2011 2nd International Conference on Networking and Information Technology
IPCSIT vol.17 (2011) © (2011) IACSIT Press, Singapore

Design of a Tile-based High-Radix Switch with High Throughput

Wang Kefei[1], Fang Ming[2] and Chen Songqiao[2]

[1]School of Computer, National University of Defence Technology, Changsha, China
[2]School of Information Science and Engineering, Central South University, Changsha, China
[1] wkf.working@gmail.com ,[2] Fangming121@yahoo.com.cn, Chensq@mail.csu.edu.cn

Abstract—Research and trends reveal that high-radix switch is a necessity in future HPC design. HOL blocking limits the throughput of an N×N crossbar to less than 58% and this can be even worse in high radix switches because most technologies proposed to eliminate HOL blocking cannot be scalable to high radix switches. In this paper, a new efficient architecture for high-radix switches is proposed. The architecture, referred to as Hierarchical Asymmetric Crossbar (HAC), relies on the fact that in an asymmetric crossbar, where input ports are much less than output ports, effect of HOL blocking on crossbar throughput is very small or negligible. Thus an HAC architecture of N×N high radix switch can be formed by N/m smaller m×N asymmetric crossbars to achieve high throughput without any special endeavours to eliminate HOL blocking. A prototype of 32×32 HAC high radix switch prototype is introduced in the paper. Clock cycle simulation reveals that in such a switch the impact of HOL blocking is almost **thoroughly** nearly all eliminated and the switch throughput can be as high as 94%. The HAC based switch can be implemented in TILE-based array structure to simplify the switch design.

Keywords-Switch, Hierarchical Crossbar, Throughput, HPC, TILE

1. Introduction

Interconnection network is the most important infrastructure in high performance computing systems and it must provide low latency and high bandwidth at a moderate cost and power consumption. The number of end nodes in high performance computer (HPC) increases rapidly in recent years and till June 2011, all the top 10 most powerful HPCs have thousands or tens of thousands processors [1]. In the next five years, HPC with peak performance as high as 10P ~50PFlops will be produced. There will be hundreds of thousands processors integrated in these systems. Design of High Performance Network (HPN) for these systems faces great challenge on performance, scalability, reliability, cost and power consumption.

The efficiency of interconnection network largely depends on the switch design. Kim and Dally prove that in large scale HPNs high-radix switches with narrow channels are able to achieve lower packet latencies than low-radix switches with wide channels [2]. The basic reason is that with high-radix switches the hop count in the network decreases. The other benefits from high-radix switches are lower cost and lower power consumption as the total number of switches and links to build a network is reduced. Two key points make high- radix switches feasible. First, during the last 20 years, the pin bandwidth of a switch has increased up to 25~40Gbps [3]. It is easy to build a high performance network port using less pins, for example, design of a 40Gbps port using 10Gbps pin just like Infiniband QDR will consume only 20 Pins. Consequently, without pin limitation a switch can host more ports in it. Second, evolution of chip process is another important pushing force. As chip process evolves from 90nm to 40nm or 28nm, both the cell size and power consumption go down, on the other hand the working frequency goes up. It is easier to integrate more high speed SERDESs, more network ports, and more packet buffers in a switch, achieve higher working frequency (over 1GHZ) and keep die size of chip and power consumption at a feasible level, in another words, build a high radix switch. Myrinet have built a 32-port switch chip [4]. Infiniband QDR switch chip integrates 36 ports [3]. Cray YARC is a 64 ports switch chip [17]. IBM uses a 47-port HUB in its BlueWater system.

Maybe in the next few years, we can see high radix switch with hundreds of ports and over 10Tbps aggregated bandwidth.

However, designing high-radix switches presents major challenge on achieving high throughput. Switch efficiency, especially Input Queued (IQ [6-7]) switch, will be largely impacted by the Head-Of-Line (HOL) blocking experienced within the switch. An IQ switch uses only memories/buffers at the input ports, and does not use speedup. In such switches, an N×N switch only requires N memories, this character makes IQ structure based switch very popular in switch design, especially when the switch chip cannot afford enough buffer resources. In IQ switches, the HOL blocking problem appears when a packet at the head of a queue is blocked (because it is requesting an occupied resource), thus preventing packets in the same queue to make forward progress even if their requested resources are free. Normally, HOL blocking limits the throughput of an input queued N×N crossbar to less than 58.6% [8].

One solution to eliminate the HOL blocking in IQ switches is the use of N queues at every input port and mapping the incoming packet on a queue associated to the requested output port. This technique is known as Virtual Output Queuing (VOQ) [9-10]. In order to achieve high throughput, VOQ switches need a central arbiter to resolve the maximum matching problem among N2 virtual input ports and N output ports. Many arbitrate algorithms have been proposed, such as iSLIP, PIM, LRU, etc. Some of these algorithms can achieve nearly 100% throughput for Low-Radix VOQ switches, in another point of view, totally eliminate the negative impact of HOL blocking [11-12].

However, VOQ structure cannot scale smoothly to High-Radix switches. One reason is that the queue requirements of VOQ increases quadratically with the number of switch ports (N2). Therefore, as the number of ports increases, this solution becomes too expensive. The other important reason is that all relatively matured arbitrating algorithms used in VOQ switches require multiple iterations to realize maximum matching. For example, iSLIP, which have been proved that can get 100% throughput, needs O(logN) iterations. Multi-iteration can be implemented in one cycle and be affordable when number of switch ports is low, for example, less than 8. But when switch ports increase, it is too complex to adopt these multi-iteration algorithms, as a trade-off, iterations must be cut off to an affordable value. Consequently the VOQ switch still suffers from HOL blocking.

Contrast to IQ switch, there are many alternative structures of switches, such as Output Queued (OQ) switch, Combined Input Output Queued (CIOQ) switch, Buffered Crossbar (BC) switch. All these types of switch have been evaluated in detail in [13]. For Low-Radix switch, some of them have no HOL blocking problem, some of them can effectively lower down the impact of HOL blocking. But all of them have been sentenced to have scalability problems to be adopted in High-Radix switch [13].

Hierarchical Crossbar (HC) was proposed in [5] and claimed suitable for High-Radix switch. Basically, this solution divides the big $N \times N$ crossbar into (N/p)2 smaller p×p sub-switches, each sub-switch has p input buffers and p output buffers. So a HC switch has 2p(N/p)2 memories totally. HC structure can be relatively easy to scale to High-Radix. The sub-switch radix, p, influence memory requirements and performance of HC switch in contrastive manner. As p increases, the switch uses less memory but has relatively lower performance. On the other hand, as p decreases, the switch uses more memory but has relatively higher performance. By properly selecting parameter p, the HC switch may achieve a good trade-off between cost and efficiency. However, HC still suffer from HOL blocking problem in its $p \times p$ sub-switches, and consequently limits the throughput of the whole switch. For example, performance evaluation shows throughput of YARC is nearly 87% [14].

HC switch has another important character, and maybe the most valuable character, that if p=$\sqrt{n}$, the switch will has N smaller $\sqrt{n} \times \sqrt{n}$ sub-switches, and the whole switch can be easily partitioned to N homogeneous blocks (named TILE in Cray YARC switch, YARC has 64 ports and constitutes 64 TILEs). This TILE-based switch architecture has been convinced to be very convenient for chip design, floorplan and wire routing in the chip die.

In this paper, we propose a new switch architecture for High-Radix switch referred to as Hierarchical Asymmetric Crossbar (HAC). The basic and most important principle behinds HAC is that in an Asymmetric m×N crossbar, when m is much less than N, impact of HOL blocking can be much lower comparing to a symmetric N×N crossbar, hence a HAC built by N/m smaller asymmetric m×N sub-switches can easily achieve high throughput without any special optimization.

The rest of the paper is organized as follows. In Section 2, the HAC switch architecture is described. In Section 3, evaluation of the proposed architecture will be presented. In section 4, an optimized TILE-based implementation of a 32-port switch using HAC structure is introduced. Finally, in Section 5 some conclusions are provided.

2. Hac Switch Architecture

In 1987, MJ Karol etc revealed and defined the HOL blocking problem in IQ based switch in [8], and proved that HOL blocking limits throughput of $N\times N$ crossbar to less than 58%. Since then, a large number of switch architecture have been proposed aiming to eliminate HOL blocking. Nearly all the research are based on symmetric $N\times N$ crossbar because a switch with symmetric N input ports and N output ports is very nature as shown in Fig.1(A). In this paper, at the first time, we study asymmetric crossbars as Fig.1(B) shows. An asymmetric crossbar (ASC) has m input ports and N output ports. Symmetric crossbar can be viewed as a special case of ASC with $m=N$. We focus our attention on those ASCs with m less than N in this paper.

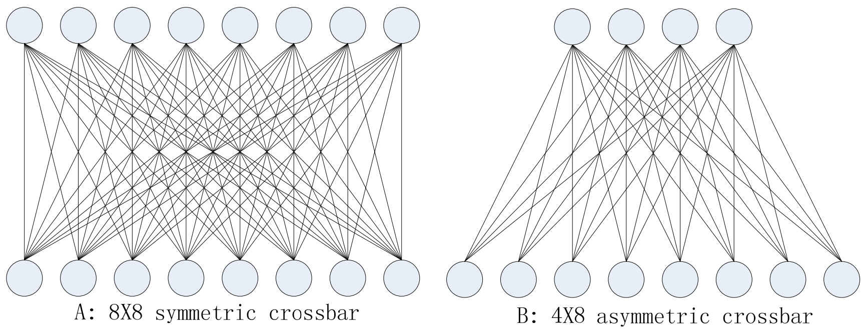

Figure 1 . Symmetric and Asymmetric Crossbars

First, we design a simple IQ based *32×32* symmetric crossbar as the basic test bench. By enabling or disabling some input or output ports and configuring each input port's route destination distribution model, we test some typical ASC's throughput without doing any optimization. The test is based on random traffic model and the traffic payload is 100%. Packets injected into the switch have variable number of flits, randomly from 1 to 6. In order to kick off any unnecessary disturbing factors related to buffer depth, we assume that buffers in each input port are infinite. The result is listed in Table 1.

As shown in Table 1, symmetric crossbars have relatively poor throughput just like many previous works having revealed. As N/m ratio increases, throughput of ASCs improves rapidly, which confirms our theory analyses. For those ASC with *N/m >= 4*, (*2×8*, *2×16*, *2×32*, *4×16*, *4×32*), we think the throughput is reasonably high enough for us to ignore the impact of HOL blocking.

TABLE I. THROUGHPUT IN TYPICAL ASCS

Input Ports /output ports	2	4	8	16	32
2	72.65	86.06	93.32	97.11	98.75
4	X	63.77	80.02	89.8	95.2
8	X	X	60.07	77.22	88.18
16	X	X	X	58.57	75.84
32	X	X	X	X	57.76

The test suggests us to build a hierarchical structure for High-Radix switches, dividing the big $N\times N$ crossbar into N/m smaller $m\times N$ ASCs. Using carefully selected N/m ratio, we can achieve high throughput in High-Radix switch without any other special endeavours in elimination of HOL blocking. Fig.2 depicts a *32×32* High-Radix switch implemented using *4×32* asymmetric crossbars.

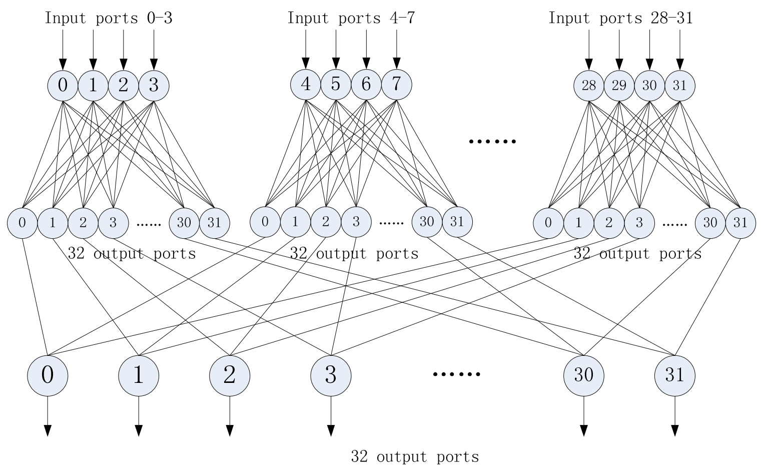

Figure 2 . 32×32 Crossbar implementation using HAC

We name this switch structure as Hierarchical Asymmetric Crossbar (HAC) structure. Commonly, the big $N×N$ crossbar is divided into two levels, first level includes N/m $m×n$ sub-switches, each sub-switch has m input buffers, one for each related input ports, and has N output ports, one for each output ports of the switch. The second level includes N small m×1 multiplexer.

Totally, the switch should include N input port buffers and N^2/m intermediate buffers. A switch must have a big input buffer in each input port to accommodate data/packets coming from the physical link, the input buffer depth is determined by the Round Trip Time (RTT) and numbers of Virtual Channels (VC) in the physical link. We can directly use these buffers as HAC's sub-switch input buffers, so there are only N^2/m additional intermediate buffers required by the HAC structure. Since High-Radix switch always use the most advanced chip process, we think this memory requirement cannot be a design limitation in High-Radix switch.

In Fig.2, N=32 and m=4. The switch will have 8 homogeneous 4×32 sub-switches, using 32 input buffers and 256 intermediate buffers.

3. Evaluation of Hac Switch

In this section we evaluate the performance of the HAC switch architecture. For the purpose of sufficient evaluation, a 32×32 HAC switch is implemented in RTL code and simulated at the clock cycle level. Each port of the switch is attached to an end node. Each end node injects traffic at the maximum link rate, one flit per cycle if the input buffer has idle space to accommodate it. We use the credit-based flow control [15][16] between the end node and the input port. Packets injected into the switch are randomly distributed to 32 output ports, and packets have variable number of flits, randomly from 1 to 6.

Previous works have revealed that the buffer depth of intermediate port in a multi-level switch structure can influence the throughput of a crossbar significantly. So we first test the crossbar with different intermediate buffer depth to identify the appropriate buffer depth to achieve high throughput. Three buffer depth, (32, 64, 128), have been tested, and the results are shown in Fig.3.

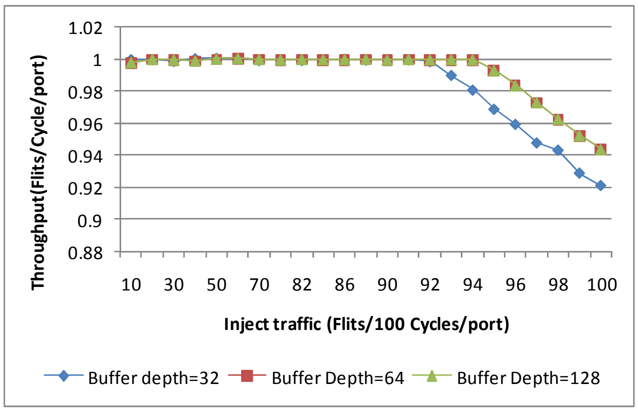

Figure 3 . Switch throughput evaluation with different buffer depth

For the test of preferred depth 64, the throughput keeps nearly 100% till the packet injecting rate increases to 94%. Even injecting in full rate, HAC can achieve 94% throughput. So we can claim that the HAC structure is really very efficient. Another result hiding in the test is that the throughput of 32×32 switch is very close to the throughput of its 4×32 sub-switch. We can reasonably deduce that the second level m×1 multiplexer will cause little throughput degression to the whole switch when the intermediate buffer is deep enough.

Another important performance of switch is packet delay. We do this test using the same configuration with throughput test, but keep setting the buffer depth to 64. First, we do the test with fixed packet length, setting packet length to 1, 2, 3, 4, 5 and 6 flits respectively. Second, we do the test with variable packet length, setting packet length randomly but constraining the maximum packet length to 1, 2, 3, 4, 5 and 6 flits respectively. In each test, we record and analyze the maximum flit delay and the average one. These test results are shown in Figure 4 to Figure 7.

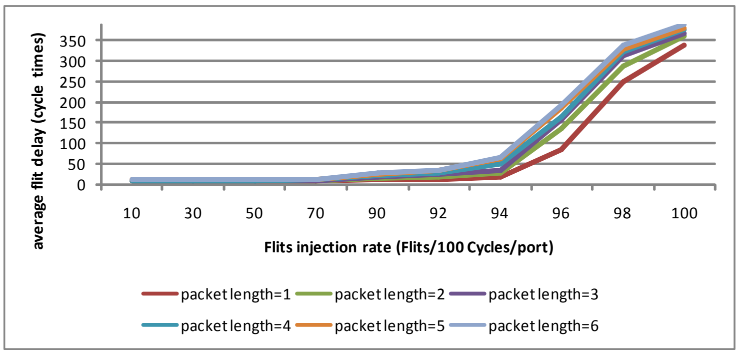

Figure 4 . Average flit delay with fixed packet length

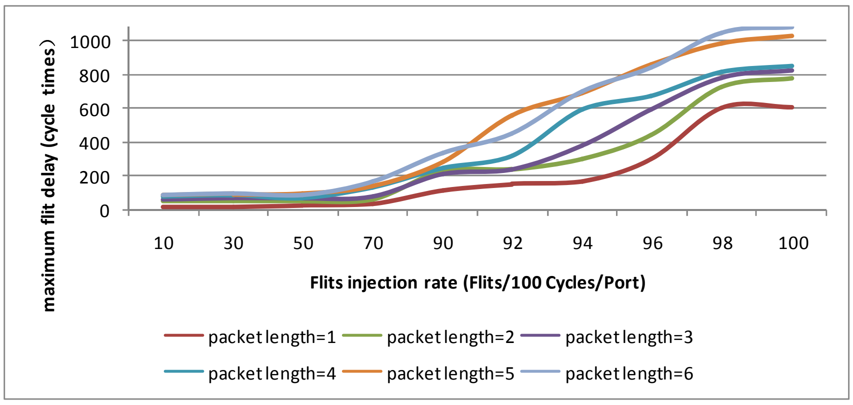

Figure 5 . Maximum flit delay with fixed packet length

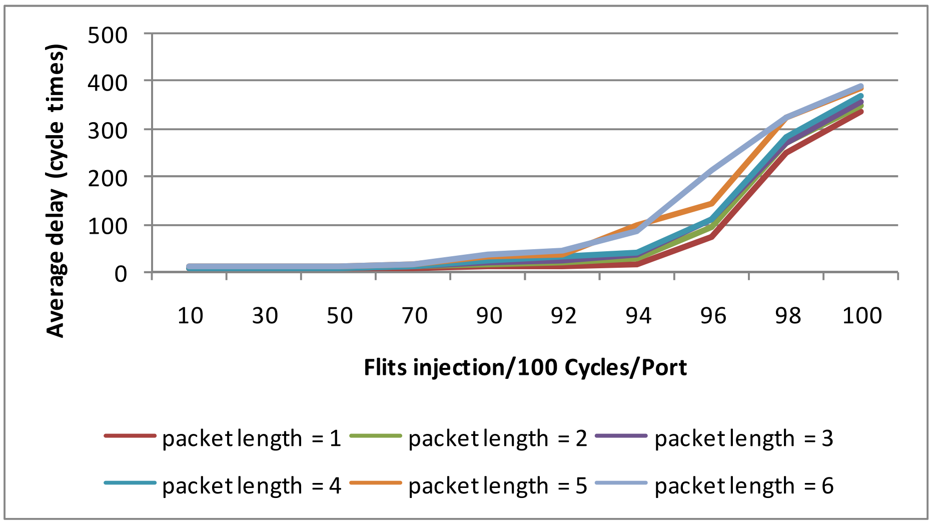

Figure 6 . Average flit delay with variable packet length

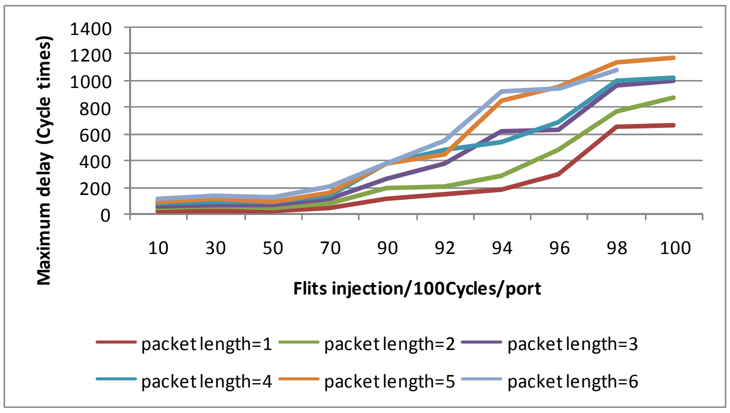

Figure 7 . Maximum flit delay with variable packet length

These test results show that the HAC structure also performs well in packet delay and the saturated traffic payload is as high as 94%. When the switch is not saturated, average packet delay is very low. Once the traffic payload is greater than the saturated traffic, average packet delay grows very quickly.

4. Hac Structure Optimization

It is easy to design a low radix N×N crossbar when N is less than 8. As N goes up, for example, N>=16, some new difficulties arise. First, for frontend logic design, it is hard to design a high performance arbiter to schedule input packets to output ports in high working frequency. Second, for backend floorplan and wire routing in die, big crossbar means very high density wire in a small area and long distance from one port to some other ports, these may make wire routing very difficult and hard to realize high working frequency.

Partition and structural optimization should be done in the design of high radix switch to simplify logic design and back-end processing. YARC provides a paradigm in structural high radix switch design [17]. It partitions its 64 ports and switch fabric into 64 homogenous blocks (named TILE) and organizes these TILEs in an 8×8 array. Each tile contains one port related logics, an 8×8 switch and associated buffers. Each tile only exchanges data between those tiles in the same row and column with it. TILE based structure lowers down the crossbar radix from 64×64 to 8×8, hence simplifies and optimizes the crossbar implementation. Homogenous Tile and Array organization regularize the switch fabric design and limit wire routing direction and distance, and also make logic design and back-end processing, such as floorplan and wire routing, easy to implementation in silicon.

HAC structure can also be regularized and optimized to TILE-based structure. From Fig.2, 32-ports HAC based switch is formed by 8 4×32 sub-switches. 4×32 sub-switch is relatively simple than 32×32 but still too complex for back-end processing. It is necessary to partition a 4×32 sub-switch to some smaller blocks, just like Fig. 8 and Fig. 9 show.

Fig. 8 shows a typical 4×32 sub-switch implementation. The sub-switch is partitioned to an arbiter and a big crossbar. Input ports transfer their requests to the arbiter. The arbiter gives out its grant signals to individual output ports to control the 4:1 MUXs to select which input ports can transfer data to their destination output port. The 4×32 arbiter may be implemented as a central arbiter or as some distributed arbiters. That is not the matter we care in this paper. So we don't discuss its implementation in detail.

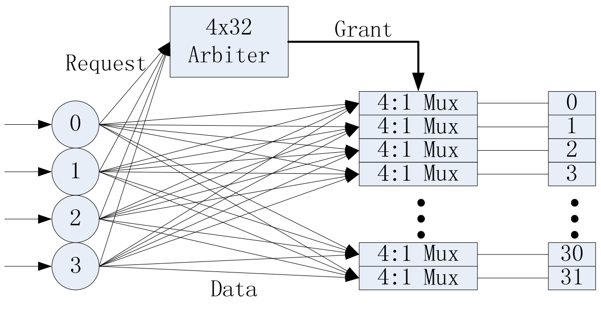

Figure 8 . Partition 4×32 sub—switch

In Fig. 9, we partition this typical 4×32 sub-switch to two levels and five small blocks, and we will show the path between the arbiter and the MUXs. The first level is the 4×32 Arbiter Block (AB) acceptting requests from the four input ports buffer (IPB) and issuing its grant signals to control the four IPBs to output their data (IPD, Input port data). The second level includes four homogenous 4×8 crossbars (XB). IPDn (n=0,1,2,3) from IPBn will be broadcasted to four crossbar's XIPmn (m=0,1,2,3). For example, IPD0 is broadcasted to XIP00, XIP10, XIP20 and XIP30. Along with data from IPD to XIP, there will be the grant signals output from the arbiter block used to control the XBs to route data to its output ports. There are no arbiters in the four crossbar blocks and the crossbars degrade to pure routing path. In order to tolerate long wire from the arbiter block to the crossbar, we can insert any cycles of repeater on the path to realize high working frequency. These repeaters must repeat data and grant signals synchronously.

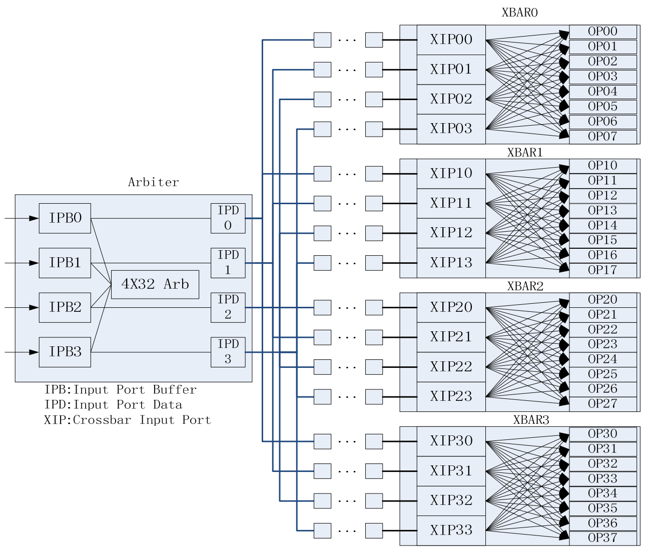

Figure 9 . Partitioned 4×32 sub-switch

The partitioned sub-switch in Fig.9 is logically equivalent to the typical sub-switch in Fig.8, while it is divided to five smaller blocks and these blocks can be physically distributed. After partitioning the 4×32 sub-switch, we can re-arrange the entire 32×32 switch depicted in Fig.2 to a TILE-based regular structure as shown in Fig. 10 to Fig. 12.

The switch is formed by two types of TILEs named PTILE (Port Tile) and ATILE (Arbiter TILE) respectively. There are 32 PTILEs in the switch, each PTILE associated to one of the 32 ports in the switch. There are 8 ATILEs in the switch. All these TILEs are arranged in an 8×5 array. Each row has four PTILEs and one ATILE. Detailed structure of PTILE and ATILE are depicted in Fig.11 and Fig.12 respectively.

The PTILE has seven interfaces listed below:

(1) Input data from the link of switch port.
(2) Output data to the link of switch port.
(3) Input data relay to the ATILE in the same row.
(4) Four data channels from the ATILE in the same row.
(5) Grant signals from the ATILE in the same row.
(6) Seven sub-switch output ports to PTILEs in the same column.
(7) Seven sub-switch input ports from PTILEs in the same column.

The PTILE includes three parts:

(1) One port logic realizing a port.
(2) A 4×8 sub-switch implementing the XB block in Fig.9.
(3) A 8×1 Mux implementing the second level 8×1 MUX of the 32×32 switch in Fig.2.

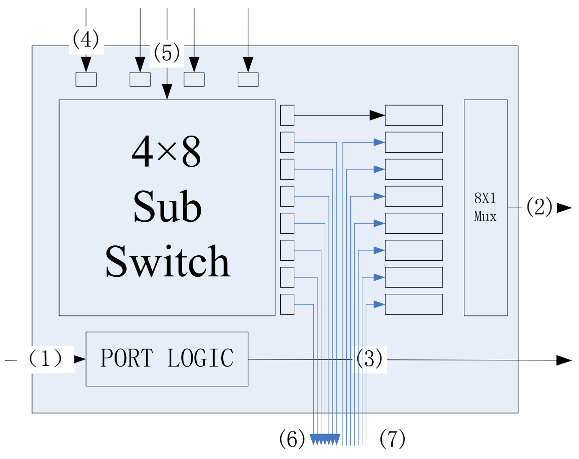

Figure 11. Detailed structure of PTILE.

The ATILE just implements the arbiter block in the Fig.9, receives data from four input ports (four PTILEs) in the same row, stores data in four data buffer, arbitrates among four input ports and grants them to transfer data to 32 output ports.

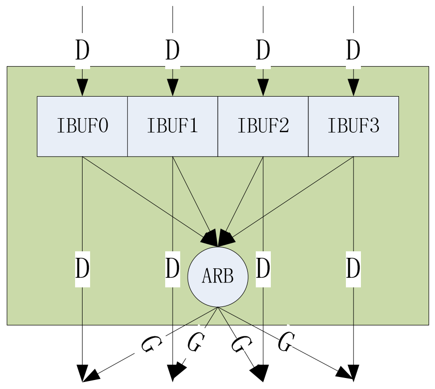

Figure 12. Detailed structure of ATILE.

So in one row, the four PTILEs and one ATILE form a 4×32 sub-switch depicted in Fig.9. There are 8 rows and 8 4×32 sub-switches constructing the 32×32 switch. 32 ports of the switch are distributed in 32 PTILEs. In a column, there are 8 PTILEs and the sub-switch in these PTILEs has the same output ports group, just including those ports distributed in these 8 PTILEs. PTILEs will only exchange data among PTILEs in the same row or in the same column.

All 32 PTILEs and 8 ATILEs are homogenous. Hence we need only realize one PTILE and one ATILE, then replicate them for 32 and 8 times respectively, and connect them regularly. Evidently, such a regular TILE-based structure can simplify the switch implementation greatly just like YARC has done.

5. Conclusions

High-radix switches are becoming a necessity in future HPC design. Unfortunately, although current switch structures is efficient and suitable for Low-Radix, they cannot scale smoothly to High-Radix due to low switch efficiency or high cost, thus preventing them to be applied in building large switches. In this paper, we proposed a new architecture for high-radix switch. The proposed HAC structure relies on a very simple principle that an asymmetric crossbar outperforms symmetric crossbar in throughput and packet relay latency. Thus an N×N big crossbar can be divided into some asymmetric m×N sub-switches in the first switch level and united by N/m m:1 multiplexers. By selecting proper N/m ratio, throughput of N×N switch can be close to one single m×N crossbar. One sample 32×32 HAC switch has been implemented using 4×32 smaller sub-switches. Clock cycle level simulation and evaluation show that this switch can achieve throughput as high as 94%. A regular TILE-based implementation structure is also discussed in the paper. The 32×32 switch is organized as an 8×5 array. Each array includes four PTILEs and one ATILE. Switch design can be simplified to design and implement only a PTILE and an ATILE, replicate them for many copies and connect these copies regularly.

6. References

[1] Top 500 organization. Available at http://www.top500.org.

[2] J. Kim, W. J. Dally, B. Towles, and A. K. Gupta. Microarchitecture of a high-radix router. 32nd Annual International Symposium on Computer Architecture (ISCA '05), pages 420–431, 2005.

[3] Infiniband trade association. http://www.infinibandta.com.

[4] Myrinet, Available at http://www.myrinet.com

[5] Quadrics qsnet. Available at http://doc.quadrics.com.

[6] M.Karol andM.Hluchyj. Queuing in high performance packet-switching. IEEE J. Select. Areas. Commun, 6:1587–1597, Dec. 1998.

[7] N. McKeown, M. Izzard, A. Mekkittikul, W. Ellersick, and M. Horowitz. The tiny tera: A packet switch core. IEEE Micro, 17:27–33, Jan./Feb. 1997.

[8] M. J. Karol and et al. Input versus output queueing on a space-division packet switch. IEEE Transactions on Communications, COM-35(12):1347–1356, 1987.

[9] T. Anderson, S. Owicki, J. Saxe, and C. Thacker. High-speed switch scheduling for local-area networks. ACM Trans. on Computer Systems, 11(4):319–352, Nov. 1993.

[10] Y. Tamir and G. L. Frazier. High-performance multi-queue bu ▪ ers for vlsi communications switches. SIGARCH Comput. Archit. News, 16(2):343–354, 1988.

[11] N. McKeown and T. Anderson, "A Quantitative Comparison of Iterative Scheduling Algorithms for Input-Queued Switches," Computer Networks and ISDN Systems, Vol. 30, No. 24, pp. 2309-2326,December 1998.

[12] N. McKeown, "The iSLIP Scheduling Algorithm for Input-Queued Switches," IEEE/ACM Transactions on Networking, Vol. 7, No. 2, pp.188-201, April 1999.

[13] G. Mora, J. Flich, J. Duato, E. Baydal, P. López, O. Lysne. "Towards an Efficient Switch Architecture for High-Radix Switches". ACM/IEEE Symposium on rchitectures for Networking and Communications Systems. San Jose, CA, Dec 3-5, 2006.

[14] Cray. http://www.cray.com

[15] W. J. Dally. Virtual-channel flow control. Proceedings of the 17th annual International Symposium on Computer Architecture, pages 60–68, 1990.

[16] J. Duato, I. Johnson, J. Flich, F. Naven, P. Garcia, and T. Nachiondo. A new scalable and cost-effective congestion management strategy for lossless multistage interconnection networks. Proceedings of the 11th International Symposium on High-Performance Computer Architecture, pages 108–119, Feb. 2005.

[17] S. Scott, D. Abts, J. Kim, and W. J. Dally. The BlackWidow High-radix Clos Network. In Proc. of the International Symposium on Computer Architecture (ISCA), Boston, MA, June 2006.