *et.al.* [8] suggest hierarchical crossbar organization in which the high radix switch is built using low radix sub-switches. TabArb may be employed in such low radix sub-switches to improve overall performance of high radix routers, as well.

In summary, the main contributions of this paper are:

• We develop TabArb – a table lookup based crossbar arbitration mechanism that results in improved matching power via multiple request forwarding. Table-lookup based arbitration enables offline computation of the maximum cardinality matching for the set of forwarded requests. The lookup delays are small enough to enable single-cycle or 2-cycle arbitrations (assuming an aggressive 10 FO4 clock cycle) for DOR and minimal, adaptive network configurations, respectively. TabArb achieves up to 14.8% increase in saturation throughput in a minimal, adaptive 8x8 mesh network over Alpha 21364's arbitration algorithm (SPAA [11]) for some configurations. Other configurations, which present no opportunity for improvement via improved arbitration observe a small performance penalty.

• Our *Partial Request Forwarding* (PaRF) mechanism captures most of the matching power of full request forwarding without any degradation in arbitration throughput. Further, PaRF can also be used to simplify the table implementation with corresponding area and lookup time improvements.

The rest of this paper is organized as follows: Section 2 provides a brief background on the switch arbitration problem and discusses related work. Section 3 describes our table-based arbitration technique and analyzes its memory requirements for various points in the design space. Section 4 describes the evaluation methodology. We present simulation results in Section 5. Section 6 summarizes and concludes this paper.

2 Background and Related Work

Section 2.1 describes the baseline network and router architecture we assume. Section 2.2 provides a brief background on the arbitration problem for input queued networks in general in the specific context of our domain of interest – minimal-routed, 2D mesh/torus networks. Section 2.3 discusses related work.

[3] "FO4" refers to the delay of a signal through an inverter that drives four other inverters of the same size. This is a technology independent measure of delay in circuits and is the metric of choice for timing analysis.

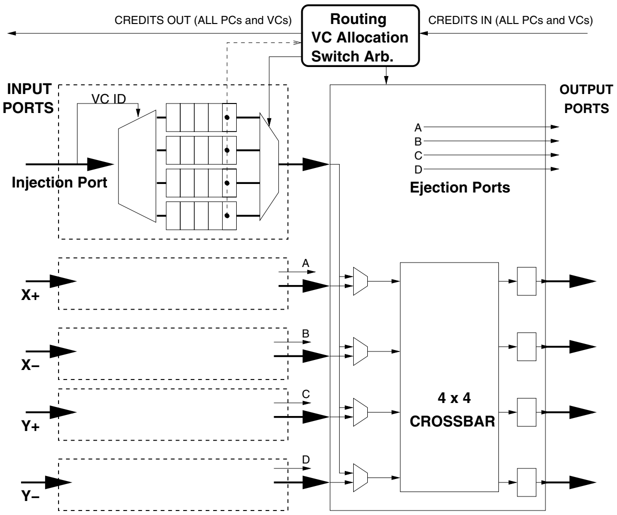
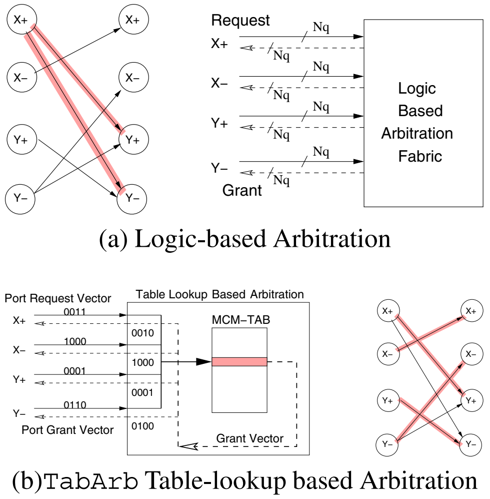


Figure 1. Base Router Model



(a) Logic-based Arbitration

(b)`TabArb` Table-lookup based Arbitration

Figure 2. Arbitration Implementation

2.1 Baseline Network and Router Architecture

In this paper, we consider packet-switched, two-dimensional mesh/torus networks. Each packet is composed of individual flow control units ("flits") with one header flit, data flits and a tail flit. We assume a virtual channel flow control [3] router with several virtual channels per physical channel as shown in Figure 1.

The physical input ports are labeled X+, X-, Y+ and Y- for the dimension (X/Y) and direction (+/-) they traverse. Each flit arrives on a physical input port and is delivered to the appropriate virtual channel buffer (input buffer associated with a virtual channel) by examining the *virtual channel identifier (VCID)* included with each flit's control information (See Figure 1). It then progresses through various stages in the router before it is delivered to a neighboring router. The first pipeline stage—the *routing* stage—determines the packet's possible output physical channels. Next, the packet (i.e. header flit) contends for a virtual channel at the input port of the adjacent node. After successful VC allocation, the packet competes for a switch port in the switch arbitration stage. Though we externally show a 5x4 crossbar to account for the injection port, internally, we use a 4x4 switch with separate injection multiplexers. A similar design has been used previously by Wang *et.al.* in the context of other crossbar optimizations [21]. Our architecture assumes four ejection ports (A, B, C or D in Figure 1). The number of ejection ports is as recommended by Balakrishnan *et.al.* [2] who demonstrate that a single ejection port can be a bottleneck. Further, a flit headed toward an ejection port does not block any other flit headed toward a network port. The focus of this paper is on the arbitration of the 4x4 crossbar connecting the network input ports to the network output ports. Finally, on switch grant, the packet is delivered across the crossbar and physical channel to an input queue of the neighboring node. Our router pipeline model assumes that there's a one cycle delay after a flit is delivered to the neighboring node before it's input buffer is marked free to allow for credit propagation.

2.2 Crossbar Arbitration

The crossbar arbitration stage matches requesting input ports to free output ports such that no output port is assigned to more than one input port and no input port is granted more than one output port. Clearly, the higher the number of such matches, the better the performance since both bandwidth and latency will improve.

The arbitration problem can be stated as a bipartite matching problem. Consider a bipartite graph with two sets of vertices. The first set consists of vertices corresponding to each input port and the second set contains vertices corresponding to each output port. For each flit at the head of the each of the queues at any input port, we add an edge in the graph from its input port to the output ports it wants to traverse. For example, Figure 2(a) shows a sample bipartite graph assuming that the flits at the head of the queues of port X+ want output ports Y+ and Y- (highlighted parts). Each of the other edges may be similarly interpreted. Traditional logic based arbiters consider requests from each input queue and assert the grant signal for the winning contenders, as shown in Figure 2(a). As long as the grants correspond to some bipartite matching in the graph, it is deemed to be a valid arbitration outcome since it ensures that (a) only one flit leaves each input port and (b) no more than one flit is headed to the same output port. Among all possible valid outcomes, the *maximum cardinality matching (MCM)* is the preferred outcome since it maximizes crossbar and link utilization.

2.3 Related Work

There exists a large body of work on the crossbar arbitration problem for input queued switches with many suggested approaches [1, 4, 5, 7, 9, 10, 11, 14, 18]. We only provide a brief overview of selected literature.

Throughout this paper, we consider the SPAA arbitration algorithm [11] as the benchmark to compare against. Though there exist many arbitration algorithms that have significantly higher matching power than SPAA such as Parallel Iterative Matching (PIM [1] and Wavefront Arbiter (WFA [18]), SPAA has been shown to achieve higher overall throughput [11] (because of increased arbitration throughput). SPAA uses simple two stage arbitration. In the first stage (local arbitration) each port nominates the *least-recently selected (LRS)* packet and forwards its request to the output ports. In the second stage (global arbitration), each output grants the request of the least-recently selected requestor. We make some minor changes to SPAA to make it compatible with our baseline router pipeline and our switch dimensions. The matching algorithms are exactly as used in SPAA.

Peh *et.al.* provide details of the construction of logic-based arbitration fabric [14]. Its matching power is likely to be similar to that of SPAA due to use of local and global arbitration schemes with round-robin (SPAA's LRS policy is also demand-slotted round-robin) arbitration at each level. We discuss a generalized form of local and global arbitration called *partial request forwarding* in Section 3.2.

There exist other powerful arbitration mechanisms that are geared towards Internet scale systems with VOQ-based input queues or a combination of input, crosspoint and output buffers [10, 15] which offer throughput and link-utilization guarantees under various traffic assumptions. However, multiprocessor interconnection networks and on-chip networks which need good average case throughput have typically used VC flow control with a limited number of virtual channel input queues rather than the larger number of queues required for VOQ.

3 `TabArb` Arbitration Algorithm

Conceptually, our approach aims to replace the logic based arbitration fabric with a table-lookup as shown in Figure 2(b). We define the *Port Request Vector (PRV)* as a bit vector that encodes all the output port requests of any given input port. One naive way to encode the PRV is to use a 4-bit vector (one bit per output port). If the input port requires an output port, the corresponding bit is set. For example, the PRV for the X+ input port is shown to be "0011" indicating that the port is requesting output ports Y+ or Y-. The bit-vector formed by concatenating all PRVs is called the *Aggregate Request Vector* (ARV). The ARV is a 16-bit vector for the encoding described above.

The lookup table for `TabArb` uses the ARV to index into a table and each entry contains the *Aggregate Grant Vector* (AGV) that matches input and output ports. Since the lookup table can be computed offline, the table can be populated with AGVs that maximize the matching (i.e., the MCM) for any given ARV. The maximum cardinality matching represented by the AGV is shown with highlighted edges in Figure 2(b). The AGV is divided into separate *Port Grant Vectors* (PGVs) and distributed to each port. In the example shown in Figure 2(b), the PGV for port X+ is shown as "0010" to indicate that it was granted the Y+ port. The PGV can be smaller since each input port may get no more than one grant. The grants can be recorded in 3 bits each identifying which, if any, of the four requested ports has been granted. Thus, the AGV is 12-bits wide.

Note, our mechanism does not need to incorporate the information on free output ports since our design assumes flit-by-flit arbitration. Thus, every output port is free after each cycle. There may be no free slots in the input queues at the neighboring nodes. In such a situation, there will be no request for that port anyway since flow control mechanisms (i.e., credit back-propagation) will ensure that no packet competes for that port.

Since our conceptual table-lookup employs read-only tables, it is possible to implement the table lookup via combinational logic by treating the contents of the table as a truth table. A naive combinational logic implementation of the above technique would use the 16-bit ARV as its inputs and generate the 12-bit AGV as its output. However, such an implementation fails to exploit several possible optimizations that reduce the complexity of the table implementation without reducing (or with minimal reduction of) the matching power of the full MCM table.

The remainder of this section discusses optimizations which exploit two specific properties. First, we exploit the restrictions that routing algorithms may impose (Section 3.1). Second, we discuss a technique to forward only a subset of requests to the ARV without a significant degradation in the matching power. Section 3.2 describes variants of this technique called *partial request forwarding* and its impact on complexity. Section 3.3 quantifies the timing properties and area overheads of a combinational logic based implementation of our arbiter. Section 3.4 describes anti-starvation mechanisms for `TabArb`.

3.1 Exploiting Routing Restrictions

In this section, we describe how two common routing restrictions—minimal routing and dimension-ordered routing—can be exploited to significantly reduce the complexity of the TabArb lookup table.

Minimal Routing: Minimal routing algorithms ensure that a packet gets closer to the destination with each network hop. A packet at a particular input port cannot request the same output port since it would essentially return to the same node it came from, thus violating minimal routing. We can exploit this observation to reduce the size of the request vectors with no loss of information. Accounting for minimal routing, each packet may request a subset of three possible ports. Incorporating this reduction produces a nontrivial reduction in complexity since it reduces the length of each PRV by one bit and the ARV by 4 bits. Logically, this corresponds to a factor of 16 reduction in the size of the truth table from 64K-entries to 4K entries. In general, the delay through a combinational logic block need not be proportional (or even correlated) to the size of the truth table. However, we use the size of the truth table as an intuitive measure of the complexity for now. We demonstrate that the relative rank ordering of complexities indicated by truth table sizes are indeed preserved in the final area/delay analysis in Section 3.3. Further, this scheme reduces the required size of the AGV to 8 bits because each PGV must record one of four possibilities: a grant of one of three ports or a grant of no port at all.

Dimension-Ordered Routing: Dimension ordered routing (DOR), a popular and simple routing algorithm offers another opportunity for further reducing the table size. Without loss of generality, we assume that packets traverse the X dimension before they traverse the Y dimension. A packet entering on the Y- input port can only request the Y+ output port or no port[4] at all. A request for the Y- port would violate minimal routing and requests for the X+ or X- ports would violate DOR. Similarly, a packet in the Y+ port queue can either request the Y- port or not request any port. Thus, the PRV of the Y+ and Y- ports can be represented in one bit each. Since packets in the X+ and X- port can still request any combination of the three ports, their PRVs remain 3 bits wide. The ARV is now an 8-bit (3+3+1+1) vector conceptually indexing into a 256 entry truth table.

From the above observations, we arrive at a 256-entry truth table for minimal DOR routers and a 4K-entry truth table for other minimal routers including minimal adaptive routers. We demonstrate in Section 3.3 that the DOR arbiter can be implemented with single-cycle access. However, the more broadly applicable minimal adaptive router requires two-cycle table-lookup. This poses a problem because of the dependency between successive arbitrations. The PGV of the i^{th} arbitration must be determined before PRV of the $(i+1)^{th}$ arbitration can be determined. For the 2-cycle arbitration, it implies that a new arbitration can only commence every other cycle. This reduction in arbitration throughput is serious performance bottleneck and results in degraded performance (i.e., lower saturation throughput). We address this problem in the next section.

3.2 Partial Request Forwarding

Thus far, we assumed that each port's PRV encodes the requests of *each* flit that is at the head of a FIFO at that port. Relaxing this condition to allow the PRV to reflect only a fraction of all requests at that port offers an interesting trade-off. An obvious disadvantage is that the matching outcome of the arbitration may not be optimal since the table-lookup has incomplete request information. On the other hand, the key benefit is that it allows pipelining of arbitration requests. The $(i+1)^{th}$ PRV is no longer dependent on the i^{th} PGV as long as it forwards requests of flits that were not included in the i^{th} PRV.

In general, we consider all arbitration schemes which forward only a partial view of the port's requests to be *partial request forwarding* (PaRF) schemes. PaRF defines a space of possible designs. On one end of the spectrum is the full request forwarding design described earlier. On the other extreme is a design wherein the request from only one flit is forwarded by each port. Intermediate design points correspond to various matching power vs. arbitration throughput trade-offs. SPAA's design corresponds to an extreme point in the PaRF design space as it forwards the request of only one flit[5] in each PRV.

The second interesting outcome of PaRF is that the size of the truth table (our measure of complexity) can be reduced if all combinations of output requests are not possible under the PaRF scheme. For example, consider a PaRF scheme in which only one flit's requests are being forwarded. Further, assume that each flit may request at most one output. Note, this assumption does not imply that routing restrictions are being imposed. For example, oblivious routing algorithms request at most one output port though the choice of that output port may be randomized. Even adaptive routing algorithms may be "temporally adaptive" as in the Alpha 21364. In "temporally adaptive" schemes, the flit's output requests are adaptive in time. For any given arbitration, each flit requests a single output port. However, if it fails the first time, the same flit may try for another port in subsequent attempts.

[4] This effectively means that it is requesting the ejection port which will be handled outside the main crossbar.

[5] In the Alpha 21364 router, each port forwards the nominations of two packets to two different SPAA arbiters. However, we exclude consideration of the secondary arbiter since such arbiter duplication can be achieved with any arbitration scheme.

For the above class of routing algorithms where each flit can request at most one output port, consider the possible port request combinations if exactly one flit's requests are forwarded. That flit may request one of the three possible output ports (i.e., the number of ways to choose one ports from three ports = $\binom{3}{1}$). It is also possible that it requests no cross-bar ports at all since it may request the ejection port (i.e., the number of ways to choose zero ports from three ports = $\binom{3}{0}$). Hence, the number of unique bit combinations represented by the PRV is at most 4 when using PaRF with one request forwarding. The above observation can be exploited to re-encode the four unique PRVs in two bits. Thus, a router with all ports forwarding only one request reduces the table size by a factor of two for each port. This implies that even a fully adaptive (albeit with temporal adaptivity) router can achieve table size of 256. As stated before, the matching power of PaRF scheme which forwards only one request is lower. However, this data point represents an additional matching-power vs implementation-cost trade-off since it reduces the size of the table. Note, the PRV is a 3-bit vector whenever the port forwards more than 1 output request.

Heterogeneous PaRF:
The number of requests forwarded from each port need not be the same. We define the "PaRF$< i, j, k, l >$" scheme to mean that the scheme forwards the requests of up to i flits from the X+ port, j flits from the X- port, k flits from the Y+ port and l flits from the Y- port. By this notation, the PaRF variant discussed in the previous section is represented as PaRF$< 1, 1, 1, 1 >$.

An alternate PaRF$< 3, 3, 1, 1 >$ configuration offers an interesting design point. Because it forwards more requests than PaRF$< 1, 1, 1, 1 >$, its matching power is expected to be higher. The PRVs for the ports are 3-bits, 3-bits, 2-bits and 2-bits respectively resulting in a 10-bit ARV (or a 1K-entry truth table). Our results (Section 5.3) show that PaRF$< 3, 3, 1, 1 >$ is adequate to capture all the benefits of TabArb.

3.3 Table Lookup Access Times and Area Overheads

The previous sections used truth table sizes as a heuristic measure of complexity. In order to quantify the overheads more precisely, we have obtained delay and area overheads as reported by a synthesized (using Synopsys' Design compiler) VHDL model of our mechanism. In this section we briefly present the conclusions of our area and timing measurements. We omit the detailed methodology and area/delay results due to lack of space.

The three key conclusions are:

- When synthesized to optimize area, the DOR arbiter can be accommodated in one 10 FO4 clock cycle. However, the arbitration stage of the adaptive routers (both PaRF and FuRF) require a two-stage arbitration pipeline. While FuRF requires a pipeline bubble after each arbitration attempt, PaRF can be fully pipelined as described in Section 3.2. We refer to the shallow pipelined router architecture with a single-cycle stage each for routing, VC allocation, switch arbitration and transfer as the 'RVST' pipeline. The DOR router uses the RVST pipeline configuration. For the adaptive routers, we use a deeper pipeline with two stages of VC allocation and two stages of switch arbitration, which we refer to as the 'RV2S2T' pipeline.
- The DOR (Adaptive, PaRF) configuration has modest area overhead and requires the area equivalent of one (four) input queue buffer(s) that holds eight flits of 64 bits each. For comparison, a router with five physical input ports and four virtual channels per physical input port has twenty (20) such input queues.
- The adaptive router with FuRF has significant area overhead—larger by two orders of magnitude compared to the simpler configurations (i.e., DOR and adaptive with PaRF). This represents a second drawback for the FuRF adaptive router since there is also an arbitration throughput penalty due to the dependence between consecutive arbitrations. As such, while we include the FuRF adaptive router for comparison, our recommendation is for the PaRF version.

3.4 Fairness

The previous sections focused purely on the two aspects of arbitration performance: matching power and arbitration throughput. This section examines the fairness of TabArb. One possible concern with table-lookup based arbitration schemes is that multiple attempts with the same ARVs generate the same AGVs. Thus, a contender that lost in one arbitration attempt (either because it was not part of any MCM matching or because it was not part of the MCM matching stored in the table even though it was a part of an alternate MCM matching) is consistently declined in each subsequent attempt as well. This problem could lead to starvation wherein some contenders consistently lose out.

We adopt existing anti-starvation mechanisms, also used in the Alpha 21364 router [11], into TabArb. This approach optimizes for performance assuming that starvation is the uncommon case. However, starvation is guaranteed not to occur since any packet that does get starved (as detected by timeouts) supersedes any other packet. Multiple anti-starvation requests are served in FIFO order. We observed from our experiments that the effect of varying the timeout threshold of anti-starvation mechanism is intuitive. Too low a threshold results in reduced overall throughput due to frequent invocation of the anti-starvation mechanism. Too high a threshold results in the possibility of some packets suffering increased latencies. A threshold of 20 cycles was experimentally found to be the best. We do not include these results due to lack of space.

Extensions TabArb focuses on minimal, two dimensional mesh and torus networks. This is a broadly applicable class of networks that has sparked renewed interest with the advent of onchip networks [20, 19, 13, 16]. TabArb does not scale in area and delay to high radix routers or non-minimal routing routers since they effectively increase the length of the request vector. Some high-radix routers may use hierarchical cross bars which are built with low-radix crossbar subswitches [8]. TabArb arbitration may be a candidate for allocation of subswitch ports.

Summary: In this section, we described the TabArb arbitration algorithm which achieves table-lookup-based maximum cardinality matching and multiple request forwarding. The area overheads of TabArb are modest. The only high-overhead version (FuRF adaptive) also suffers from poor arbitration throughput. However, *Partial request forwarding* can recapture high arbitration throughput without significantly compromising on matching power. Finally, we demonstrated that PaRF also significantly reduces the area overhead. In the next two section, we evaluate TabArb in comparison with SPAA.

4 Evaluation Methodology

We use a modified version of the PoPnet [17] network simulator. PoPnet models a four-stage router pipeline similar to our basic RVST pipeline. We assume that the logic delay of the routing stage and the physical traversal to the neighboring node can be accommodated within one clock cycle (10 FO4) each. The arbitration stage takes 1 or 2 cycles depending on the configuration (see Section 3.3). For the configurations where switch arbitration takes 2 cycles, we assume that VC allocation also takes two cycles; otherwise it takes one cycle. The anti-starvation logic in our arbitration mechanism uses a timeout of 20 cycles. The network was simulated for 100,000 cycles which was seen to be adequate for the network to achieve steady state. The reported latency includes queuing time before a packet is injected into the network and is measured till the tail flit is drained at the destination node. The channels are full-duplex bidirectional links.

The basic router model assumes the architecture discussed in Figure 1. We evaluate the performance of TabArb for two network configurations listed in Table 1.

- **Lite network** : This configuration uses a 4x4, 2D-mesh topology and dimension ordered routing. This is similar to the interconnection network used in the RAW processor prototype [19]. We use single-flit packets to simulate operand transport. This configuration uses the RVST pipeline.

Parameter	lite	aggressive
Topology	2D Mesh	2D Mesh
Network Size	4x4	8x8
Routing	DOR	Adaptive
Router Pipeline	RVST	RV2S2T
Packet size	1 flit	5 flit
VCs/PC	4	8
RF Scheme	FuRF	PaRF
PaRF scheme	–	$< 3, 3, 1, 1 >$
Arb. Latency	1 cycle	2 cycles
Cycles b/w arbs.	1	1

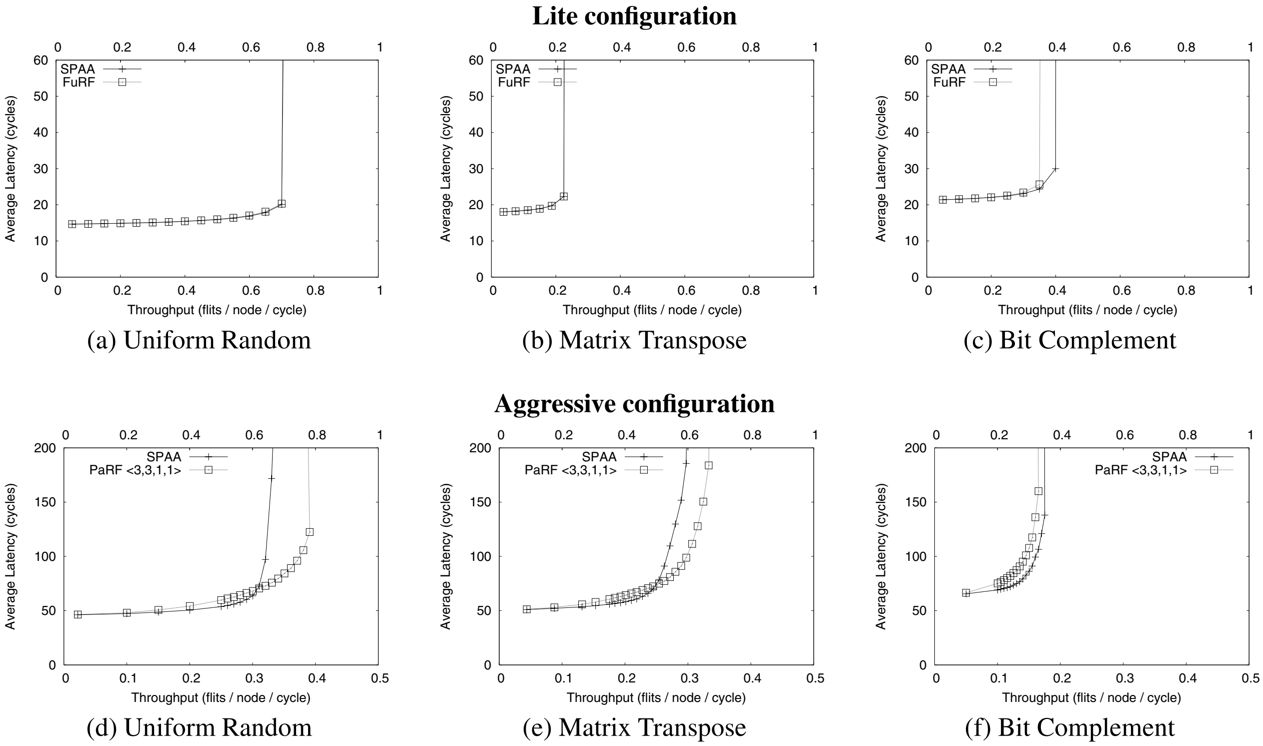
Table 1. Network Configurations

- **Aggressive Network** : For this configuration, we use an 8x8, 2D mesh network with fully adaptive routing. The best performing scheme was the PaRF$< 3, 3, 1, 1 >$ scheme. This configuration uses the area-optimized version of the table implementation which corresponds to the RV2S2T pipeline. However, a new arbitration can begin each cycle because of PaRF.

We use Duato's adaptive routing algorithm with deadlock avoidance for the adaptive routing configurations [6]. We simulate a steady offered load at various injection rates. We consider three different traffic permutations, *uniform random (*ur*), bit complement (*bc*) and matrix transpose (*mt*)*. These communication patterns differ in the way a destination node is chosen for a given source node with bit co-ordinates $(a_{n-1}, a_{n-2}, \ldots, a_1, a_0)$. The bit co-ordinates for the destination nodes are $\overline{(a_{n-1}, a_{n-2}, \ldots, a_1, a_0)}$ for *complement* and ($a_{(n/2)-1}$, $a_{(n/2)-2}$, $\cdots$, a_0, $a_{n-1}, a_{n-2} \cdots, a_{(n/2)0}$ for *matrix-transpose*. We define saturation throughput as the delivered throughput at which average packet latency is twice the latency in an unloaded network.

5 Results

The four primary conclusions from our simulations are: (1) For the aggressive configuration, TabArb scheme outperforms SPAA improving the saturation throughput by 14.8% and 11.6% for ur and mt traffic patterns, respectively. The improvement in performance is predominantly due to increased matching power. TabArb schemes achieve the same arbitration throughput as SPAA (i.e., one arbitration every cycle). TabArb provides no benefit for the lite configuration.



(a) Uniform Random

(b) Matrix Transpose

(c) Bit Complement

(d) Uniform Random

(e) Matrix Transpose

(f) Bit Complement

Figure 3. Overall Performance

(2) There is a small performance penalty for the `bc` traffic pattern in both configurations. This is because `bc` is a bisection-bandwidth bound pattern and crossbar throughput is not a performance bottleneck.

(3) `PaRF`'s contribution in improving arbitration throughput is extremely important for performance in the `aggressive` configuration. The adaptive algorithm with `FuRF` is worse than SPAA in spite of higher matching power. However, the adaptive router with `PaRF` achieves 14.8% improvement in saturation throughput over SPAA.

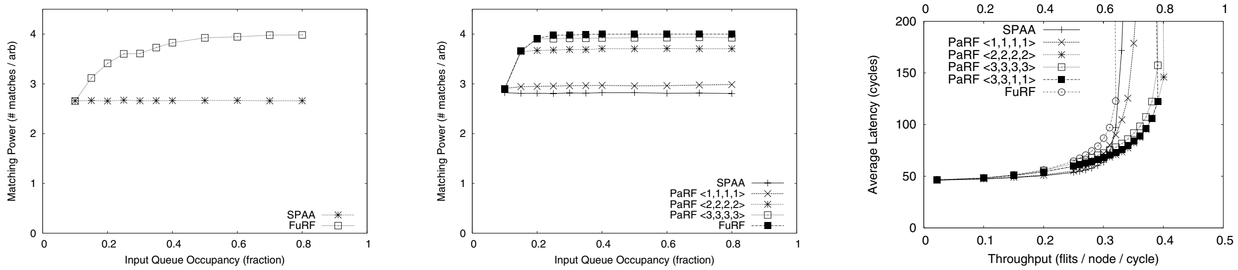
(4) `PaRF` captures much of the matching power of `FuRF` when reasonable fractions of requests (e.g., $< 3, 3, 1, 1 >$ and $< 2, 2, 2, 2 >$) are forwarded. Significant degradation in matching power is felt only at extreme data-points such as the `PaRF`$< 1, 1, 1, 1 >$ case, where its matching power is 27% lower than that of `FuRF`.

The remainder of this section describes the results in detail. First, we describe the overall performance of `TabArb` schemes. Next, we quantify the loss of matching power due to `PaRF`. Finally, we evaluate the impact of loss of arbitration throughput.

5.1 Overall performance

The simulation results presented in Figs 3(a), (b) and (c) correspond to the three traffic patterns (`ur`, `mt` and `bc`, respectively) for the `lite` configuration. Similarly, Figs 3(d), (e) and (f) correspond to the three traffic patterns for the `aggressive` configuration. Each graph has two curves – one each for SPAA and `TabArb`. Each graph has delivered throughput on the x-axis at two different scales: (a) the x-axis below specifies the absolute value (flits/node/cycle) of throughput and (b) the x-axis above normalizes delivered throughput to network capacity. The graphs plot average packet latency in cycles on the Y-axis. For any given curve, the delivered throughput increases without much increase in latency at low loads. This is as expected because network packets do not experience contention before network saturation. However, when load approaches saturation throughput, we observe a sudden and sharp increase in latency. The curve that saturates at the highest load represents the better router.

There is no significant improvement for the `lite` configuration. This is because the same phenomenon that helped optimize the table implementation complexity of `TabArb` (i.e., DOR routing restrictions) also limits the opportunity for `TabArb` to outperform DOR. When 50% of the input ports (Y+ and Y-) offer no flexibility in choice of output ports, the opportunity to improve performance diminishes. `TabArb` improves saturation throughput by 14.8% and 11.6% for `ur` and `mt` traffic patterns, respectively.



(a) Matching Power (FuRF, DOR) (b) Matching Power (PaRF, Adaptive) (c) Performance (PaRF)

Figure 4. Matching Power, Arbitration Throughput tradeoffs in TabArb

5.2 Matching Power

To measure matching power independently from performance, we computed the average number of matches per arbitration at various levels of queue occupancy. For example, a point at the 0.5 mark along the X-axis corresponds to a point where half the input VC queues have a flit at the head of the queue as a candidate for switch arbitration.

Figure 4(a) compares the matching power of SPAA with that of TabArb for the lite configuration. Note that the matching power saturates fairly quickly and stays constant beyond that. Since the table-lookup delay for the lite configuration can be comfortably accommodated in a single clock cycle of 10 FO4, we do not examine the matching power of PaRF schemes for this configuration.

Figure 4(b) quantifies the loss in the matching power of various PaRF schemes for the adaptive router configuration. The graph has six (6) curves. One curve is for full request forwarding (FuRF) and another is for SPAA. The remaining four correspond to each of the following four PaRF schemes: PaRF$< 1,1,1,1 >$, PaRF$< 2,2,2,2 >$, PaRF$< 3,3,1,1 >$ and PaRF$< 3,3,3,3 >$.

FuRF's matching power is similarly 42% higher than that of SPAA for the aggressive configuration. However, as we show in the next section, FuRF's performance is severely degraded due to its non-pipelineable 2-cycle arbitration latency.

The trends for PaRF are exactly as expected. Increasing the amount of request forwarding results in improved matching power but with diminishing returns. In practice, we find that PaRF$< 3,3,1,1 >$ offers adequate matching power and increasing it beyond that offers little or no improvement in saturation throughput.

5.3 Importance of Arbitration Throughput

From the above section, we observe that the matching power of PaRF varies with the amount of request forwarding. Here, we examine the overall impact on performance of the various PaRF schemes. Figure 4(c) plots the latency-throughput curves for various arbiter configurations for the adaptive router (aggressive) with the ur traffic pattern.

We observe that, other than at the extreme data-point (i.e., PaRF$< 1,1,1,1 >$), there are significant gains in saturation throughput. PaRF$< 3,3,1,1 >$, which is the configuration with the smallest improvement in matching power, captures almost all the improvement in performance. Subsequent increase in request forwarding either does not help, or actually hurts performance in the case of FuRF. We observe that FuRF's latency starts increasing at low load levels and quickly escalates. This is primarily because of reduced arbitration throughput.

6 Conclusion

Crossbar arbitration is a performance-critical step in routers. The two metrics for evaluating crossbar arbitration mechanisms are (a) matching power and (b) arbitration throughput. Previous arbiter designs have delivered on one of the metrics to the exclusion of the other. The main contribution of this paper is the development of a table-lookup based arbitration algorithm for minimally routed, 2D networks that delivers both superior matching power and arbitration throughput.

Our technique exploits two key insights. First, traditional arbiters forward only one request from each input port as contenders for the output ports, thus eliminating much of the matching potential. Our design forwards multiple (or all) requests from the input ports to the subsequent arbitration stage exposing much of the matching potential to the arbiter. Second, the online computation of good matches incurs significant delays. Our approach employs offline-computed maximum cardinality match tables to maximize matching power. The actual arbitration is reduced to a table-lookup. Since the tables are read-only tables, the actual implementation of table-lookup is achieved by hardwired combinational logic. The area overhead of the combinational logic is modest – equivalent to the area of four input queues with eight entries each. Other optimizations include (a) the utilization of DOR's routing restrictions to reduce implementation complexity and (b) application of partial request forwarding to increase arbitration throughput without significantly degrading matching power.

With the optimizations in place, `TabArb` can operate in a single 10 FO4 clock cycle for DOR routed mesh/torus networks. For adaptive routed networks, `TabArb` requires two clock cycles, but the arbitration throughput is preserved. Though, there are no gains for the DOR router (mainly due to lack of opportunity), `TabArb` achieves a 14.8% improvement (over SPAA) in saturation throughput for adaptive routers.

Acknoledgements We would like to thank the anonymous referees for their feedback and suggestions to improve this paper. This work is supported in part by NSF grant CCF-0541385, Purdue Research Foundation XR Grant No. 6904010 and Purdue University.

References

[1] Thomas E. Anderson, Susan S. Owicki, James B. Saxe, and Charles P. Thacker. High-speed switch scheduling for local-area networks. *ACM Trans. Comput. Syst.*, 11(4):319–352, 1993.

[2] S. Balakrishnan and D. K. Panda. Impact of multiple consumption channels on wormhole routed k-ary n-cube networks. In *Int'l Parallel Processing Symposium (IPPS '93)*, pages 163–167, 1993.

[3] W. J. Dally. Virtual-Channel Flow Control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–205, March 1992.

[4] Paolo Giaccone Devavrat Shah and Balaji Prabhakar. Efficient randomized algorithms for input-queued switch scheduling. *IEEE Micro*, 22(1):10–18, January/February 2002.

[5] H. Duan, J. W. Lockwood, S. M. Kang, and J.D. Will. High-performance OC-12/OC-48 queue design prototype for input-buffered ATM switches. In *INFOCOM'97*, pages 20–28, Kobe, Japan, April 1997.

[6] J. Duato. A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks. *IEEE Transactions on Parallel and Distributed Systems*, 4(12):1320–1331, December 1993.

[7] Paolo Giaccone, Devavrat Shah, and Balaji Prabhakar. An implementable parallel scheduler for input-queued switches. *IEEE Micro*, 22(1):19–25, January/February 2002.

[8] John Kim, William J. Dally, Brian Towles, and Amit K. Gupta. Microarchitecture of a high-radix router. In *ISCA '05: Proceedings of the 32nd Annual International Symposium on Computer Architecture*, pages 420–431, 2005.

[9] M. G. Ajmone Marshan, A. Bianco, and E. Leonardi. Rpa: A flexible scheduling algorithm for input buffered switches. *IEEE Transaction on Communications*, 47(12):1921–1933, December 1999.

[10] Nick McKeown. The islip scheduling algorithm for input-queued switches. *IEEE/ACM Trans. Netw.*, 7(2):188–201, 1999.

[11] S. S. Mukherjee, F. Silla, P. Bannon, J. Emer, S. Lang, and D. Webb. A Comparative Study of Arbitration Algorithms for the Alpha 21364 Pipelined Router. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASP*, pages 223–234, 2002.

[12] Shubhendu S. Mukherjee, Peter Bannon, Steven Lang, Aaron Spink, and David Webb. The alpha 21364 network architecture. In *HOTI '01: Proceedings of the The Ninth Symposium on High Performance Interconnects (HOTI '01)*, page 113, Washington, DC, USA, 2001. IEEE Computer Society.

[13] R. Mullins, A. West, and S. Moore. Low-latency virtual-channel routers for on-chip networks. In *Proceedings of the 31st annual international symposium on Computer architecture*, page 188. IEEE Computer Society, 2004.

[14] L. S. Peh and W. J. Dally. A Delay Model and Speculative Architecture For Pipelined Routers. In *Proceedings of the Seventh International Symposium on High-Performance Computer Architecture (HPCA-7)*, pages 255–266, January 2001.

[15] R. Rojas-Cessa, E. Oki, and H.J. Chao. Cixob-k: combined input-crosspoint-output buffered packet switch. In *IEEE Global Telecommunications Conference, 2001. GLOBECOM '01*, volume 4, pages 2654–2660, 2001.

[16] DaeHo Seo, Akif Ali, Won-Taek Lim, Nauman Rafique, and Mithuna Thottethodi. Near-optimal worst-case throughput routing for two-dimensional mesh networks. In *ISCA '05: Proceedings of the 32nd annual international symposium on Computer architecture*, pages 432–443, 2005.

[17] L. Shang, L. S. Peh, and N. K. Jha. Dynamic voltage scaling with links for power optimization of interconnection networks. In *Proceedings of the 9th IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 79–90, Feb 2003.

[18] Y. Tamir and H. C. Chi. Symmetric crossbar arbiters for vlsi communication switches. *IEEE Trans. Parallel Distrib. Syst.*, 4(1):13–27, 1993.

[19] M.B. Taylor, W. Lee, S. Amarainghe, and A. Agarwal. Scalar Operand Networks: On-chip interconnect for ILP in Partitioned Architectures. In *Proceedings of the International Symposium on High Performance Computer Architecture*, pages 341–353, 2003.

[20] Michael Bedford Taylor and Others. The raw microprocessor: A computational fabric for software circuits and general-purpose programs. *IEEE Micro*, 22(2):25–35, 2002.

[21] Hangsheng Wang, Li-Shiuan Peh, and Sharad Malik. Power-driven design of router microarchitectures in on-chip networks. In *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, page 105. IEEE Computer Society, 2003.