A Cost-Efficient Router Architecture for HPC Inter-Connection Networks: Design and Implementation

Yi Dai, *Member, IEEE*, Kai Lu, Liquan Xiao, and Jinshu Su

Abstract—High-radix routers with lower latency and higher bandwidth play an increasingly important role in constructing large-scale interconnection networks such as those used in super-computers and datacenters. The tile-based crossbar approach partitions a single large crossbar into many small tiles and can considerably reduce the complexity of arbitration while providing higher throughput than the conventional switch implementation. However, it is not scalable due to power consumption, placement, and routing problems. Inspired by non-saturated throughput theory, this paper proposes a scalable router microarchitecture, termed Multiport Binding Tile-based Router (MBTR). By aggregating multiple physical ports into a single tile a high-radix router can be flexibly organized into different tile arrays, thus the number of tiles and hardware overhead can be considerably reduced. For a radix-64 router MBTR achieves up to $50 \sim 75\%$ reduction in memory consumption as well as wire area compared with a hierarchical switch. We theoretically deduce the sufficient and necessary conditions for the asymmetrical crossbar to achieve un-saturated relative 100 percent throughput. Based on this observation we analyze the MBTR throughput and derive the condition that should be satisfied by the MBTR design parameters to yield 100 percent throughput. We further discuss how to make a trade-off between MBTR parameters based on the constraints of performance, power and area. The simulation results demonstrate MBTR is indistinguishable from the YARC router in terms of throughput and delay, and can even outperform it by reducing potential contention for output ports. We have fabricated a 36-port MBTR chip at 28 nm, providing 100 Gb/s bidirectional bandwidth per port, with a fall-through latency of just 30 ns. Internally it runs at 9.6 Tb/s, thus offering a speedup of $1.34\times$.

Index Terms—High-radix router architecture, non-saturated throughput theory, ASIC implementation, arbitration logic

1 Introduction

With the emerging big data analytics, machine learning, and business optimization applications on the scale-out cloud data centers and other high performance computing (HPC) systems consisting of large numbers of thin nodes, it becomes crucial to design interconnection networks to efficiently manage the distributed computing and storage resources. Such large-scale interconnection networks require large number of integrated nodes, high per-node computation capacity, large memory bandwidth, and high inter-node parallelism, making it extremely challenging to meet the design constraints of cost, scalability, power consumption, performance, and reliability. For example, state-of-the-art large-scale supercomputers like the Tianhe-2 system has tens of thousands of compute nodes, and are striving to progress toward larger sizes [1]. An exascale system would require hundreds of thousands of interconnected nodes. On the other hand, with the extraordinary growth in parallelism, HPC' s performance is determined more and more by how data is communicated among the increasing number of computing resources rather than the arithmetic operations being performed [2]. Although the remote memory access latency and bandwidth at the individual nodes can be improved, it is the large node count that poses a tough challenge on the design of high-bandwidth low-latency inter-node communication.

Traditionally large-scale interconnection networks were built with low-radix routers where the number of ports is relatively small. However, the recent trend is to build networks with higher radix switches with more narrow ports rather than fewer wide ones running at higher bandwidth. The main reason for this is that signal rates have gone up faster than the size of the packets sent by network protocols. High-radix routers reduce network cost by lowering the network diameter while providing a high bisection bandwidth and path diversity. One great challenge facing high-radix routers is how to create a scalable router microarchitecture. Crossbar architectures are indispensable in modern off-chip and on-chip interconnection networks because they are non-blocking and deterministically fair, and can deliver performance guarantees. However, crossbar designs with the radix of N are hard to scale due to their $O(N^2)$ point-to-point wires, the complexity of arbitration logic and the power consumption of the buffers.

The microarchitecture of high-radix switches has attracted a lot of attention since Kim et al. quantified the benefits and proposed a hierarchical crossbar switch called Hiera [3].

• *The authors are with the Department of Computer Science, National University of Defense Technology, Changsha 410073, China. E-mail: {daiyi, kailu, liquanxiao, jinshusu}@nudt.edu.cn.*

Manuscript received 11 Mar. 2018; revised 29 July 2018; accepted 22 Sept. 2018. Date of publication 1 Oct. 2018; date of current version 13 Mar. 2019. (Corresponding author: Yi Dai.)
Recommended for acceptance by J. Zhan.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2018.2873337

Hiera partitions a single large crossbar into many small subswitches, or tiles, and places intermediate buffers at the inputs and the outputs of these subswitches, thus decomposing the centralized arbitration into three-stage distributed and separable arbiters. This architecture was later applied to the radix-64 YARC router deployed in the Cray BlackWidow system [4], and was further evolved and implemented in the Gemini and Aries routers, which are widely used in current Cray XT and XC systems [5], [6]. The 48-port Aries router consists of 48 tiles arranged in a 6×8 matrix, each integrating one physical link. In essence, YARC and Aries adopt the same router architecture. The main improvement from YARC to Aries is to combine four high performance network interface controllers (NIC) and a 48-radix router. Eight of the tiles connect to NIC for packet injection, and the other forty tiles connect to SerDes lanes to provide intergroup and intragroup interconnection [6].

The commercial success of the hierarchical, tiled router represented by YARC can be attributed to two unique characteristics. First, it partitions a N-radix router into N tiles thus decomposing a centralized arbitration into many distributed and asynchronous arbiters. Compared with conventional crossbars, this not only lowers the arbitration complexity, but also reduces the number of output ports driven by each input port. Furthermore, each tile is identical and and produces a very regular structure for replication and physical implementation in silicon [6]. Second, it distributes input traffic to multiple parallel tiles in the same row and thus effectively mitigates the head-of-line (HOL) blocking such that a 100 percent throughput can be achieved [7]. The drawback of the YARC router is that they still require an excessive number of wires and silicon area and hence, do not scale well to meet the power efficiency challenge of high-radix network interconnection. Existing scalable routers attempt to lower the wiring complexity and buffer requirements by applying network topology to subswitch interconnection [8] or by using bufferless switching [10].

In this paper, we propose a Multiport Binding Tile-based Router (MBTR) architecture that provides a more scalable and resilient solution for a high-radix router implementation. MBTR's resilience exists in its configurable design parameters and adaptable architecture for different design requirements. However, a crucial architectural constraint of YARC is that each tile is bound to dispatch and multiplex traffic from and to only one bidirectional port. Hence, by simply binding multiple ports into one tile, MBTR can flexibly organize a N-port router into arbitrary array of tiles, thus lowering area and power requirements due to reduced number of tiles. Also, MBTR inherits all the advantages of YARC' s structure such as the reduced HOL blocking, high throughput, and a considerable reduction in global wires, memory, and registers. From a tile' s perspective, the number of row and column buffers are respectively equal to the number of input ports in the same row and the number of output ports in the same column. Therefore, the buffer overhead of each tile is ultimately determined by radix N for both YARC and MBTR. The total memory requirement is proportional to the number of tiles. By reducing the number of tiles, MBTR can not only reduce the memory consumption but the complexity of global wiring considerably. Furthermore, MBTR makes it possible to achieve a good tradeoff between resource constraints and arbitration performance based on design targets. Performance is generally indistinguishable from a YARC router and can even outperform it by reducing potential contention for output ports.

The contributions of this paper include:

1) We propose a scalable and flexible design approach based on multiport binding to build high-radix routers with reduced memory, area, and global wire requirements while maintaining 98 percent throughput under uniform traffic [4].
2) We propose a non-saturated relative 100 percent throughput theory that redefines design targets for the parallel multi-stage switch architecture. We theoretically prove that 100 percent absolute throughput under random output selection can be achieved by the MBTR, if and only if non-saturated relative 100 percent throughput can be established by its subswitches. We further provide design criteria about determining the structural parameters of the MBTR in order to meet different design constraints.
3) Some simple but efficient schemes for economical integrated circuit (IC) implementation are systematically presented in detail, such as the credit management for on-demand dynamic VC allocation, and reliable data transmission based on VCT-based flow control. Particularly we implemented a two-stage arbiter structure which can be completed within one clock cycle and achieves almost the same performance as the multi-iteration iSLIP algorithm [11], but with much lower hardware complexity.
4) We report the implementation of an ASIC MBTR chip with 28 nm technology that internally runs at 9.6 Tb/s and provides a fall-throughput latency of just 30 ns, running at 700 MHz.

The paper is organized as follows. In Section 2, we introduce the microarchitecture of MBTR, and evaluate the wire and buffer requirements with different parameters. In Section 3 the non-saturated throughput theory is proven and the sufficient and necessary conditions of 100 percent throughput for MBTR are theoretically deduced. Some optimization schemes for MBTR hardware implementation are presented in Section 4. We use cycle-accurate simulation to evaluate the throughput, delay and arbitration performance of our approach in Section 5. Section 6 reports the ASIC implementation of our MBTR chip. Finally, the related and future work is discussed in Section 7 and we conclude in Section 8.

2 MBTR Design

In this section, we present the MBTR microarchitecture. First, we outline the structure of the tile-based routers and present the basic concept of high-radix router design based on multiple homogeneous tiles. Second, we describe the microarchitecture of MBTR, evaluate its hardware cost with different design parameters.

2.1 Microarchitecture of Tile-Based Routers

We start with an analysis of the advantages and limitations of the YARC router which is used to build the Cray BlackWidow folded-Clos network [12]. By organizing a high radix router into multiple tiles, YARC effectively reduces

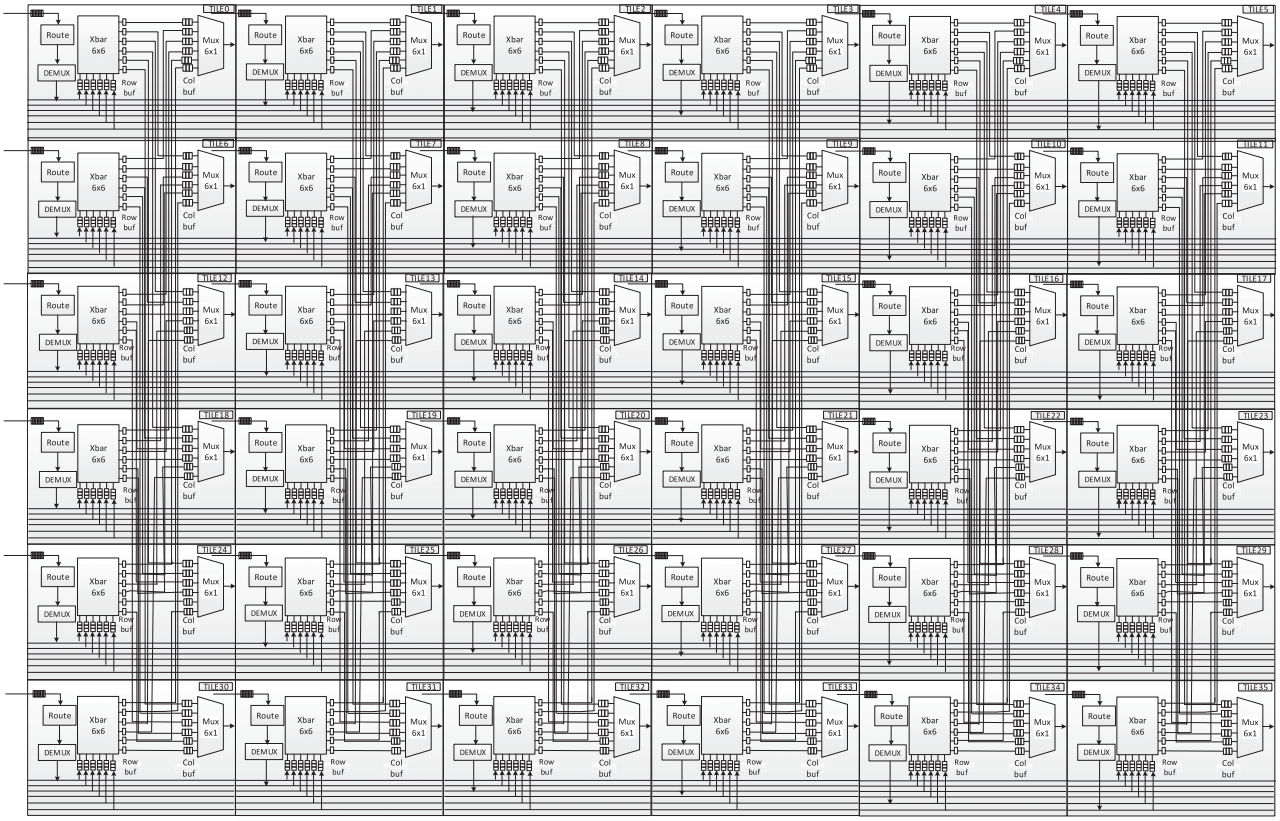

Fig. 1. Example of a radix-36 YARC router with 36 tiles organized as a 6×6 matrix.

the arbitration logic and wire complexity compared with a conventional crossbar organization. Fig. 1 shows an example of a radix-36 router similar to YARC, with 36 tiles organized as a 6x6 matrix. Each arriving packet from a high-speed link is first buffered at the input buffer and then dispatched to the corresponding row buffer via a demultiplexer, following the routing results. As each row buffer is dedicated to one input link, each tile contains 6 row buffers corresponding to the 6 input ports in a row, and 6 column buffers each associated with one output port in a column. Essentially, each demultiplexer is implemented by a virtual-channel (VC) arbiter, choosing one packet from multiple VC buffers to route to its destined row buffer. The subswitch routes the packets from row buffers to their destined column buffers. As shown in Fig. 1, all 6 tiles in the same column are fully connected to each outputs via point-to-point column channels. Packets buffered in the column buffers will be placed on the outgoing line via a 6x1 multiplexer, when their requests are granted. By introducing row buffers and column buffers at subswitches and output ports respectively, the YARC router decouples the input, subswitch and output arbiters and allows them to schedule packets in an independent and asynchronous way, which considerably improves the scalability of the arbitration logic for higher numbers of ports. Moreover, the YARC' s highly symmetrical and regular structure makes it suitable for floorplan and wire routing in chip design.

However, YARC' s structure is still hard to scale to higher radixes because of its high memory requirements and global wiring complexity. As shown in Fig. 1, each tile contains all the logic and buffers associated with one input port and one output port. In essence, YARC replaces an $N \times N$ router with N subswitches of size $\sqrt{N} \times \sqrt{N}$. When port counts increase, the arbitration logic, the memory, and the global wire overhead associated with each tile also increase linearly. On the other hand, the subswitch radix, which dominates the tile' s logic complexity, is determined by the number of ports in the same row and the same column, ultimately derived from the radix N. Each tile' s hardware complexity remains relatively stable no matter how we organize the YARC structure. If we can substantially reduce the number of tiles, the on-chip resource and energy efficiency can be improved considerably. Based on this observation, we propose the MBTR architecture for high radix router design, which is the first attempt to reduce the number of tiles while maintaining high performance in terms of throughput and delay.

2.2 MBTR Architecture

Ref.[3] first proposes the hierarchical crossbar organization to overcome the limitations of conventional centralized arbitration, and further reducing the buffer area with large subswitches. It is worth noting the architectural difference between the YARC and the hierarchical crossbar. YARC shifts the output queues of the subswitches (i.e., column buffers) to their destination output ports instead of scattering them at different subswitches. This considerably simplifies the final output arbitration and avoids global arbitration logic to coordinate subswitches in the same column. Besides the above enhancements taken from YARC, MBTR explores wider design space by using the asymmetrical crossbar, and multiple ports per tile. Hence MBTR can build a $N \times N$ switch with (N/A) subswitches of size $m \times n$, where A is the number of physical ports aggregated

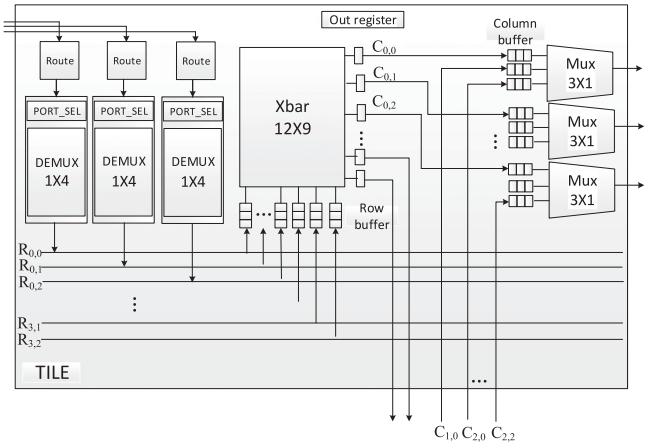

Fig. 2. The structure of multiport binding tile with 3 ports integrated ($R = 3$, $C = 4$, $A = 3$).

in each tile and m, n are the numbers of row and column buffers of the subswitch, respectively. By encapsulating the subswitch and A input and output ports into a tile, a N-port router can be flexibly organized into a $R \times C$ dynamic powertiles, where R is the number of row tiles and C is the number of column tiles. Unlike YARC, $R \times C = N$ is no longer a prerequisite in MBTR. Although the crossbar logic still grows as N^2, the memory overhead increasingly dominating the chip area scales at the rate of $(\frac{m+n}{A})N$ instead of N^2.

Fig. 2 demonstrates the structure of a multiport binding tile with 3 ports integrated. Therefore, a radix-36 router needs only 12 tiles organized as a 3×4 matrix, as shown in Fig. 3. The route unit and demultiplexer are set for each input port, and multiplexers are set for each output port. Each input port has a dedicated row buffer in each tile of the same row, thus there are 12 row buffers in the subswitch. The demultiplexer accesses 4 different row buffers (one per column) via one row bus by asserting the corresponding valid signal. As shown in Fig. 2, the row bus $R_{0,0}$, $R_{0,1}$ and $R_{0,2}$ are respectively dedicated to the 0th, 1th and 2th input port of the tile at column 0. The route unit decodes the packet' s destination port with the row number, column number and output port number of the destination tile. The demultiplexer 1×4 routes the packet to the row buffer based on the row number of the destination tile, and the subswitch routes the packet to the column buffer according to the destined column number and the final output port number. As shown in Fig. 2, the tiles in the same column at row 0, 1 and 2 connect to the 0th output port via the column bus $C_{0,0}$, $C_{1,0}$ and $C_{2,0}$. In this way, each output port is fully connected with all tiles in the same column. After arriving at the head of a column buffer, the packet will be scheduled onto the output link by the 3×1 multiplexer.

Although the MBTR uses a higher subswitch size of 12×9 instead of 6×6 of the YARC, the area of crossbars, demultiplexers and multiplexers mainly composed of combinatorial logic takes less than 24 percent of the core area (including tiles and their interconnection channels). Also, with the development of semiconductor process technology this ratio tends to decrease continuously. The on-die memory and wires between tiles, which can be effectively reduced by the MBTR, in contrast, have a significant proportion of 76 percent of the core area.

By integrating multiple ports into a single tile, MBTR can flexibly organize an N-port switch into $R \times C$ tiles, such that $N = R \cdot C \cdot A$, where R is the number of row tiles, C is the number of column tiles, and A is the number of ports integrated into each tile (clearly, C and R should be divisors of N). Indeed, one can see that YARC is a special case of MBTR when $A = 1$, and $R = C = \sqrt{N}$. MBTR generalizes the idea of YARC by introducing a variable number of asymmetric subswitches to reduce the number of tiles and thus effectively lower the memory consumption and global wiring complexity. As shown in Fig. 3, the radix-36 MBTR with 12 tiles contains 144 row buffers, 108 column buffers, 36 row channels and 108 column channels. This is a reduction of 42 percent in buffers and 50 percent in wire area compared with the YARC router shown in Fig. 1. Also, MBTR facilitates configurable design parameters to make the tradeoff among the subswitch radix, the power and area consumption.

To elaborate the above design concepts, we now present a general description of the MBTR architecture.

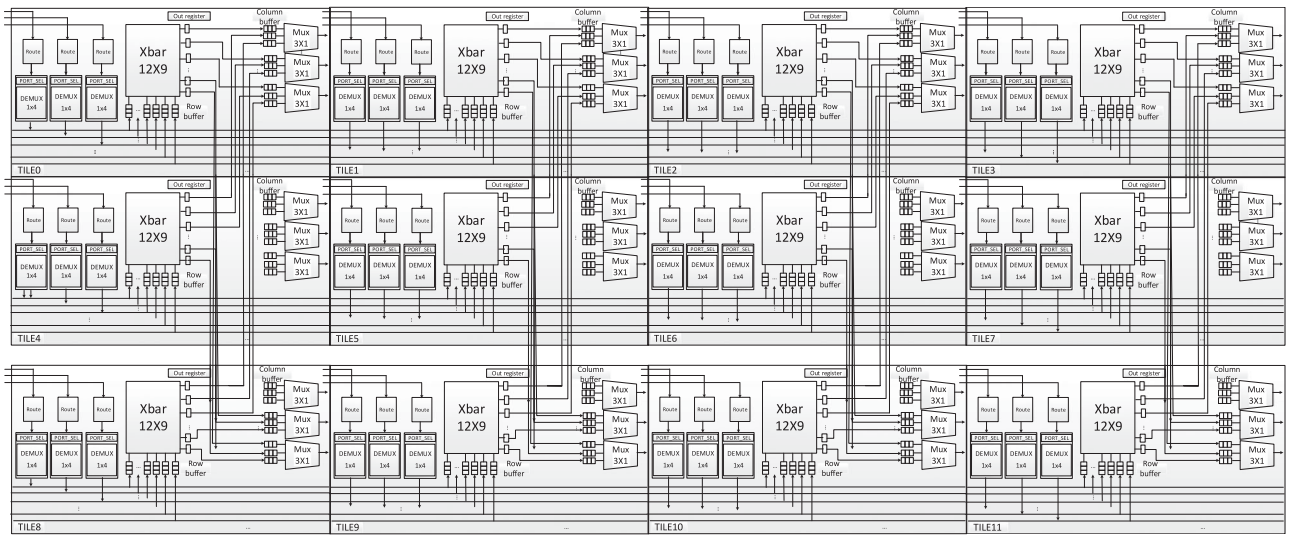

Fig. 3. Example of a radix-36 MBTR architecture with 12 tiles organized as a 3×4 matrix (R=3, C=4, A=3).

TABLE 1
Buffer and Wire Consumption of MBTR with Different Design Parameters

No.	Design Parameters			Subswitch Radix	Buffer Area	Wire Area
	A	R	C			
1	1	8	8	8×8	1,024	32,768
2	4	4	4	16×16	512	16,384
3	4	2	8	32×8	640	8,192
4	4	8	2	8×32	640	32,768
5	8	2	4	32×16	384	8,192
6	8	4	2	16×32	384	16,384
7	16	2	2	32×32	256	8,192

1) The number of subswitches, or tiles, in a row and column are R and C, respectively, yielding $R \cdot C$ tiles/subswitches. There are A input/output physical ports integrated in each tile.
2) The configuration of the MBTR subswitch includes $C \cdot A$ input ports or row buffers, $R \cdot A$ output ports or column buffers, and the arbitration logic.
3) All the $C \cdot A$ input ports in the same row are connected to the corresponding row buffer of each tile in the same row via row buses. Thus, a $1 \times C$ demultiplexer is set at each input port to dispatch the incoming packet to C row buffers, one for each tile following the route results.
4) All the subswitch output ports in the same column destined to the same output are connected to the output via a corresponding column bus. Thus, we have a $R \times 1$ multiplexer at each output to transfer packets from different tiles in the same column to the output.

MBTR is composed of $R \cdot C$ identical tiles that are interconnected symmetrically, making it suitable for routing and floorplanning in an ASIC implementation. To sum up, the MBTR structure has $(R \cdot C)(C \cdot A) = N \cdot C$ row buffers and $(R \cdot C)(R \cdot A) = N \cdot R$ column buffers (since $R \cdot C \cdot A = N$), N row buses, and NR column buses. Therefore, the MBTR subswitch buffers scale at the rate of $N(R+C)$, and the wire area scales at the rate of N^2R. Since the chip area is mainly dominated by SRAM and global buses (or channels), MBTR achieves much better energy efficiency than YARC.

In contrast, YARC partitions a single N-port switch into $N(\sqrt{N} \times \sqrt{N})$ subswitches, requiring $N\sqrt{N}$ row buffers and $N\sqrt{N}$ column buffers, for a total of $2N\sqrt{N}$ buffers. In addition, YARC has N row buses, or horizontal broadcast channels, and $N\sqrt{N}$ vertical column channels such that all N output ports can be connected to all of the $\sqrt{N}$ tiles in the same column. Since row wires and column wires are orthogonal to each other, wire area is generally estimated by multiplying the number of row channels and the number of column channels [8]. As a result, the area of the horizontal and vertical wires dominates the router area with a radix N, which scales at the rate of $N^2\sqrt{N}$. These all lead to area and power inefficiency.

Table 1 demonstrates the hardware overheads of the radix-64 MBTR with different design parameters. With the value of A increasing, the subswitch radix increases but the buffer and wire requirements decrease considerably. When $A = 4, R = 4, C = 4$, overall buffer and wire resources

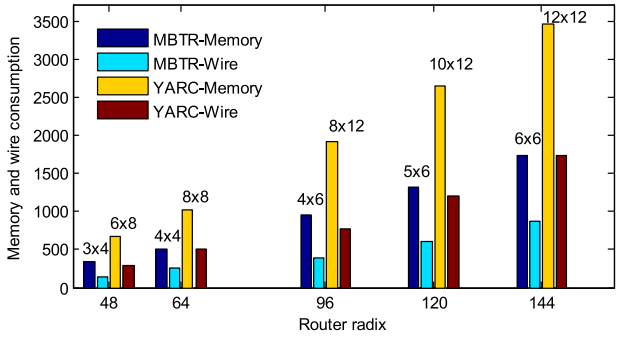

Fig. 4. Memory and wire consumption with the increasing of radix.

can be reduced by half compared to YARC (recall that YARC is a special case of MBTR where $A = 1, R = C = \sqrt{N} = 8$). Commercial routers often rely on high-speed SRAM to match the ultrahigh off-chip bandwidth. However, the area ratio of SRAM over logic increases due to poor array density of SRAM. For example, wires, SRAM and logic in our MBTR-36 respectively take 46, 30, 24 percent of the core area. Thus it is a practical and effective way to reduce the area by using large subswitches that mainly consist of combinatorial logic, and fewer tiles. For the complexity of wiring, the number of row channels is constant N, and the number of column channels is determined by the number of tiles in a column. Therefore wire area can be reduced by decreasing the value of parameter R. As shown in Table 1, all the MBTR configurations with the same value of R will have the same wire area. According to the analysis in [7], decreasing the number of subswitch output ports by reducing R, in turn, will affect the switch throughput by aggravating output port contention. It is very important to make a tradeoff between MBTR parameters to meet different design requirements.

The memory and wire overhead is also evaluated with the increasing of radix N. For simplicity, the parameter A is fixed to 4 and the radix N varies from 48 to 144. MBTR and YARC can be organized into different tile matrix for different radix as shown in Fig. 4. The total number of row and column channels is measured as the wire consumption. Also, a smaller R is preferred for both MBTR and YARC to reduce the number of column wires. For example, a tile matrix of 5×6 and 10×12 is respectively chosen for MBTR and YARC with the radix of 120, instead of 6×5 and 12×10. As shown in Fig. 4, the memory and wire overhead of YARC increases by a large margin with the radix increasing, however, that of MBTR grows smoothly and suggests about 50 percent reduction compared with which of YARC.

3 Throughput Analysis

By dispatching the input traffic to multiple parallel tiles, the internal congestion and port contention can be effectively relieved in the tiled router, and a higher router throughput can be achieved even with relatively lower subswitch throughputs. This is because the subswitch throughput required to deliver 100 percent throughput can be reduced if the intermediate row buffers are always in non-saturated state. For example, YARC can reduce the subswitch input load to $1/C$, with random output port selection, where C is the number of tiles in a row. Therefore, it would seem intuitively reasonable that since the subswitch throughput at

most equals to its maximum input load, a locally relative 100 percent throughput could be achieved, when the packet injection rate is lower than the theoretical throughput of the subswitch. We prove this non-saturated relative 100 percent throughput theory in this section and further deduce the necessary and sufficient conditions that should be satisfied by MBTR parameters to achieve 100 percent throughput. The theoretical analysis results demonstrate that reducing the number of tiles might not affect the switch parallelism if the relative 100 percent throughput is achieved by subswitches.

With the multi-stage parallel switching introduced in high-radix router design the utilization of the input queues is usually less than 100 percent and even no more than 50 percent. So we wonder what our design target should be when the input queues become non-saturated. In this section, we propose a relative throughput theory for parallel non-saturated switches and theoretically prove that both YARC and MBTR achieve 100 percent throughput with the randomly selected output ports. To proceed further we have the following definitions.

Definition 1 (Saturated and non-saturated state). *A switch is said to be in saturated state if there are always packets waiting in every input queue, which means whenever a packet is transmitted through the switch a new packet immediately replaces it at the head of the input queue, otherwise a switch is in non-saturated state.*

Definition 2 (Saturated throughput). *The throughput (i.e., the utilization of output ports or the fraction of time the output ports are busy) of a switch under saturated state is its saturated throughput.*

However, this ideal 100 percent throughput is hard to obtain. Input queued (IQ) switches using only *N* memories at the input ports for radix *N*, have a constrained throughput of 58.6 percent due to the effect of HOL blocking [13]. The virtual output queuing (VOQ) technique (using a dedicated input queue for each output port at each input port) can eliminate HOL blocking and achieve 100 percent throughput, but with memories increasing quadratically with the number of ports (N^2). Output queued switches using *N* memories at the output ports, achieve 100 percent throughput but require the memory and crossbar logic to run *N* times faster than the input link. As the number of ports increases, VOQ and OQ solutions become too expensive and infeasible to implement in hardware. Most of the commercial switches and routers are based on IQ switches, the subswitch of MBTR also adopts IQ structure to reduce hardware complexity.

Since the utilization of output ports will, in turn, determine that of input ports, the design target of a high-throughput switch is to achieve the largest possible output utilization in order to enable the utilization of each input link to reach 100 percent. Actually all throughput analysis is based on a saturated state assumption to approximate the optimal output port utilization or throughput. For example, M. J. Karol et al. assume all input queues are saturated and output requests are randomly selected, and prove the output port utilization converges to the asymptotic 0.586 for an input queued switch with large *N*[13]. We propose the definition of saturated throughput in order to contrastively analyze non-saturated, relative, 100 percent throughput when input queues are not saturated in the parallel switch architecture. Similarly in the following proof we assume that packet arrivals on each input link are governed by independent and identical Bernoulli processes. Each packet, composed of flits, has equal probability of requesting any output ports. Hence, every output port has the same utilization that is equal to the switch throughput. The notation used in the following proofs is described in Table 2.

TABLE 2
Definition of Notations

Notation	Description
N	the number of physical ports of a router
A	the number of physical ports integrated in a tile
R	the number of tiles in a column
C	the number of tiles in a row
M	the average utilization of the input ports
S	the the average utilization of the output ports
T	the ratio of C to R

Lemma 1. *Consider a $p \times q$ Input Queued switch, let ρ_0 be the saturated throughput, then if and only if $M \leq \rho_0 q/p$ we have $S = M(p/q)$ which means the input traffic volume (Mp) is equal to the output traffic volume (Sq), and the switch is under the steady balanced state.*

Proof. Assume this is not true, i.e., when the average utilization of input ports $M \leq \rho_0 q/p$ we have $S < M(p/q)$. Then the input traffic absorbed by the output ports can be calculated as *Sq* less than total input traffic volume *Mp*. Thus, there must be packets accumulating in the input queues, and ultimately input queues become saturated. Therefore, *S* should be equal to the saturated throughput ρ_0, then $M > \rho_0 q/p$ which contradicts the hypothesis of $M \leq \rho_0 q/p$, and we have $S \geq M(p/q)$.

On the other hand, the input traffic absorbed by the output port denoted as *Sq* should not be greater than the whole input traffic *Mp*, thus $S \leq M(p/q)$. We have $S = M(p/q)$ and prove the sufficiency of the Lemma 1.

It' s easier to prove the necessity of $M \leq \rho_0 q/p$ for concluding that $S = M(p/q)$. Assume it is not true, then we have $M > \rho_0 q/p$ which yields $S > \rho_0$ contradicting that ρ_0 is the saturated throughput. □

Definition 3 (Relative 100 percent throughput). *For a $p \times q$ IQ switch, the relative 100 percent throughput is achieved if the input port utilization* M *equals to the possible maximum input load, and the ratio of output traffic volume to the input traffic volume reaches 100 percent, namely, $\frac{Sq}{Mp} = 100\%$.*

Lemma 1 gives a sufficient and necessary condition under which a relative 100 percent throughput is achieved. Particularly, for an $N \times N$ IQ switch, the relative 100 percent throughput could be achieved if $\rho_0 \geq M$. For a $p \times q$ asymmetric *IQ* switch, its saturated throughput should satisfy $\rho_0 \geq Mp/q$ to ensure relative 100 percent throughput. Actually both YARC and MBTR dispatch the input traffic to multiple parallel tiles in the same row, hence each input queue (i.e., row buffer) of the subswitch is always non-saturated, even the input link become saturated. The relative 100 percent throughput theory provide us a guidance to lower the design target of the parallel non-saturated subswitches

without degrading the whole performance. The following proof of Lemma 2 delves the throughput of MBTR and derives the condition that must be met by MBTR design parameters to achieve 100 percent throughput.

Lemma 2. *For an $N \times N$ MBTR organized as a $R \times C$ matrix, 100 percent throughput is achieved if* R *and* C *satisfy*

$$\frac{1}{1+T} \geq 1/C,$$

where $T = C/R$

Proof Sketch We first analyze the subswitch' s output port utilization based on queuing theory and establish the condition under which the relative 100 percent throughput can be achieved by each subswitch. Consequently, the throughput or utilization of column ports can be calculated by Lemma 1. Focusing on tiles in the same column, the 100 percent output link utilization can be aggregated by column ports each from one tile in the same column. As independent, statistically identical traffic arrives on each input link and destination ports are randomly selected, other output link utilization can be inferred similarly. Finally 100 percent system throughput follows from all output link utilization. □

Proof. Each homogeneous subswitch can be denoted as a $p \times q$ IQ switch. We define B_t^i as the number of blocked packets at the heads of row buffers that are requesting column port *i* at clock cycle *t*; I_t^i denotes the number of in-service packets traversing to output port *i* at clock cycle *t*. For a saturated switch we have

$$\sum_{i=1}^{q} I_t^i + \sum_{i=1}^{q} B_t^i = p.$$

Let $E(I), E(B)$ respectively represent the mean number of in-service and blocked packets for a certain output port per clock cycle we get

$$\begin{aligned} qE(I) + qE(B) &= p \\ E(I) + E(B) &= p/q. \end{aligned} \tag{1}$$

Since $E(I)$ is the average number of packets transmitting to the output port per cycle, it is equal to the packet average arrival rate denoted by λ times packet average service time of the output port denoted as $E(S)$, that is

$$E(I) = \lambda E(S) = \rho_0, \tag{2}$$

where the output utilization denoted by ρ_0 is the fraction of time the output port is busy and can be calculated from the arrival rate and the service time.

The blocked packets waiting in the buffer destined to the same output can be treated as customers waiting in the queue to be serviced, which is a typical $M/G/1$ queuing model, and the mean queue length $E(Q)$ is equivalent to the mean number of blocked packets $E(B)$. By the theory of $M/G/1$ queuing model in [14] we have

$$\begin{aligned} E(B) &= \lambda E(W) \\ E(W) &= \frac{\rho_0}{2(1-\rho_0)}(C_S^2 + 1)E(S) \\ E(B) &= \frac{\rho_0^2}{1-\rho_0}\frac{C_S^2+1}{2}, \end{aligned} \tag{3}$$

where $E(W)$ denotes the mean waiting time for an $M/G/1$ queue, and C_S^2 is the squared coefficient of the service time. In the case of deterministic service time $C_S^2 = 0$, we have

$$E(B) = \frac{\rho_0^2}{2(1-\rho_0)}. \tag{4}$$

Since $T = AC/AR = p/q$, substituting the right-hand side of (2) and (4) into (1), we obtain

$$\rho_0 + \frac{\rho_0^2}{2(1-\rho_0)} = T$$

$$\rho_0 = (T+1) - \sqrt{T^2+1} \qquad (\rho_0 \leq 1), \tag{5}$$

when $T = 1$ for symmetric switches, $\rho_0 = 2 - \sqrt{2}$, this result is consistent with the saturated throughput of $N \times N$ IQ switch deduced in [13].

However, deterministic service time means the time taken to switch each packet is constant, which implicitly assumes that all the packets have the same size and the port allocation is done at packet level. Hence Ref. [13] conducts the proof based on the assumption of synchronous switching with fixed-length packets. That is impractical in most high-performance routers, in reality, variable-length packets are segmented into flits or cells. The flits are not independently routed, as each payload flit follows the same path as its corresponding head flit like VCT switching or wormhole switching. Therefore, the service time of each output follows exponential distribution, then $C_S^2 = 1$ according to Ref. [14], which yields

$$E(B) = \frac{\rho_0^2}{1-\rho_0}. \tag{6}$$

By substituting the right-hand side of (2) and (6) into (1), we obtain

$$\rho_0 + \frac{\rho_0^2}{1-\rho_0} = T \tag{7}$$

$$\frac{\rho_0}{1-\rho_0} = T$$

$$\rho_0 = \frac{T}{1+T}. \tag{8}$$

As MBTR distributes packets to *C* parallel tiles in the same row the maximum input load of subswitches should be equal to $1/C$ when the utilization of external input links reaches 100 percent. From Lemma 1, we know if and only if

$$M \leq \rho_0 q/p, \tag{9}$$

the input traffic volume will equal to the output traffic transmitted through the subswitch and the non-saturated relative 100 percent throughput is achieved. Since the maximum input utilization $M = 1/C$, substituting the right-hand side of (8) into (9), we have

$$\frac{1}{1+T} \geq 1/C, \tag{10}$$

be the necessary and sufficient condition under which the maximum input traffic buffered at row buffers of the subswitch equals traffic destined to column buffers. Fixing our attention to MBTR tiles in the same column, the total input traffic destined to all output links in the same column comes from row buffers of all subswitches in the same column that can be calculated as

$$\frac{Rp}{C} = R \cdot A \cdot 100\%, \tag{11}$$

where A is the number of ports integrated in a MBTR tile, and there are up to $R \cdot A$ output links in the same column. From Lemma 1, since each subswitch achieves relative 100 percent throughput the utilization of the column output ports should be $p/Cq = 1/R$. Each column output is connected to its column buffer in a point-to-point way thus the input port utilization of the multiplexer $R:1$ will be $1/R$. Note that each column buffer is essentially an output queuing buffer as all buffered packets are destined to the same output. Hence the MBTR' s final multiplexer $R:1$ is an output queue (OQ) switch in which there is no HOL blocking between packets. Since the multiplexer' s output link bandwidth is R times the input port utilization $1/R$. The throughput of the output link is 100 percent so as to deliver 100 percent traffic from all R column buffers. As a result, summing up to $R \cdot A$ output links in the same column deliver all $R \cdot A \cdot 100\%$ input traffic as shown in the Equation (11). The same conclusion can be inferred for other column of tiles. Consequently, 100 percent throughput is achieved for MBTR when the Equation (10) is satisfied and Lemma 2 is proven. □

We have the following important conclusions from Lemma 2. First, YARC architecture with $C = R = \sqrt{N}$ can achieve 100 percent throughput under random output port selection. Second, for MBTR architecture with $C \geq 3$, 100 percent throughput can be achieved as long as $T \leq 2$. However, the ideal traffic model assumed for theoretical analysis is impractical in the real network. MBTR essentially provides a relative internal speedup when the subswitch become unsaturated. According to our experience in MBTR parameter exploration, a relative speedup of two could be sufficient to provide competent performance under uneven traffic, because at which a CIOQ switch (that refers to the IQ switch with speedup) can precisely mimic an OQ switch under identical inputs [15]. Take our design MBTR-36 for example, the subswitch provides a relative speedup of $\frac{C}{1+T} \approx 1.7$, and the simulation results demonstrate MBTR is indistinguishable from YARC in terms of throughput and delay even under hotspot and exponential traffic model, but with much lower memory and wire consumption. YARC provides a relative speedup of $\frac{\sqrt{N}}{2}$ that is over provisioned and infeasible for large N.

By combining above theoretical results with practical experience, we summarize the following design criteria for the MBTR parameter selection.

1) The hardware complexity of the $(A \cdot C) \times (A \cdot R)$ subswitch should be given primary consideration. Although the hierarchical and multi-stage schedule technology can implement a higher radix arbiter [16], [10] such as radix 32, its logic would require a much longer clock cycle or a pipelined structure. The main reason that YARC and MBTR can achieve 100 percent throughput instead of 58 percent constrained by HOL blocking, is to distribute traffic to multiple parallel subswitches, thus mitigating the output congestion. When increasing the parameter C, the switch parallelism could be improved, meanwhile the decreased input load helps to reduce the required saturated throughput of subswitches according to Lemma 1. On the other hand, increasing the parameter R (i.e., increasing the value of q) could improve the input utilization (referring to the Equation (9)) by reducing output port contention. In practice, the available maximum radix of the arbitration that meets the frequency and performance requirements will largely determine the number of tiles that can be integrated in a row and a column.
2) The total number of tiles should be controlled carefully. Dividing a high-radix router into too many small tiles not only increases the power and area, but makes floorplanning and routing more difficult. As discussed Section 2.2, the number of subswitch buffers is determined by the total number of tiles, so integrating more ports in one tile will reduce the total number of tiles and is always beneficial for reducing memory requirements. Besides, arbitration complexity, appropriate parallelism and Lemma 2 should be taken into consideration.
3) Slightly adjust the values of R and C under a given number of tiles. As each output port is fully connected to R tiles in the same column, reducing R would be an easier option to reduce column wires, but might incur potential congestion at subswitches. However, Lemma 2 suggests slightly decreasing R to reduce wire area might not affect the ultimate throughput, since the row buffers are unsaturated. Also, from the floorplanning and practical perspective, the value of R and C should be very close to each other.

As shown in Table 1, by integrating four ports in one tile, the 2nd configuration partitions a 64-radix router into several 16×16 middle-scale subswitches and halves the buffer and wire overhead. In this case, each input is dispatched to four tiles in the same row, and the packet injection ratio at each row buffer is nearly 25 percent, thus the relative speedup is about 2. This configuration can not only achieve the relative 100 percent throughput, but also has reasonable hardware complexity and proper tile numbers, meeting the design criteria 1 and 2. In contrast, the 3rd configuration saves wire area considerably with $R = 2, C = 8$, but it violates the criteria 3 as the 32×8 sub-switch deteriorates output contentions and the large gap between the value of R and C is always adverse to the practical floorplanning. By further coordinating the value of R and C, stricter buffer and area constraints can be satisfied. Although the 5th configuration with R=2, C=4 reduces the wire area by half compared with the 6th configuration, it degrades the saturated throughput of subswitches from 67 to 33 percent due to aggravating output port contention. Finally, the last one with $R = 2, C = 2$ reduces buffer and wire areas to the minimum, which is only 25 percent of those required by YARC. However, it is

not recommended because the hardware complexity of 32×32 sub-switches is sharply increased. Besides, the saturated throughput of 32×32 switches equals to the row port utilization of 50 percent, and no speedup is provided.

4 Optimization Approaches

In this section we present some crucial optimization approaches for the hardware implementation of MBTR including credit management algorithm, reliable data transmission, flow control and packet arbitration structure. These schemes achieving outstanding performance, but with relatively lower hardware overhead, are full of practical value for economical IC implementation.

4.1 Credit Management Mechanism

Most router designs adopt multiple, parallel virtual channels in order to prevent protocol-related deadlock, to provide express channels for multi-priority traffic, and to reduce system delay by allowing blocked packets to be passed. Since the VC buffers in the router consume a significant amount of the dynamic and leakage power [17], they impact both network throughput and power.

To reduce the buffer space requirement without degrading the throughput of the router, shared buffer management mechanisms like dynamically allocated multi-queue (DAMQ) are widely used to improve buffer utilization. We propose a fair credit management scheme that can efficiently assign buffers on demand among bursty or uneven VCs. The main idea of our buffer management scheme is to divide the data space into two parts, private buffers (PB) that can only be used by a particular VC and shared buffers (SB) that can be used by all VCs. Each VC' s private buffer is assigned to ensure that starvation would not occur, even when some greedy VCs occupy most of the shared buffers. Considering different VC has different maximum packet length, the depth of private buffer can be configured by configuration registers. Thus we have $DB_{depth} = SB_{depth} + \sum_{i=0}^{v-1} PB_{depth}^{VC_i}$ where DB_{depth} is the depth of the whole DAMQ, SB_{depth} is the depth of shared buffers, v is the number of VCs and $PB_{depth}^{VC_i}$ is the depth of private buffers assigned to VC_i. It is noticeable that SB and PB is a logical concept that represents the usage attribute of buffer units and their locations in DAMQ are random and dynamic instead of fixed and static. We define fairness as each VC being allocated shared buffers on demand as much as possible which is different from equal partition among competing VCs. In order to prevent greedy VCs from overusing shared buffers, the shared credits used by each VC are counted by PSC_{VC_i} and compared with a threshold value $PSC_{threshold}^{VC_i}$ denoting the maximum number of shared credits (SC) occupied by VC_i. Once PSC_{VC_i} is greater than its threshold $PSC_{threshold}^{VC_i}$ the released credit of VC_i will be added to SC, otherwise it is added to its private credit denoted as PC_{VC_i}. As the VC request should not be blocked if the shared or private credit is available, this strategy could not prevent the greedy VC from using shared credits instantly. However, in a long term, those used credits that exceed the threshold, in turn, will be ultimately released and shared by all VCs again. By dynamically allocating the returning credits between private and shared buffers (referring to the credit accumulation described below), we implement on-demand allocation of shared buffers, further improving the DAMQ utilization and throughput.

For credit-based flow control, the size of a VC queue is determined by how its credits are accumulated and consumed. When VC_i has a flit to send, it prefers using shared credits first. The rules for credit consumption for VC_i are as follows:

1) If $SC > 0$ which means the shared buffer has available space to receive a flit, then send this flit meanwhile decreasing *SC* and increasing PSC_{VC_i} by one;
2) Else if $SC = 0$ and $PC_{VC_i} > 0$ which means there is no available shared buffer but the private buffer assigned to VC_i is available to receive a flit, then send this flit meanwhile decreasing PC_{VC_i} by one;
3) Else if $SC = 0$ and $PC_{VC_i} = 0$, block this flit in buffer.

When a flit of VC_i is read out from DAMQ, a credit-release signal will be sent to the sender, and the credit accumulation process for VC_i is described as following:

1) If $PSC_{VC_i} < PSC_{threshold}^{VC_i}$ and $PC_{VC_i} < PB_{depth}^{VC_i}$ the released credit should be added to PC_{VC_i} by increasing PC_{VC_i} by one;
2) Else if $PSC_{VC_i} < PSC_{threshold}^{VC_i}$ and $PC_{VC_i} \geq PB_{depth}^{VC_i}$ release one credit to *SC*.
3) Else if $PSC_{VC_i} \geq PSC_{threshold}^{VC_i}$ add one credit to *SC* and decrease PSC_{VC_i} by one.

This credit accumulation scheme guarantees the fairness among competing VCs by limiting each VC' s occupancy of the shared buffers in a reasonable range. The simulation results show that in various scenarios, such as unbalanced VCs with 25 percent of the VCs occupying 80 percent of the traffic, and bursty VCs with a maximum bursty length of 3 to 6 packets, the MBTR almost maintain the same throughput and delay as that of random VC distribution (not presented in Section 5). This verifies our credit management scheme can effectively eliminate the negative effect of uneven and bursty VCs on buffer utilization.

4.2 MBTR Chip Pipeline Structure

In this section, we describe the packet processing of MBTR in detail based on per-cycle behavior of the pipeline. Fig. 5 shows the pipeline structure of MBTR, consisting of three main procedures. First, the demultiplexer dispatches incoming packets to the corresponding row buffers associated with their input ports. Second, the subswitch schedules the packets from row buffers to their destination output ports. Finally, packets arrive at column buffers via column channels and wait to be transferred to the final output by the multiplexer.

To support robust data transmission with error correction and checking capabilities, an error correction code (ECC) is coupled with each DAMQ for single error correction and double error detection. As shown in Fig. 5, ECC is only generated at the first input DAMQ (ECCG module), and the remaining row and column DAMQs only check the ECC and report error states (ECCC module), which effectively reduces the critical path delay of the entire pipeline.

As shown in Fig. 5a, each input port is associated with a route table which can be configured and read by a configuration and state register (CSR) agent. Before arriving packets

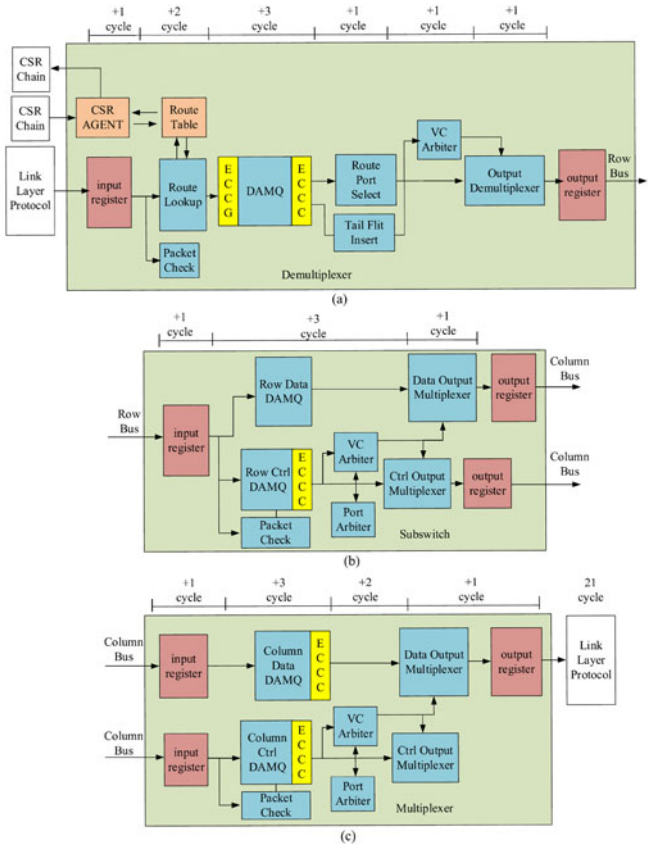

Fig. 5. MBTR pipeline structure of (a) Demultiplexer, (b) Subswitch and (c) Multiplexer.

are buffered in the input DAMQ, their destination output ports are determined by route lookup results. As an incomplete packet, such as one without a head or tail, will interfere with arbiter processing and even lead to the pipeline stall. Packet Check and Tail Flit Insertion blocks designed to guarantee the packet integrity. For example, when some fatal errors happen at the link layer and the corresponding packet buffer needs to be emptied to restore the link. In this case, the link layer asserts this exception to the router, the Flit Insertion block will check the integrity of each arriving packet after the exception assertion, and insert the body and tail flits for the incomplete packet. In order to make full usage of the path diversity provided by high-radix networks, a configurable route table is set for each input to support adaptive routing constructed by the software. The route lookup results, together with a set of available output ports, will be written to the input DAMQ for future route processing. The Route-Port-Select module decodes the internal routing information for each packet, including the row number, column number and final output port of the destination tile. Then the VC arbiter will select a packet from the DAMQ in a round-robin manner, and the output demultiplexer activate the corresponding valid signal of the row bus to connect the packet to its row buffer. Since each row buffer is associated with only one input port, simple flow control will be sufficient, with no need for port arbitration to allocate these buffers. Being beneficial from the asynchronous and independent scheduling, the per-VC credit counters of the destined buffer is retrieved locally by the distributed VC arbiters, rather than central credit maintenance and packet scheduling.

At the second stage, the subswitch schedules the packets buffered in the row buffers to their destination column buffers. As shown in Fig. 5b, each port includes a packet-check block to examine packet exceptions such as hop count overflow, VC cross error and blocking or starvation state of each VC' s packet. The internal routing and QoS related information are stored in the control DAMQ, each corresponding to the packet buffered in the data DAMQ. An ECC check (ECCC) block is coupled with each DAMQ to detect ECC errors and report ECC state to a DAMQ state register. To schedule these multi-VC shared buffers, a two-level arbitration is implemented by VC arbiters connected with port arbiters. Each VC arbiter selects one request among multiple VCs packet requests to participate in its destined port arbitration, finally the port arbiter will grant one of the requests from all source VC arbiters. This two-level arbitration can be implemented within a single 700 MHz clock cycle. The grant signal is delivered to both the data and control output multiplexers, so the granted packet' s control information as well as the packet itself are transferred to the destination column buffers.

In the third stage, shown in Fig. 5c, the final output multiplexer places packets from column buffers on the outgoing line. Essentially, each column DAMQ is associated with a VC arbiter to generate the output request to the port arbiter. Finally, the port arbiter grants one packet to be delivered to the output link via the multiplexer. As shown in Fig. 5, the number of cycles taken by functional components is marked on top. The demultiplexer, subswitch and multiplexer take up to 21 cycles for a flit traversing through the router pipeline, i.e., 30 ns fall-through latency with 700 MHz frequency. As for the delay experienced in the physical and link transceiver, it takes about 18 ns in the link layer and 36 ns in the physical layer regardless of the code and decode delay for forward error correction (FEC).t

4.3 Flow Control Scheme

YARC uses wormhole flow-control internally and VCT flow control on external links due to buffer size constraints. However, the wormhole is more likely to block the packet and adds complexities to deadlock-free flow control [19]. Considering VCT' s advantages of higher routing flexibility and better use of virtual channels, we prefer VCT over the wormhole for lower network delay.

To implement VCT flow control, only those packets with enough credits for their downstream buffers are eligible to participate in the VC arbitration. The VC arbiter checks the availability of credits for the whole packet when the head flit requests corresponding output port, and the grant signal will keep valid until reading out the tail flit. This scheme guarantees the continuous scheduling of a whole packet and no packet will be blocked due to the lack of credits. Moreover, after one arbitration, the granted VC' s priority is adjusted to the lowest, and the next round contention will be started over according to the updated VC priority. As discussed in [3], an ideal VC arbiter would allow all input VCs to monitor the available status of all output VCs they are waiting for. Such an arbiter would be prohibitively expensive, with N^2v^2 wiring complexity [3]. In contrast, we implement a credit management module coupled with each DAMQ to maintain available credits with Nv wiring

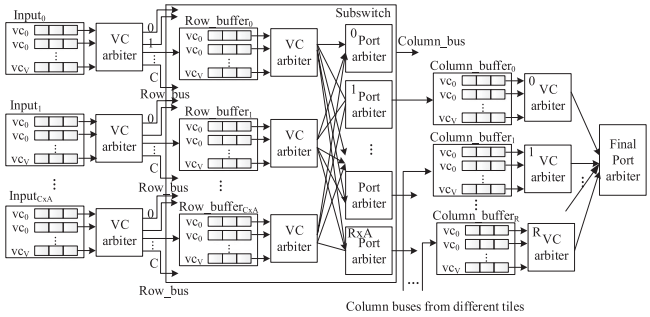

Fig. 6. MBTR arbitration structure. The input VC arbiters associated with each input DAMQ; subswitch arbiters consisting of VC arbiters coupled with each row buffer and port arbiters associated with each subswitch output port; and VC arbiters coupled with each column buffer and the final port arbiter corresponding to each output link perform arbitration independently and asynchronously, thus limiting wire and arbiter complexity.

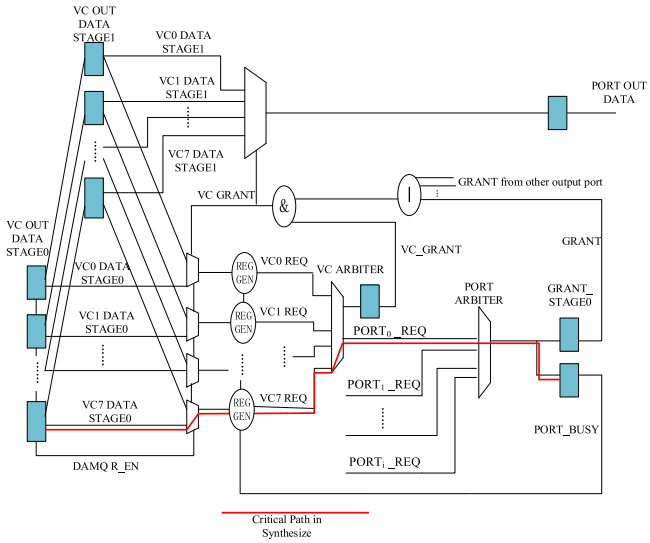

Fig. 7. Block diagram of two-level arbitration logic.

complexity. Each arbiter at different switching stage works independently and asynchronously as shown in Fig. 6. The packet arbitration is performed based solely on the credit status of the downstream buffer instead of global buffer occupation info, thus effectively simplifying the complexity of credit monitoring.

4.4 Two-Level Arbitration Structure

Due to the addition of row buffering at subswitches and column buffering at the output multiplexers, arbiters in a tile-based router architecture are decoupled into three stages: input arbiter, subswitch arbiter and output arbiter. This decoupling simplifies the complexity of centralized arbitration, but it also incurs additional delay from the distributed arbitrations. We have to overcome this potential performance penalty.

The MBTR arbitration structure is shown in Fig. 6. The input arbiters, subswitch arbiters and final output arbiters work in an asynchronous and independent way. Because each row buffer is dedicated to one input port, there is no port contention among input ports. The $C \cdot A$ VC arbiters, each coupled with an input DAMQ, will schedule all incoming packets to their corresponding row buffers.

The subswitch arbitration consists of $C \cdot A$ VC arbiters each associated with one row buffers, and $R \cdot A$ port arbiters each corresponding to one output port of the subswitch. The whole arbitration process can be done within one clock cycle and guarantee fairness among multiple VCs. No packets will be blocked by any other VC' s packet as long as every incoming packet is complete. First of all, the link layer guarantees the packet integrity, MBTR also has many error tolerant mechanisms, such as packet integrity check and packet tail insertion, to insure the packet integrity.

In a subswitch with v VCs, each VC arbiter performs $v:1$ arbitration from v VC queues and sends the granted packet' s request to its destination port arbiter. Then each port arbiter will receive at most $C \cdot A$ requests from all row buffers of the subswitch and grant one among these requests. When packets arrive at column buffers, there are up to R column buffers each from a tile in the same column, destined to the same output link. Similarly, the VC arbiters associated with each column buffer generate valid requests to the final port arbiter. Finally, the final port arbiter selects the packet with the highest priority to access the output link. This two-level arbitration mechanism is actually applied to both the subswitch arbiters and multiplexer arbiters.

Fig. 7 shows the critical path, highlighted by the red line, of the two-level arbitration pipeline after synthesis. In the first stage, the request generation block collects valid VC requests from each VC_OUT_DATA according to the VC' s available credits and sends at most v requests to the local VC arbiter. The VC arbiter chooses one of these requests and forwards it to the corresponding port arbiter, depending on its destination. Each output port is coupled with one port arbiter which receives requests from all correlated VC arbiters and finally grants one VC arbiter. Only when both the grant from the local VC arbiter and a grant from the destined port arbiter are asserted, can the multiplexer read out the granted VC' s packet consecutively until the tail flit. More specifically, once the port arbiter grants a VC request from a certain input, it will invalidate all other input requests destined to itself by feeding back the PORT_BUSY signal to the request generation blocks until the tail flit of the granted packet has been scheduled, as shown in Fig. 7. This scheme keeps ungranted input ports from continuing to request this output port during the transmission of the whole packet. Also, by invalidating the requests that would result in an arbitration failure during the next round of port competition, VC arbiters have to choose other VC requests destined to other output ports. This simple feedback mechanism eliminates the throughput loss resulting from some predictable arbitration failures. The PORT_BUSY signal is registered from the GRANT signal outputted by a port arbiter, this final arbitration result helps VC arbiter speculatively exclude some potentially failed port requests and improve the arbitration efficiency.

The simulation results suggest that our two-level arbitration scheme can effectively improve throughput and achieve comparable performance to the multi-iteration iSLIP algorithm. As shown in Fig. 7, the critical path from DAMQ output data through the VC arbiter, through the port arbiter, and to the final grant can be done within a single 700 MHz clock cycle. In addition to the reduction of critical path delay, another reason to have the arbitration completed within one clock cycle is to guarantee the

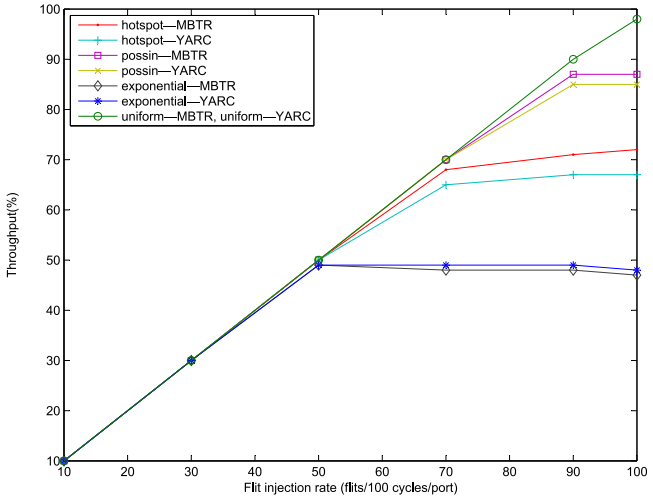

Fig. 8. Throughput of MBTR and YARC under uniform, hotspot and exponential traffic models.

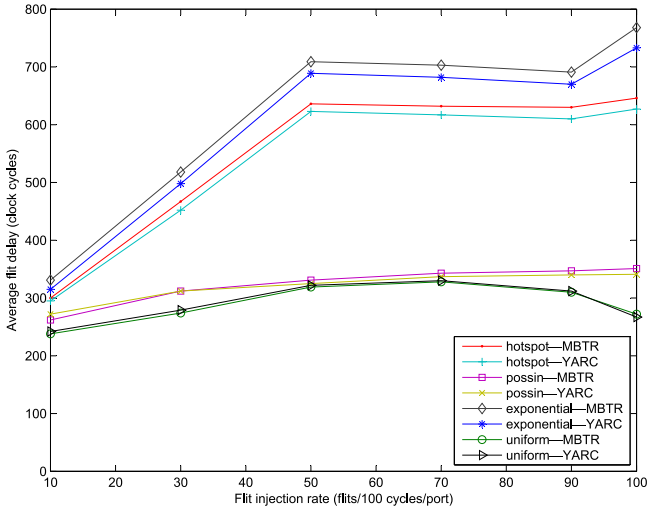

Fig. 9. Average flit delay of MBTR and YARC under uniform, hotspot and exponential traffic models.

promptness and accuracy of arbitration results fed back for speculative arbitration. Because the delayed PORT_BUSY signal might mislead VC arbiters to ignore some valid requests that could be successfully granted. However, our scheme can totally avoid this kind of throughput loss.

5 Evaluation

We implement MBTR in register transfer level (RTL) and conduct cycle-level simulation experiments to evaluate the delay and throughput of MBTR. An independent simulation environment is also developed to verify our two-level speculative arbitration scheme has performance comparable to the multi-iteration iSLIP algorithm [11].

Our simulation models a radix-36 MBTR with $A = 3$, $R = 3, C = 4$. We measure the delay and throughput performance under both uniform and unbalanced traffic. A packet' s delay is calculated from the time that the first flit arrives at an input port until the time that the last flit departs from an output. For the purpose of comparison, a radix-36 YARC model with 6×6 subswitches is also built in RTL, synthesized, and simulated at cycle level. Moreover, some technical enhancements of MBTR, such as two-level arbitration (TLA), increased number of VC to eight and DAMQ buffer, not supported by Cray YARC, are also implemented for YARC model. In a word, except for the architectural difference, all implementation schemes keep the same as each other for a fair simulation comparison.

The throughput and delay performance are evaluated by the cycle-accurate simulation. Each port under measurement is connected to a node model that generates traffic to the switch and mimics the behavior of the link layer, such as credit-based flow control, random physical errors, and link jitter. We generate uniform traffic by allocating destination ports evenly among injected packets. With hotspot traffic, every end node steers half of its load to one third of the output ports covering $[0 : N/3]$. For exponential traffic, the packet destinations follow an exponential distribution.

Fig. 8 shows the throughput of MBTR and YARC under various traffic models, where we can see that they have almost identical throughput. More specifically, their throughputs both reach 98 percent under uniform traffic with full load, but decrease by 27 and 50 percent under hotspot and exponential traffic, respectively. This is because all the packets destined to different output ports are buffered in the same queue for cost-efficient memory usage. The unbalanced distribution of destination ports, which incurs HOL blocking, has great influence on the performance of both YARC and MBTR. However, MBTR outperforms YARC by a small margin under hotspot traffic when the load is higher than 70 percent.

We have similar results in terms of delay performance as shown in Fig. 9. The average flit delay of MBTR is very stable even under heavy load, and demonstrates comparable performance with YARC under uniform and possin traffic model. Especially in uniform traffic, the average flit delay is even reduced with a higher load over 70 percent, as the throughput is always increasing with the input load. However, YARC outperforms MBTR by a tiny margin, under hotspot and exponential traffic model. In this case, although the flit delay increases significantly when the load is over 90 percent, it is still bounded. Unlike the general switch simulation with the assumption of infinite buffer size, and regardless of the lossless flow control, in our experiments, because of the credit-based flow control, the flit injection rate under steady state will actually become the best-effort sending rate at an assigned speed. As a result, when the switch cannot tolerate the load, the flit injection rate will go down since the packets will be blocked in the upstream buffer until the credits are available. Therefore, MBTR and YARC demonstrate bounded latency under full load, due to the backpressure of the flow control along with the limited real buffer size in our switch models.

We further compare the power and area of MBTR and YARC by synthesizing the RTL code in Synopsys Design Compiler (Synopsys DC) with a 28 nm cell library. As the dynamic power is closely relevant with the toggle rate and hard to be evaluated accurately. We use a conservative estimate of 50 percent toggle rate for SRAMs and 20 percent toggle rate for logic. The synthesis report demonstrates SRAMs largely determine the dynamic and leakage power, and MBTR achieves 35 and 33 percent reduction respectively in the area and power overhead.

The main reason we prefer two-level arbitration (TLA) over iSLIP is its simplicity, which can help to achieve higher operation frequency. iSLIP is a centralized scheduling algorithm used to find a maximum match between inputs and outputs [11]. With increasing iSLIP iterations, the wire complexity of the arbitration logic increases and becomes infeasible for hardware implementation. As discussed in Section 4.4, MBTR adopts distributed three-stage scheduling, and TLA is applied twice, to the second and third stages. Being beneficial from the intermediate row buffers and multistage asynchronous scheduling, MBTR demonstrates more stable and higher performance than a single-stage switch according to the experiment results shown later. On the other hand, if we treat the MBTR as a flat switch and apply iSLIP to it, it is hard to quantify how many clock cycles would be needed by a real circuit to complete the centralized iSLIP algorithm with $O(N^2)$ complexity. To make a fair comparison, we construct an independent simulation environment of a flattened switch to evaluate throughput and delay of TLA and iSLIP.

We implement an 8x8 switch with each port containing 8 VCs organized as a DAMQ buffer. To apply iSLIP to this flat switch a 64-to-64 iSLIP algorithm treating each VC as an independent requesting source is implemented with 1, 2, and 3 iterations. We also assume this centralized iSLIP scheduling can be completed within one clock cycle. For TLA, both VC and port arbiters use round-robin scheduling.

Not surprisingly, increasing the number of iterations per round can improve iSLIP performance. As shown in Fig. 10, the iSLIP algorithm with 1 iteration (iSLIP-1) has the poorest performance, TLA achieves delay and throughput performance comparable to iSLIP-2 , and iSLIP-3 has the highest throughput and lowest delay under uniform traffic. We have obtained similar results under hotspot traffic. Although the performance of all scheduling algorithms degrades apparently under hotspot traffic, due to HOL blocking effect. TLA unexpectedly outperforms iSLIP-2 by a small margin and has comparable throughput to iSLIP-3 under hotspot traffic as shown in Fig. 10a. When the number of iterations is increased to 4, iSLIP performance remains almost the same. This is because iSLIP theoretically converges to the optimal performance after 3 iterations.

The area and frequency are also evaluated by Synopsys DC with 28 nm synthetic library. Under the target frequency of 800 MHz, only TLA meets the timing requirement, the negative slack of iSLIP is respectively -197, -933, and -1651 ps for 1, 2, and 3 iterations. Since the clock cycle for 800 MHz is 1,250 ps, the corresponding achievable frequency could be approximately inferred as 600, 400, and 300 MHz. TLA also has 8, 23 and 36 percent less combinational area compared with iSLIP-1, iSLIP-2, and iSLIP-3. In sum, TLA achieves performance similar to iSLIP, but with higher frequency and less hardware cost. It is worth noting that the hierarchical schedule structure like MBTR is more stable than iSLIP as the throughput loss from uniform traffic to hotspot traffic is 26 percent for MBTR, but goes up to 47 percent for iSLIP.

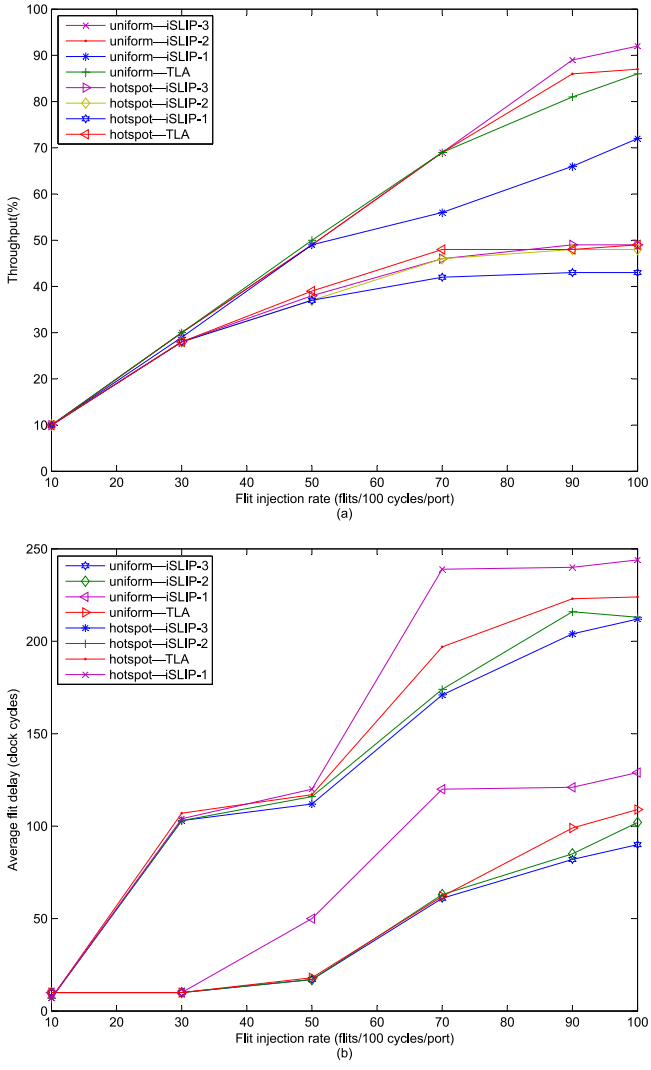

Fig. 10. Throughput and average flit delay comparison of iSLIP and two-level arbitration (TLA) respectively under uniform and hotspot traffic models.

6 ASIC Implementation

We have fabricated the MBTR architecture with 28 nm ASIC technology. This section reports the issues we have encountered at the back-end process stage and how we solved them. There are eight independent VC buffers organized as a shared DAMQ to effectively reduce the memory consumption. The multi-VC shared memory space is logically divided into a private part and a public part. Private space can only be occupied by a specific VC, while the public space is shared by all VCs. Reserving private space for each VC guarantees that memory would not be completely occupied by greedy VCs, also, each VC would not be blocked by any other VCs. On the other hand, for a credit-based flow control, each VC needs to hold at least one packet. Besides, the credit RTT depth that means the number of clock cycles taken from credit usage to new credit arrival should be reserved to ensure completely pipelined processing without being stalled for lack of available memory. Therefore, the minimum private buffers for each VC have to hold one longest packet plus the credit RTT depth. As a result, the size of private space is 8 packets plus sum of RTT depths for full pipeline. In our design, public memory occupies about $1/3$ of the private space. As a result, each row/column DAMQ has a size of about 12 longest packets for 8 VCs. The DAMQ distribution in each tile is shown in Fig. 11a.

The MBTR chip floorplan is shown in Fig. 11b. The high-speed SerDes cores widely used for off-chip interconnection

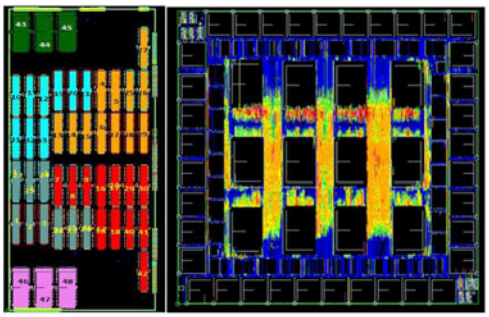

Fig. 11. SRAM distribution (left) and floorplan (right) of the MBTR chip.

are distributed around the perimeter of the chip so they can be physically close to the I/O pads. In order to avoid long wires between tiles and link layers, each tile chooses the closest ports to interconnect. Due to the limited area for routing global wires, the row and column channels become crowded to excess as shown in Fig. 11b. Besides, the longest transmission distance for high-speed wires running at 700 MHz is restricted to 3 mm without any relay. In order to overcome the distance limitation of long wires, we register all input and output signals of each tile and relay all interface signals inside the tile itself instead of scattering relay logic between tiles. In this way, each tile only has interconnection wires with its neighboring tiles. As shown in the chip floorplan of Fig. 11b, the MBTR chip area is mainly dominated by SRAMs and the area routing global wires, which take 76 percent of the core area (including tiles and their interconnection channels). Due to the reduction of 42 percent in SRAMs and 50 percent in wire area, MBTR outperforms YARC in the area overhead by a 35 percent reduction.

In terms of the energy consumption, MBTR consumes around 128 Watts including 98 Watts dynamic power and 30 Watts leakage power. For the dynamic power of 98 W, 73 W about 72 percent is consumed by the 144x25 Gb/s SerDes cores, or approximately 0.5 W per link, $0.5\,W/25\,Gb = 20\,pJ/bit$. SRAMs used for buffers and clock network only respectively occupy 4 and 16 percent of the total dynamic power. Therefore, for 28 nm process, SerDes cores mainly dominate the energy consumption. However, as semiconductor process technology scales from 14 nm today towards 7 nm and beyond, leakage power becomes a substantial portion of the total power [2]. The leakage power distribution shows SRAMs occupy 47 percent of the total leakage power in contrast of 17 percent consumed by SerDes cores. On the other hand, the MBTR logic including combination logic and sequential logic has surprising 38 percent of total leakage power. As a result the memory and logic consumption will play more and more important role in power consumption in the near future. From this perspective, MBTR structure will be beneficial to higher energy efficiency than YARC by effectively reducing the number of tiles.

Although SerDes bandwidth and semiconductor process capacity of 28 nm almost doubled compared with those of 45 nm, the number of high-speed pins surprisingly decreased. Because the increased power consumption demands more Vdd (Voltage drain drain) pins for power-supply, and higher performance needs more ground (abbreviated "Gnd") pins. Hence it becomes very difficult to package more high-speed IO pins within a single chip. It could be predicted that for 28 nm process, 10 Tb/s of throughput and 170 W of power might reach the highest limitation of SerDes-based interconnection networks. Another issue incurred by high-speed SerDes is the transmission reliability. With a 45 nm process and 14 Gb/s (per lane) bandwidth, the SerDes bit error rate (BER) is about 10^{-15}, but increases by two orders of magnitude to 10^{-13} under 28 nm process with 25 Gb/s (per lane) bandwidth. Moreover, the forward error correction (FEC) adopted in the 25 Gb/s SerDes to improve the physical transmission reliability, bring about 200 ns code and decode delay compared with 14 Gb/s singnaling.

7 Comparison with the State-of-the-Art

In this section, we briefly review the related work on high-radix interconnection network design, then compare our design with the state-of-the-art, especially IBM' s latest SCOC switch, and conclude with our future research plan.

Interconnection network design for modern high-performance systems, such as exascale, is becoming increasingly challenging due to the requirements of low latency, high bandwidth, low power consumption, and low cost. With the interconnection switches dominating the efficiency of the interconnection network, high-radix routers have emerged as a promising solution for such challenges.

With the pin bandwidth increasing from 10 to 25 Gb/s and 50 Gb/s [18], and semiconductor technology scaling from 28 nm towards 7 nm and beyond, it becomes feasible to integrate more components such as high speed SerDes (serializer/deserializer), network ports, and packet buffers on a single chip with higher clock frequency, without increasing the die size or causing a large increase in power density. High-radix routers, with their ability to utilize the growing off-chip bandwidth, have become the most efficient and economical way to build high-performance interconnection networks for supercomputers and datacenters. For example, the Cray YARC with 64 ports has been deployed in the BlackWidow system [4] to build one of the first high-radix networks; and the recent 48-port Aries router developed by Cray has been widely deployed in current Cray XC series network [6]. Aries adopts the same architecture as YARC, but with wider data paths, increased VC number from 2 to 8, and Dynamically Allocated Memory Queue, just like the enhancements made by MBTR.

Intel introduced a new HPC-optimized fabric technology known as Omni-Path Architecture (OPA), that aims to address performance and scaling weaknesses of HPC interconnect network. By inheritance of certain technologies from Cray Aries and QLogic InfiniBand, the Intel OPA' s 48-port switch demonstrates 21 percent lower switch fabric latency and 60 percent lower power consumption compared with InfiniBand EDR [9]. The OPA switch integrates 48 physical ports each providing 100 Gbps bandwidth and 8 VCs, four physical ports comprise one Mport, and up to 12 Mports interconnected by an internal crossbar.

IBM implemented a 136-port SCOC (scalable clos on-chip) switch using the clos topology to integrate 136 bidirectional 25 Gb/s ports, internally running at 9.9 Tb/s with a speedup up to 1.45x [10]. SCOC partitions the 136 input/

output ports into 34 groups each containing 4 ports thus multiplexing four 34×34 middle-stage crossbars to interconnect these groups. Scheduling in SCOC is realized by per-group and per-port arbiters that perform different scheduling algorithms asynchronously. This mechanism reduces the arbitration complexity from 136×136 to 34×34. SCOC reduces the throughput loss incurred by non-accepted grants at the output arbiters via a multi-iteration scheduling handshake and a fast internal connection. A multi-stage hierarchical schedule mechanism was developed to provide high frequency and fairness for the 34-to-1 arbitration, which outperforms the traditional flat crossbars and hierarchical switches.

It is also worth mentioning that SCOC is designed specifically for Ethernet networks where packet discard is allowed, which helps to relieve heavy memory burden. Our proposed MBTR is designed for HPC lossless networks with very high demand on delay and bandwidth, which makes it more challenging to implement. Each port of MBTR consists of four 25 Gb/s SerDes, providing 100 Gb/s line rate. While the datapath width of the route pipeline is 24 B with 700 MHz operating frequency, which achieves 134 Gb/s bidirectional bandwidth per port. Hence, MBTR internally runs at 9.6 Tb/s, offering a speedup of 1.34x.

We also use a simple and highly-efficient arbitration to achieve higher frequency than SCOC. Besides, adaptive routing is implemented by providing escape ways governed by bubble flow control[20]. Half of VCs used as adaptive channels perform fully adaptive routing, and the other half VCs used as escape channels perform deterministic routing to resolve the deadlock incurred by a cyclic dependency between adaptive channels.

We now discuss MBTR power consumption, where 57 percent comes from the SerDes cores. A current, state-of-the-art SerDes core has an energy efficiency of about 20 pJ/bit. Using this as a reference, an exascale computing system with 400 Gbps/port would reach a power consumption of 8 W per port. To design an interconnection network for one hundred thousand nodes, the nD-Torus network (assuming n=5) requires 1,000,000 router ports and a 4-level Fat Tree network needs 800,000 router ports. The corresponding power consumption would be 8 and 6.4 MW, just for the SerDes cores. Therefore, such a SerDes-based electrical interconnection is obviously infeasible for exascale computing. On the other hand, although the number of chip pins increases linearly, high-speed I/O counts barely increase because more Vdd and Gnd pins are needed to deliver power and performance. Therefore, it would be hard to integrate more high-speed ports in a single chip because of pin constraints. Optical technologies such as silicon photonics and 3-D integration might be a promising direction to build future energy-efficient interconnection networks.

8 Conclusion

We propose a scalable and flexible router microarchitecture, Multiport Binding Tile-based Router. MBTR provides an efficient and resilient alternative for building affordable high-radix routers. For radix-64 routers, MBTR is able to reduce memory and global wires up to $50 \sim 75\%$, and achieve indistinguishable even better performance, compared with a state-of-the-art hierarchical crossbar design. Moreover, we propose a non-saturated throughput theory, based on this theory, we reconsider the design targets of parallel switch architectures and theoretically deduce the sufficient and necessary conditions that should be satisfied by MBTR parameters to achieve 100 percent throughput. We further explore design criteria about how to determine the MBTR parameters under the given constraints of performance, power and area. In a word, the MBTR can be adapted easily to different power and area design objectives by adjusting its design parameters. MBTR keeps all the advantages of YARC, such as high throughput, reduced HOL blocking, and a fully symmetrical structure. Moreover, we presented a simple but efficient two-level arbitration scheme that has performance comparable with the multi-iteration iSLIP algorithm, but with much lower hardware cost.

So far, MBTR chips have been successfully fabricated and being deployed in the Tianhe-2 enhanced interconnection network. Tianhe-2 is located at the National Supercomputer Center in Guangzhou, China, and was consecutively ranked No. 1 from 41st to 46th edition of the TOP500 list.

Acknowledgments

This research is supported by 863 Program of China (2016YFB0200401, 2016YFB0200200), NSFC (61502507, 61672526), the laboratory pre-research fund (9140C810106150C81001) and by FANEDD under Grant No. 201450.

References

[1] Top 500 organization. [Online]. Available: http://www.top500.org
[2] R. Lucas, J. Ang, K. Bergman, S. Borkar, W. Carlson, et al., "top ten exascale research challenges," *DOE advanced scientific computing advisory subcommittee (ASCAC) report*, (2014). [Online]. Available: http://www.osti.gov/scitech/biblio/1222713
[3] J. Kim, W. J. Dally, B. Towles, and A. K. Gupta, "Microarchitecture of a high-radix router," *ACM SIGARCH Comput. Archit. News*, vol. 33, no. 2. 2005, pp. 420–431.
[4] S. Scott, D. Abts, J. Kim, and W. J. Dally, "The BlackWidow high-radix Clos network," in *Proc. Int. Symp. Comput. Archit.*, Jun. 2006, pp. 16–28.
[5] R. Alverson, D. Roweth, and L. Kaplan, "The Gemini system interconnect," in *Proc. 18th IEEE Symp. High Perform. Interconnects*, Aug. 2010, pp. 83–87.
[6] G. Faanes, A. Bataineh, D. Roweth, T. Court, E. Froese, B. Alverson, T. Johnson, J. Kopnick, M. Higgins, and J. Reinhard, "Cray cascade: A scalable HPC system based on a Dragonfly network," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2012, Art. no. 103.
[7] K. F. Wang, M. Fang, and S. Q. Chen, "Design of a tile-based high-radix switch with high throughput," in *Proc. Int. Conf. Netw. Inf. Technol.*, 2011, pp. 277–285.
[8] J. H. Ahn, S. Choo, and J. Kim, "Network within a network approach to create a scalable high-radix router microarchitecture," in *Proc. High Perform. Comput. Archit.*, 2012, pp. 1–12.
[9] S. Chari and M. R. Pamidi, "The Intel Omni-path architecture (OPA) for machine learning," white paper, Dec. 2017.
[10] N. Chrysos, T. Minkenberg, M. Rudquist, C. Basso, and B. Vanderpool, "SCOC: High-radix switches made of bufferless clos networks," in *Proc. High Perform. Comput. Archit.*, 2015, pp. 1–13.
[11] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Trans. Netw.*, vol. 7, no. 2, pp. 188–201, Apr. 1999.
[12] S. Brick, "BlackWidow hardware system overview," in *Proc. Cray User Group*, Lugano, Switzerland, pp. 1–6, 2006. [Online]. Available: https://cug.org/5-publications/proceedings_attendee_lists/2006CD/S06_Proceedings/pages/Authors/Stephenson/Stephenson_Paper.pdf

[13] M. Karol, M. Hluchyj, and S. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Trans. Commun.*, vol. COM-35, no. 12, pp. 1347–1356, Dec. 1987.
[14] J. Sztrik, "Basic queueing theory," GlobeEdit, OmniScriptum GmbH & Co. KG, Saarbrucken, Germany, (2016). [Online]. Available: http://irh.inf.unideb.hu/user/jsztrik/publications/books/GlobeEdit_Basic_Queueing_Theory_Sztrik_2016.pdf
[15] S. Chuang, A. Goel, N. McKeown, and B. Prabhaka, "Matching output queueing with a combined input/output-queued switch," *IEEE J. Select. Areas Commun.*, vol. 17, no. 6, pp. 1030–1039, Jun. 1999.
[16] E. Oki, Z. Jing, R. Rojas-Cessa, and H. J. Chao, "Concurrent round-robin-based dispatching schemes for Clos-network switches," *IEEE/ACM Trans. Netw.*, vol. 10, no. 6, pp. 830–844, Dec. 2002.
[17] Y. Xu, B. Zhao, Y. Zhang, and J. Yang, "Simple virtual channel allocation for high throughput and high frequency on-chip routers," in *Proc. High Perform. Comput. Archit.*, 2010, pp. 1–12.
[18] H. Y. Zhang, K. F. Wang, J. M. zhang, N. Wu, and Y. Dai, "A fast and fair shared buffer for high-radix router," *J. Circuits Syst. Comput.*, vol. 23, no. 1, pp. 1–30, 2014, doi: 10.1142/S0218126614500121.
[19] L. Chen and T. M. Pinkston, "Worm-bubble flow control," in *Proc. High Perform. Comput. Archit.*, 2013, pp. 1–12.
[20] V. Puente, C. Izu, R. Beivide, J. A. Gregorio, F. Vallejo, and J. M. Prellezo, "The adaptive bubble router," *J. Parallel Distrib. Comput.*, vol. 61, pp. 1180–1208, 2001.
[21] Infiniband trade association. [Online]. Available: http://www.infinibandta.com
[22] M. Lipson, "Silicon photonics: The optical spice rack," in *Proc. Int. Conf. Photon. Switching*, 2009, p. 1.

Yi Dai received the MS and PhD degrees in computer science from the National University of Defense Technology (NUDT), China. She is an associate professor with the Department of Computer Science, NUDT. Her research interests focuses on high performance interconnection networks with a strong emphasis on high-radix router architecture, communication protocols as well as deadlock-free routing, and flow control design. She is a member of the IEEE.

Kai Lu received the PhD degree in computer science from the National University of Defense Technology (NUDT), China, in 1999. Currently, he is a professor with the University. His research interests include parallel processing and system software. He is the associate chief designer of Tianhe-2 supercomputer.

Liquan Xiao received the MS and PhD degrees in computer science from the National University of Defense Technology (NUDT), China. Currently, he is a professor with the University. His research interests include architecture of high performance computing, high speed interconnect network, system integration, and power management. He is the associate chief designer of Tianhe-2 supercomputer.

Jinshu Su received the PhD degree from the National University of Defense Technology (NUDT), China, in 2000. Currently, he is a professor with the University. His research interests include parallel processing and high speed network. He is the associate chief designer of Tianhe-2 supercomputer.