Journal of Computer Science 1 (3): 296-303, 2005
ISSN 1549-3636

Design and Simulation of an Input Queuing Packet Switch

[1]Azeddine Bilami, [2]Mustapha Lalam and [3]Mohamed Benmohammed
[1]Department of Computing Science, University of Batna, Algeria
[2]Department of Computer Science, University of Tizi ouzou, Algeria
[3]Department of Computer Science, University of Constantine, Algeria

Abstract: Many Architectures of Internet routers, ATM and Ethernet switches have been proposed and analysed in literature. Theoretically reliable and valid solutions have been developed to achieve high performances, but a lot of them are not feasible in practice for commercial and technological reasons. Few papers develop the implementation and simulation aspects. The objective of this study is the design of a packet switch with a minimum cost and hardware complexity. We propose an input-queuing architecture using a multistage interconnection network and a simple cell selection policy implemented by hardware. The switch is described and simulated using a VHDL language. Performances in terms of throughput and cell loss are evaluated.

Key words: Routing, Switch, Multistage Interconnection Network (MIN), Benes Network, Self Routing, VHDL

INTRODUCTION

A major challenge concerned to high-speed switching is related today to switch design that requires the best possible compromise between ease of implementation and goodness of performances.

Switch Functionality is Twofold:

* Managing packet buffering while selecting packets to transmit each time to avoid contentions and cell loss.
* Routing packets from their incoming ports to their destination ports.

To avoid contentions and cell loss, the incoming packets are stored in buffers. These buffers can be in inputs, in outputs, in inputs and in outputs or shared by inputs and outputs. So, a choice that a designer may have is where to place the buffers. Although, a lot of existing switches use the shared buffers technique, it has been shown through several publications that the method using input buffers is the only one which can constantly answer to the increasing needs of large switches and to the high rates of present and future communication lines. With such solution, the cost of the switch remains acceptable and the management of the queues in the buffers less complex. Nevertheless, it introduces the well known problem of the HOL (Head Of Line) blocking which is usually solved by means of scheduling algorithms. Appropriate scheduling algorithms are adopted depending on the application environment and some constraints such as line rate, number of ports to connect, cost, expected performances.

To route packets from input ports to output ports, Multistage Interconnection Networks (MINs) are very attractive. They can route in parallel the incoming packets. They have a relatively low cost and are better adapted for VLSI implementation. MINs have been used initially in multiprocessors architectures to connect the processors to memory banks. Recently and regarding to their characteristics, an other interest is granted to these networks: they are used in the Internet routers, in the ATM and Ethernet switches and are appropriate to be used to construct electro-optic switches.

Using Benes network which present the best cost among non blocking MINs and input buffers technique with a simple selection policy of maximum cells to transmit without conflicts in a cycle, we propose in this study a switch with a low cost introducing a minimum hardware complexity. It may be useful to give the meanings of the following terms that are used frequently in this study.

Permutation: is a one-to-one I/O mapping, where all inputs and outputs are active. It is called a partial permutation, if any I/O are not active.

Cell: we consider packets of a fixed size. We prefer using in the continuation of this article, the term ‘cell’ instead of ‘packet’.

Throughput: defined as the number of cells arriving to their destinations divided by the total number of departed cells from their sources in a unit time (a cycle).

Interconnection Network

Benes Network: A (NxN) Benes network (Fig. 1) is a network with N inputs and N outputs. Its dimension is r $=\log_2 N$. It is composed of $2(\log_2 N)-1$ stage of 2^{r-1}

switching elements (SE for short) and presents with Waksman network, the best cost among all the non blocking multistage networks: [1]

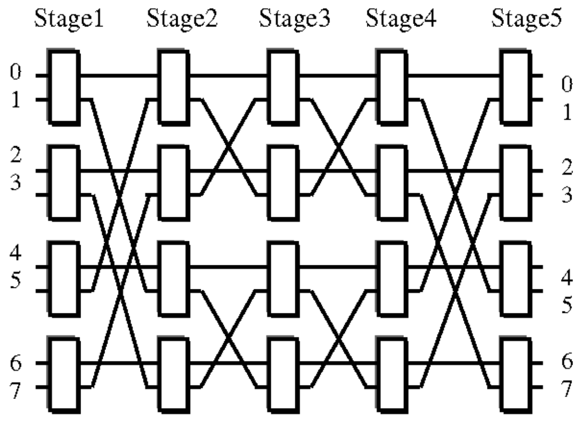

Fig. 1: A Benes Network for N=8

BENES network is dynamic and rearrangeable. It presents N/2 possible paths to establish a link between a free input and a free output. It is for this fact fault tolerant since we can always establish a path even if some switches are out of use. It offers a constant latency for all couples (input,output). The Only drawback that is known about it is the complexity of its routing control algorithms. The solution usually used is centralized: it needs a global controller, which configures the network before the transfers. This solution requires O(N.r) sequential time to position all the SE(s).

Self Routing: One of the features that give to this network a big interest, are the self-routing property: every 2x2 switch can decide to which of its outputs the incoming cell will be transmitted, depending only on the cell destination address. This means that the implementation of a complex routing algorithm, either centralized or distributed, is not needed.
It has been proved that some permutations families representing a subset of all possible permutations (N! for an NxN Benes network), can be routed automatically in only one pass, consuming O($\log_2 N$) in parallel time. Lenfant in [2], has defined the FUB (Frequently Used Bijections) family. Nassimi and Sahni have proposed in [3], an algorithm for the BPC (Bit Permute Complement) class. Boppana and Raghavendra [4] have resolved the problem for the LC (Linear Complement) family. These families are regrouped in the same class BPCL (Bit Permute Closure) of self routing permutations.
Four strategies for the BPCL self-routing permutations have been defined in [5]: TCR, BCR, LCR, HCR. In each of these strategies a particular bit of the destination address called the R-Bit (Routing Bit), determines the action to perform at the SE level. The switching command can be in this case, implemented locally at the 2x2 switch.

* TCR (Top Control Routing): R-Input is the superior input
* BCR (Bottom Control Routing): R-Input is the lower input
* LCR (Least Control Routing): R-Input is the smallest input
* HCR (Highest Control Routing): R-Input is the biggest input

The R-bit (routing bit) is defined:

$$\text{R-Bit}= \begin{cases} X_i \text{ for } 1 <= i <= r-1 \\ X_{2r-i} \text{ for } r <= i <= 2r- \end{cases}$$

where: $X_r, X_{r-1}, \ldots X_1$ is the destination address of the R-Input

Routing Algorithm: The adopted routing algorithm in this study is one of the most effective self routing algorithms for the Benes network [6].
In a first time, the load is fairly distributed on the 2x2 switches of the r-1 first network stages using LCR self routing strategy: The SE(s) are positioned according to the R-Bit of the input with the smallest destination address. We note here, that HCR strategy can be used instead LCR for the r-1 first stages. The performances will be the same: In [5], it has been shown that $|HCR| = |LCR|$.
For the r remaining stages, the switches are positioned using the classic routing algorithm applied in the omega network: if the R-bit is a 0, than the cell will be routed to the upper output of the SE, else it will be routed to the lower output.

Cell Buffering: The buffering operation consists in queuing the cells to transmit. The performances of the switch can be affected differently according to the way that is done. Different strategies are used depending on the physical site of the queues: in inputs, in outputs, in both inputs and outputs, or in shared buffers.

Output Buffers: Although output buffers give the optimal delay-throughput performance, switches that use them are difficult to achieve. In this method, every output can receive simultaneously during the same cycle, N cells from the N inputs. Thus, the switch must be able to put in the same queue and during one cycle, the N cells destined to the same output. The operation of setting in queue must therefore operate N times quickly than the rate of cell arrivals (speed up). If this solution is feasible in case of small capacity switches, it should be not possible for switches of big capacities (the N-times speed-up in the switch limits the scalability of this architecture) [7, 8].

Shared Buffers: The approach using the shared buffers is a closer solution to the previous. Instead of associating a separated queue to every output, a common memory buffer is used by all outputs. Only one queue is used to hold all cells. This method is more economic in space. It has been implemented in a lot of switches: ATLANTA of Lucent Technologies, ATLAS [9] and adopted in a lot of papers [10][11][12]. The Disadvantages of this Solution are Related to:

* The complexity of the logic to control such memory since the management of a single global queue is difficult.
* The limit of the bandwidth of dynamic RAM (DRAM) that are commonly used in order to provide large buffer storage space [13].

Input Buffers: To design high-speed switches and routers, alternatives that present now a big interest are input buffers adopted in our switch and CIOQ (Combined Input Output Queues), proposed in several recent publications [7, 14, 15]. These two techniques do not require some queues operating at multiple speed of the communication lines
The most realistic solution is the one using input buffers, because it simplifies the implementation of big capacity switches. Input buffers switches require only O(N) buffers and O(N) controllers (a buffer is associated with each input for queuing the incoming cells). Also, this technique is able to overcome the technology limitation and complexity management of a single large multi queue memory. In this strategy, the memory can be accessed simultaneously by a read and write operations. The memory must therefore, present an access time at least two times faster than the line rate (Speed up S=2). This limitation can be a serious problem only in case of very high rates (several hundreds of Gb/s). With fewer speeds, such OC48c (2.5 Gb/s) or OC192c (10 Gb/s), many solutions can be proposed to surmount it:

* Using multi-ports memories where such operations can take place in parallel.
* Combined fast static SRAM with big capacity dynamic RAM [13].
* Doubling the memory bandwidth using some memory management techniques as: 'Ping Pong' [16].
* Queuing in a parallel manner the different packets [17].

A major drawback of input buffers is related to queue managing while selecting cells to transmit at every cycle. The simplest way consists in storing the incoming cells in FIFO queues. The cells at the head of queues are the first served. This approach introduces the problem of HOL blocking (the cells in the queues can be blocked by the cells in conflicts at the heads of queues). It has been demonstrated in [18], that in this case only $2-\sqrt{2} \approx 58\%$ of the input cells are served.

Several solutions have been adopted to palliate to the problem of the HOL blocking:

* Operating the switch fabric at S times faster than the input lines (speedup). A speedup by a factor of S can remove S cells from each input port within each cycle. This scheme removes completely the HOL blocking effect. Unfortunately and like output queuing, a speedup of S=N in the switch, is not feasible actually for high values of N. For low values of S, HOL blocking phenomenal is only reduced.
* In [19], the authors propose a copy of the routing network for the control. Using the self routing ability of Banyan network in order to avoid the network set up operation before transfers (usually adopted in such networks). The control network is topologically similar to the routing network. But, it only routes the labels (destination addresses) and not the data during each cycle, to find the best matching to submit to the routing network during the next cycle. In the test phase and in case of conflicts, a random selection of other cells in the queues that generate the conflicts is performed. This procedure is repeated until obtaining the maximum matching or until the end of cycle. The use of two networks, one for the control and the second for the routing is advantageous for its simplicity and allows a high throughput, but it is expensive to implement, since it needs more SEs than other solutions based on non blocking multistage networks.
* An other approach is that which increases the number of queues while adopting the solution of virtual channels or virtual queues VOQ (Virtual Output Queues). In this scheme, N virtual output queues (corresponding to the N outputs) are associated to every queue in input. A cell that arrives at an input is queued depending on its destination in the appropriate virtual queue. At each cycle, only one cell is selected from every input (i.e. from all the associated VOQ). Some scheduling algorithms are required to find the optimal solution (Maximum Matching) of maximum inputs to transmit without conflicts. Some authors [20, 21] have oriented their researches toward this solution to achieve 100% of throughput. Nevertheless and for big values of N, it needs a large memory to implement the big number of buffers: $O(N^2)$ queues for an NxN Benes network.

* A simple and economic solution but with less performances than the previous has been proposed in [22]. The number of queues at each input port is limited to two. A queue containing the cells destined to the outputs with even addresses and the second, the cells destined to the outputs with odd addresses (from where the name of the proposed switch: 'odd-even switch'). The selection is done in a first time from the even queues. In a second time, cells of the 'odd' queues, corresponding to the same inputs will replace cells that loose in the first round. Although, the performances offered by this switch are modest, this solution represents an improvement comparatively to a simple FIFO technique while offering a throughput of about 70% without introducing an important hardware complexity.
* Using Window Based Scheduling Algorithms [23, 24]. This solution improves the throughput while choosing in a window (a part of the different queues) the cells to transmit. The selected cells are not necessarily the ones in the heads of queues. When a cell in a head of queue is in conflict, the algorithm selects another one of the same queue depending on some criteria, in a window of W depth. During a cycle, a set of a maximum N cells (a cell from each queue) is selected to be transmitted.

Scheduling Algorithms: The research of the optimal solution in the queues can be translated to a research of the maximum matching in a bipartite graph. The algorithm establishes from the set X of input addresses and the set Y of destinations addresses a bipartite graph G in such a way that every edge of G joins a vertex in X to a vertex in Y. A sub set M of edges is called matching if there are not two edges of M that have a common vertex. An example of bipartite graph, correspondent to the treated example in this article is given in figure 4.

To find the optimal solution some scheduling algorithms as MSM (Maximum Size Matching) and MWM (Maximum Weight Matching) [20] use different metrics such as the length of the queue (LQF: Longest queue First), the age of the cell in the queue (OCF: Oldest Cell First). It has been proved that used in a combination with virtual output queuing (VOQ) these algorithms can achieve high throughput ≈ 100% in case of uniform traffic. However, this is only a theoretical result. They consume respectively $O(N^{5/2})$ and $O(N^3 \log_2 N)$ and are too complex to be implemented in hardware, since they need big comparators to compare the different ages, lengths or weights of the queues.

Several other algorithms with less complexity have been proposed and implemented in hardware. MUCS (Matrix Unit Cell Scheduler) [26] finds the optimal solution by computing a traffic matrix. Iterative algorithms, such iSLIP use multiple iterations to converge on a maximal matching [25]. RRM (Round Robin Matching) algorithm [27] affects a rotating priority 'round robin' to the different queues to avoid that some of them could be not served during a long time (since their lengths or weights are less competitive) and converge toward the optimal solution on average in $O(\log_2 N)$ iterations.

The optimal solution can be also obtained by the algorithms based on simple heuristics, but without guarantee of an optimal throughput. A simple example of these algorithms is the next one: At every beginning of a cycle, the cells at the head of queues are in competition to be transmitted, those that loose, let the possibility to the cells that follow and so forth on the whole width W of a window. This process can be interrupted before, if the optimal solution is reached. No metrics are used. This algorithm based on the simple heuristic consumes O(N.W) in sequential time [24].

Proposed Switch: Our solution is based on the use of Input buffers to queue the incoming cells. The general structure of the proposed switch is shown in Fig. 2.

Cell Buffers: We consider a switch operating at maximum rate of 3,2 Gb/s. The memory access time does not represent a major problem since the required memory access time in case of input buffers and single ported memory is estimated to L/(2*R)=10ns (with data word length L=64bits and line rate R=3.2Gb/s). Although, it corresponds to a faster access time than typical access time of DRAM memories (varying from 12 ns to 70 ns), with the later development of SDRAM, larger memory bandwidth becomes possible without cost increasing. The recent Double Data Rate 266-DDR SDRAM with an 8 byte wide bus, offers a bandwidth of 266 MHz x 8 Bytes = 2.1GB/s. In case of dual ported memory (simultaneous read and write operations are possible), which is very attractive for input queuing systems, we can slow down the memory access rate by half and hence achieve more than the required bandwidth.

The logic to control the input queues is simple as only a read and write pointer need to be maintained for each queue.

Selection Process: The proposed solution is a deterministic window based algorithm. To determine the cells that can be self-routed through the routing network, in one pass without conflicts, the algorithm searches a match in a window of W width. The algorithm describe in [24] establishes the optimal solution by consulting all the NxW cells. Such policies, based on the search of the maximum matching, are not appropriated for high values of N (since an exhaustive search is time consuming). We opt for a solution with less time complexity, while looking just for cells destined to outputs not addressed by cells in the HOL vector which contains the destination addresses of cells to transmit during every cycle.

The algorithm presents at the inputs of the routing network the permutation or the partial permutation, where the maximum of outputs are solicited, to be then routed automatically. This objective is reached using a counter for every output that indicates the number of the output occurrences in the HOL vector.

In a first time, an output with an empty counter (=0) does not appear in the HOL vector. A cell addressed to this output is then looked for, in the queues on the width W of the window. If found, it will replace a cell destined to an output that appears more than once in the HOL vector.

Our algorithm runs in $O(W.(N^2-N))$ sequential time in worst case (all the N cells are addressed to the same output) and O(N) sequential time in best case (all the outputs are addressed in the HOL vector).

The selection process operates in three steps. The principle is the following:

* The cells are directed (using AIG component) according to the highest weight bit of the destination address toward a buffer in high half (HM) if the bit is 0 (corresponds to outputs 0 to N/2-1), or toward a buffer in the low half (LM) if the bit is 1 (corresponds to outputs N/2 to N-1).

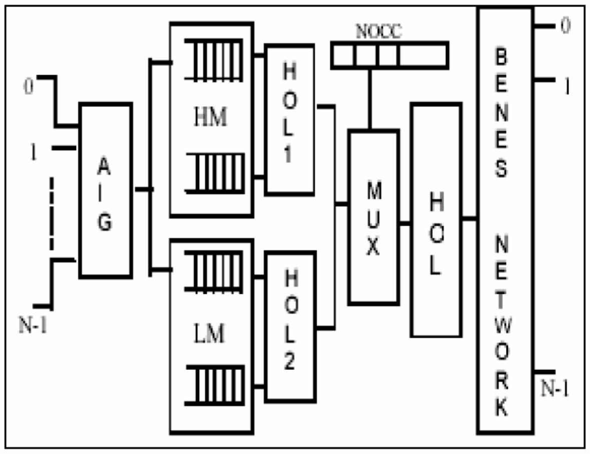

Fig. 2: Architecture of the Proposed Switch

* Compose the vector HOL1 (in parallel, HOL2) from the N cells in the head of HM queues (respectively from cells in the head of LM queues). The number of occurrences of every destination Nocc (output i) is counted. Verify that: $\forall$ i, 0<i <N/2-1,Si $\in$ HOL1 (in parallel, $\forall$ i, N/2-1<i < N-1, Si$\in$ HOL2), that means there is no occurrence counter equal to 0 or all N/2 destinations exist in HOL1 (respectively in HOL2). If this condition is satisfied, pass to step 3. Otherwise for each output i with Nocc=0, search in W for a cell destined to an output of which Nocc (i)=0, if it exists it will take a place in HOL1 (respectively in HOL2), in replacement of a cell addressed to an output of which Nocc (i) >1
* Iteratively, constitute the HOL vector to transmit while selecting cells from HOL1 and HOL2. At each iteration, HOL1(i) or HOL2(i) are in competition to take place in HOL vector. The cell with the less counter Nocc in the HOL is selected and its correspondent counter updated. If the two cells have been already selected, no cell will be transmitted from the input. A partial permutation is so obtained.

Our algorithm is illustrated through the following example:

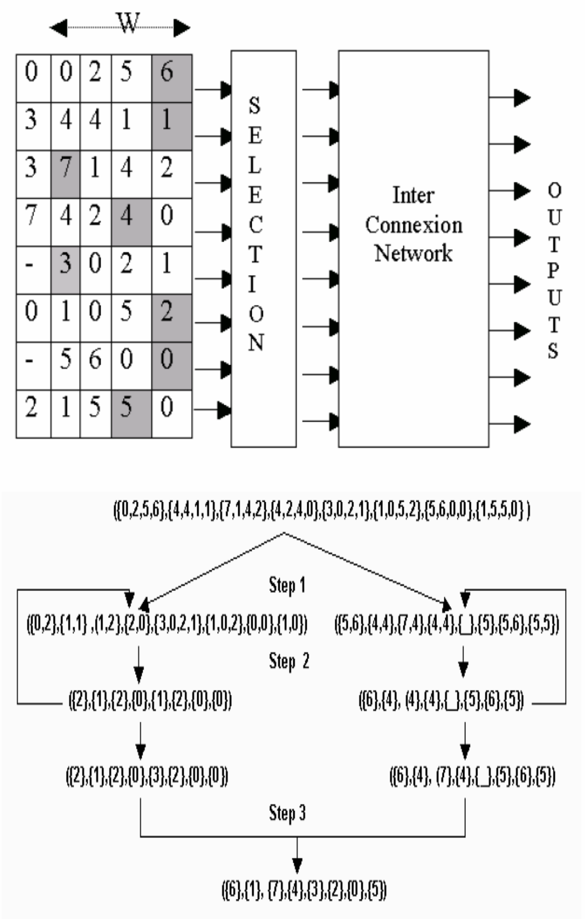

Fig. 3: An Example of Cell Selection

In case of this example, the permutation selected corresponds to the optimal solution. Yet, its research was not the objective of the algorithm. We noted through several simulations that on 100 permutations generated randomly, the solution obtained coincides with the optimal solution (maximum matching) in more than 20%. Figure 4 shows the bipartite graph corresponding to the cells of the window (W=4). The bold edges correspond to the graph of the maximum matching.

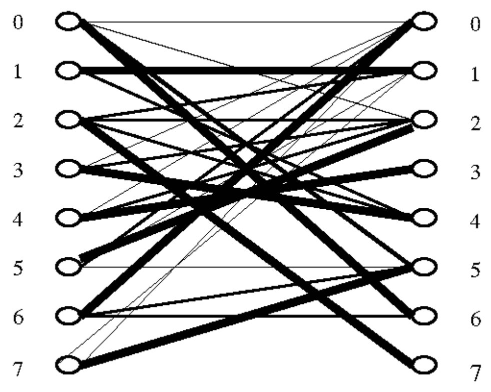

Fig. 4: Bipartite Graph

The permutation (6,1,7,4,3,2,0,5) at the output of the selection circuit is submitted to the routing network.

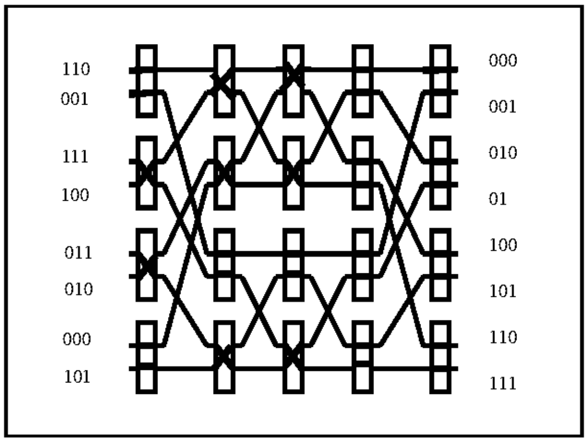

Fig. 5: Routing of P: (6, 1, 7, 4, 3, 2, 0, 5)

Simulation and Evaluation

VHDL Simulation: The switch design is based on VHDL entries. We use the standard VHDL (Very-high speed integrated circuits Hardware Description Language) to describe and verify the good functioning of the proposed switch.

A recursive construction of Benes network is used. A 2x2 SE has been described according to the algorithm given in the second section; it performs differently depending on its physical emplacement, either in the first r-1 stages or in the r following stages.

After that, we have specified the selection module in the architecture level as discussed in the previous section. We verify the functionality of each component, then we grouped all the components to form a complete switch including selection module and routing module. A functional simulation has been performed to verify the behaviour of the proposed switch. Examples of complete functioning have been verified via VHDL simulation with uniform arrivals.

RESULTS

Extensive simulations have been conducted to evaluate the performance of the proposed switch. The evaluation is made for an 8x8 switch with different sizes of the window W=2,W=4,...W=10. The circuit operates at a frequency of 100Mhz, with a 64 bit data bus. There is a maximum bandwidth of 3,2 Gb/s. On 800 cells with uniform traffic and destination cells randomly distributed among all the outputs, a set of maximum N=8 cells is delivered each cycle (= 10ns). Different performance factors are measured.

The switch delay defined (for one cell) as the average time spent between a cell arriving at the input port and departing from the switch, is shown in Fig. 6. The throughput and cell loss simulation results are given in Fig. 7.

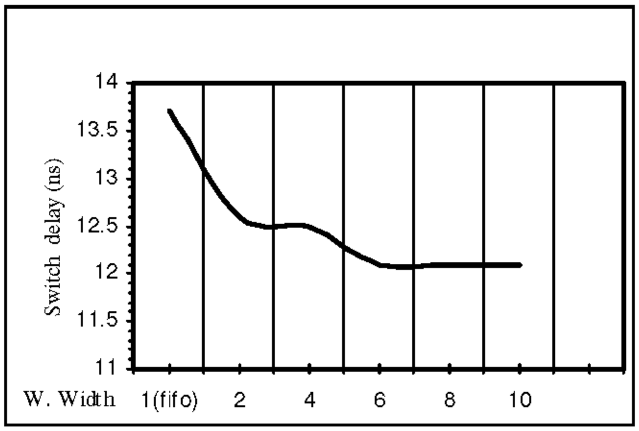

Fig. 6: Average Delay per Cell

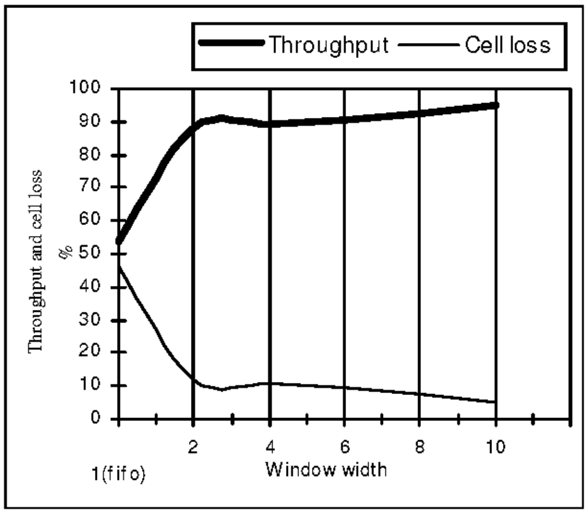

Fig. 7: Throughput and Cell Loss with Various Window Sizes

Figure 8 shows the latency at two different levels (selection circuit and routing network). We notice that the total latency is especially affected by the routing network latency. The selection procedure is faster. Thus, the research of the optimal solution could be done on larger widths of W, while exploiting the waiting times of the selection circuit. It results, an improvement of the throughput (Higher than 95% for W=10).

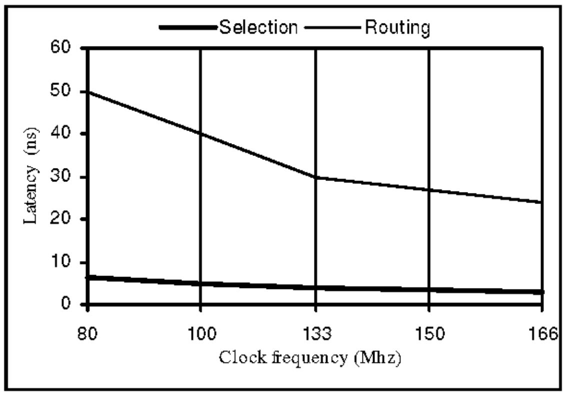

Fig. 8: Selection Circuit and Routing Network Latencies Under Different Internal Clock Rates

Cost Evaluation: The cost of a switch is closely related to the hardware complexity. The cost of input buffer switch depends on the interconnection network cost, the size and the number of buffers that it uses and the hardware complexity of the selection policy or scheduling algorithm. The cost of interconnection network is proportional to the number of switching elements and the number of links between them. Traditional techniques to find the total cost of a chip, take for granted that the cost of interconnecting subsequent stages is negligible. This means that we can assume that the cost depends only on the number of switching elements. Therefore, the cost of Benes network expressed in number of 2x2 SE is estimated to $N\log_2N-N/2$.

The switch cost remains relatively low since we adopted the following choices:

* A Benes network that presents one of the lowest costs among the non blocking multistage interconnection networks. ($N\log_2N-N/2$, against $N\log_2N$ for Banyan networks and N^2 for crossbar).
* No global controller is needed.
* A simple selection policy that does not require some big comparators, or other complex circuits doing some operations on a big number of bits.
* The number of queues or buffers used is limited to $O(N)$, comparatively to $O(N^2)$ what it is used in other solutions adopting virtual queues (VOQ)

The throughput and time complexity of our solution are compared to others, especially FIFO, simple and optimal heuristics and odd-even switch, because among the studied approaches in this study, they present the minimum hardware and time complexity.

The throughput results given in Table 1, correspond to the solution using window based scheduling algorithms for w=2, which is analogous to the Odd-Even model where contention resolution also consists of two rounds (one for polling the odd and one for polling the even queues). With higher values of W, it is obvious that the obtained throughput applying these algorithms will be better.

Table 1: Comparison with Other Solutions

Method	Time complexity	Throughput
FIFO	$O(N)$	0,586
Simple Heuristic	$O(WN)$	0,68
Optimal	$O(WN^{3/2})$	0.71
MWM	$O(N^3\log_2N)$	1.00
Odd-Even	$O(2N)$	0,71
Our solution	$O(W(N^2-N))$	0,877

The experimental results show that a throughput improvement of about 50% and 23% is achieved in comparison respectively with the FIFO strategy, Odd-Even and optimal heuristics under uniform traffic.

Future simulations are projected to study the functioning of the switch under other arrival models, especially under burst traffic.

CONCLUSION

In this study, we have attempted to show that to design high performance packet switches, theoretical solutions related to buffering, selection and routing switch functionalities could be defined. However, in practice many of these solutions providing 100% of throughput are not foreseeable.

Really, the designer has to find the best compromise between actual implementation and performances. Our proposed switch has a better throughput comparatively to studied switches involving minimum hardware and time complexity. In our design, we avoided every choice that can improve throughput and switch delay, but will have a significant impact on cost.

Finally and as future work, we plan to synthesise the switch and implement it using FPGA technology. The expected result will be a low-cost high-performance packet switch in a single chip.

REFERENCES

1. Beauquier, B. and E. Darot, 1997. On Arbitrary Waksman Networks and their Vulnerability. Tech. Rep. No. 3788, INRIA.

2. Lenfant, J., 1978. Parallel permutations of data: A benes network control algorithm for frequently used permutations. IEEE Transactions on Computers, 27: 637-647.
3. Nassimi, D. and S. Sahni, 1982. Parallel permutation and sorting algorithms and new generalized connection network. J. ACM, pp: 642-667.
4. Boppana, R.V. and C.S. Raghavendra, 1990. optimal self-routing of linear-complement permutations in hypercubes. Fifth Distributed Memory Computing Conference (DMCC-5), pp: 800-808.
5. Das N., K. Mukhopadhyaya and J. Dattagupta, 1992. Self routing in Benes network. Tech. Rep. No. E/02/92 Indian Statistical Institute, Calcutta, India.
6. Raghavendra, C.S. and V. Boppana, 1991. On Self routing in Benes and Shuffle-Exchange networks. IEEE Transactions on Computers, 40: 9.
7. Chuang, S.T., A. Goel, N. McKeown and B. Prabhakar, 1999. Matching output queuing with a combined input output queued switch. INFOCOM 99, New York, USA.
8. Brown, T., 1999. A high performance two-stage packet switch architecture. IEEE Transaction on Communication, 47: 1792-1795.
9. Kornaros, G., D. Pnevmatikas, P. Vatsolaki, G. Kalokerios, C. Xanthaki, D. Mavroidis, D. Serparos and M. Katerimis, 1998. Implementation of ATLAS 1: A single chip ATM switch with backpressure. Proceeding of IEEE hot Interconnects VI Symposium. Standford University, California USA.
10. Lauer, H.C., A. Ghosh and C. Shen, 1994. A general purpose queue architecture for atm switch. Technical Report 97-17, Mitsubishi Electric Research Laboratories.
11. Tutsch, D., M. Hendler and G. Hommel, 2001. Multicast Performance of Multistage Interconnection Networks with Shared Buffering. P. Lorenz (Ed.): ICN 2001, LNCS 2093, pp: 478-487.
12. Lyer, S. and N. McKeown, 2001. Techniques for Fast Shared Memory Switches. HPNG Tech. Rep., Stanford University TR01-HPNG-081501.
13. Garcia, J., J. Corbal, L. Cerda and M. Valero, 2003. Design and implementation of high-performance memory systems for future packet buffers. IEEE Proceedings of the 36th International Symposium on Microarchitecture.
14. Cessa, R., E. Oki, Z. Jing and H.J. Chao, 2001. cixb-1: combined input-once-cell-crosspoint buffered switch. IEEE Workshop on High Performance Switching and Routing, Dallas, TX.
15. Yang, M. and S.Q. Zheng, 2003. An Efficient Scheduling Algorithm for CIOQ Switches with Space-Division Multiplexing Expansion. IEEE INFOCOM.
16. Joo, Y. and N. McKeown, 1998. Doubling memory bandwidth for network buffers. IEEE INFOCOM, 2: 808-815, San Francisco.
17. Lyer, S. and N. McKeown, 2003. Analysis of the parallel packet switch architecture. IEEE/ACM Transactions on Networking, 11: 2.
18. Karol, M., M. Hluchyj and S. Morgan, 1987. Input versus output queuing on space division switch. IEEE Transaction Communications, pp: 1347-1356.
19. Boppana, R.V. and C.S. Raghavendra, 1999. Designing efficient benes and banyan based input-buffered ATM switches. ICC'99, Vancouver, Canada.
20. McKeown, N., A. Mekkittikul, A. Venkat and J. Walrand 1999. Achieving 100% throughput in an Input- queued switch. IEEE Transactions on Communications, 47: 8.
21. Keslassy, I. and N. McKeown, 2001. Analysis of Scheduling Algorithms That Provide 100% Throughput in Input-Queued Switches. Proc. of the 39th Annual Allerton Conference on Communication Control and Computing, Monticello, Illinois.
22. Kolias, C. and L. Kleinrock, 1996. The odd-even input queuing ATM switch: Performance evaluation. ICC., 96.
23. Cam, H., 2000. Preventing internal and external conflicts in an input buffering reverse baseline ATM switch. Intl. J. Comm. Sys., 13: 317-334.
24. Moh, W.M. and Y.F. Chung, 1997. Design and evaluation of cell scheduling algorithms for ATM switches. Proc. of IEEE Singapore Intl. Conf. on Networks, pp: 355-369, World Scientific.
25. McKeown, N., 1999. iSLIP: A scheduling algorithm for input-queued switches. IEEE/ACM Transactions on Networking, 7: 188–201.
26. Duan, H., J. Lockwood and S.M. Kang, 1998. Matrix Unit Cell Scheduler (MUCS) for input-buffered ATM switches. IEEE Communications Letters, 2: 1.
27. Yihan, Li, P. Shivendra and H.J. Chao, 2001. On the Performance of a Dual Round-Robin Switch. IEEE INFOCOM, pp: 1688-1697.
28. Park, Y.K., V. Cherkassky and G. Lee, 1993. ATM cell scheduling for broadband switching systems by neural network. Proc. of Intl. Workshop on Applications of Neural Networks to Telecommunications (IWANNT), Princeton, pp: 112-118.