PERFORMANCE ANALYSIS OF LPF BASED VOQ CROSSBAR SWITCHES

Milutin Radonjić, Igor Radusinović[1]

ABSTRACT*:*
In this paper, LPF based schedulers for VOQ crossbar switch implementation through FPGA circuits are analysed. Switch performance (average cell latency, queue lengths and loss probability) are evaluated under different traffic conditions. Results of behavioral simulations on implemented circuits are compared with simulations on modeled circuits that are not limited as implemented ones. Particularly consideration is given to required queue lengths and impact of its limitation to scheduler behavior. It is shown that queue length of 127 cells is enough to manage almost any offered load under various traffic conditions that we considered.

Index Terms: Crossbar switch, Cell delay, Loss probability, Queue length, Scheduler, Verilog.

I. INTRODUCTION

The importance of switches and routers in telecommunications networks has risen up along with increasing of Internet traffic. This fact caused intensive research in domain of switching fabrics to achieve throughput as higher as possible. There can be found different implementations of switching fabrics, as well as scheduling algorithms, in available literature.

One of the most used architectures for design of high-speed switches is crossbar architecture. Crossbar switching fabric is attractive for implementation because of its non-blocking capability and simplicity. Switching of fixed length packets (referred as cells) is a widely accepted method for design of high-speed packet switches. Variable length packets are segmented into cells upon arrival, transferred across the switch fabrics and then reassembled again before they depart.

[1] Milutin Radonjić, Igor Radusinović, Faculty of Electrical Engineering, University of Montenegro

Buffer that accepts arrived packets was primarily organized as set of input queues. But systems that use input queues have a potential problem due to head of line (HOL) blocking. This problem can be eliminated entirely using a queuing technique known as virtual output queuing (VOQ) [1] - [3] in which each input maintains a separate queue for each output.

In this paper, we analyze implementation and performance of crossbar switching fabric 4x4, with VOQ buffering method. To control crossbar scheduler, an iterative Longest Port First (iLPF) algorithm [3], is implemented. Since iLPF algorithm meet performance degradation under some traffic conditions, we treated some of its variants. The best overall performances are achieved using Longest Input Port First with Throughput Maximization (LIPFwTM) algorithm [2].

Since we discovered some instability for 4x4 crossbar scheduler with unbalance traffic and maximum offered load, analysis of 32x32 crossbar scheduler under the same traffic conditions is presented.

Implementation of scheduler hardware is explained in section 2. Section 3 covers simulation results under three traffic models: uniform, Interrupted Bernoulli Process (IBP) and unbalanced [4] (subsections a, b and c, respectively). Some aspects of actual scheduler implementation are presented in section 4. Final conclusions are given in section 5.

II. SCHEDULER HARDWARE IMPLEMENTATION

Crossbar switch scheduler has been implemented by Verilog in Xilinx Integrated Development Environment [7]. Block diagram of scheduler is shown on Fig.1.

For scheduler with iLPF algorithm, current occupancies of virtual output queues (in the observed time slot) for each input and output port are sorter inputs. In the case of LIPFwTM algorithm, sorter input is also current occupations of VOQ for input ports. Other input is not current occupation, but number of non-empty VOQs for each output port [2].

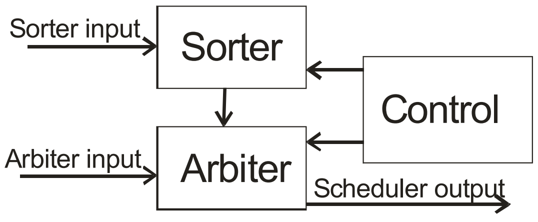

Fig. 1 – Block diagram of crossbar scheduler

Sorter output, in either case is sorted value of input parameters [1]. We implemented this sorter as a variant of Batcher sorting network [5]. Internal implementation of this sorting network and detailed configuration of one node are shown on Fig.2 and Fig.3, respectively.

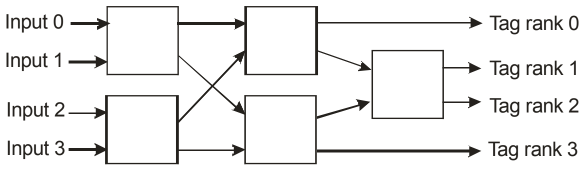

Fig. 2 – Sorting network

Since it is necessary to sort two groups of independent data (occupancies of VOQ buffers for each input and output port), two sorting networks should be instantiated to do the job simultaneously. Although, we implemented only one sorter for the reason of implementation simplicity and minor use of logic gates. Two groups of input data are sorted in time shifted moments inside the same time slot. This can be achieved by connecting memory elements (latches) to input and output of the sorter.

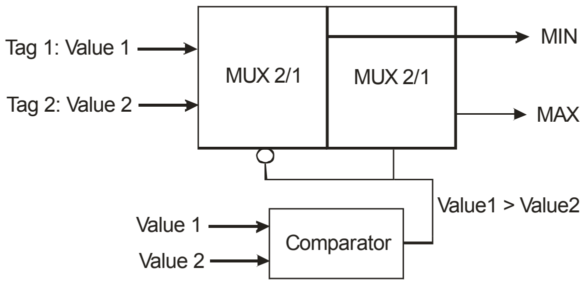

Fig. 3 – One node of sorting network

Arbiter input is a request matrix for packet transfer. This matrix contains information about packet's existence in VOQs. Request matrix is first reordered by rows according to sorter outputs, so that input port with longest occupancies comes to row 0 and input port with shortest occupancies comes to row 3. Then, it is reordered by columns in the same manner, according to VOQ occupancies for output ports [3]. Reordered matrix is input of the arbiter circuit shown on Fig.4.

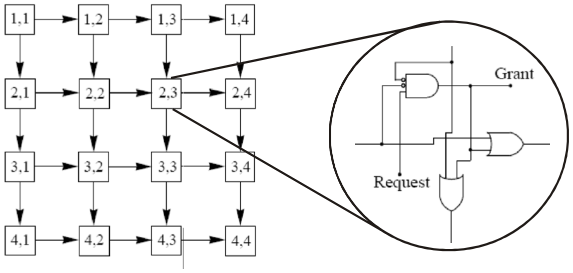

Fig.4 – Arbiter circuit

Arbiter circuit output is a grant matrix that contains information which input and output ports will be matched in this time slot. To achieve a correct grant matrix, adequate to request matrix, it is necessary to re-permute arbiter circuit's output in reversible direction by rows and by columns. Re-permuted grant matrix is output of complete scheduler [1].

III. SIMULATION RESULTS

With the validation process of implemented device we simulated scheduler's behaviour under different arrival traffic conditions. Crossbar switch performances are evaluated for uniform, Interrupted Bernoulli Process (IBP) [6] and unbalanced traffic models [4].

During the initial testing of iLPF algorithm implementation, for uniform traffic model, we discovered that grant matrix generated by this scheduler is not optimal. Actually, at some time slots matrix delivered less matches then was able to, according to input requests. For that reason, we were looking for some improvements, to use switching fabric in a better manner. Longest Input Port First with Throughput Maximization (LIPFwTM) [2] is algorithm with such a feature. Those two algorithms are different only by kind of data at the input of scheduler, so its hardware implementation is identical in both cases. In this paper, iLPF and LIPFwTM performances comparison is presented.

Relevant parameters for scheduler performance analysis are *average cell latency* and *average throughput* [1]. Results for average throughput will not be presented because evaluated algorithms achieve almost 100% throughput for most simulated traffic conditions.

These parameters are usually evaluated for unlimited input buffer size. However, with hardware implementation, size of this buffer has to be determined in advance. Chosen algorithms define number of input wires to the scheduler, according to a buffer size. Through these wires buffer sends information about its occupancy to the scheduler. For these reasons, we evaluated three more parameters: *loss probability*, *maximum input port length* and *maximum output port length*.

Loss probability is a ratio of rejected and arrived packets to the buffer input. Maximum input port (output port) length is a maximum value of VOQ occupancy for input (output) ports that was sent to scheduler during simulation, as a result of generated traffic. Both input and output port lengths are observed for iLPF algorithm. Only input port length is observed for LIPFwTM, because instead of output port length it receives number of non-empty virtual output queues, for each output port. Maximum value of this number can not exceed number of switch ports. Based on these parameters, an estimation of possible packets lost can be done for certain traffic conditions, according to limited number of scheduler input wires. Also, these cognitions can assist us to select optimum buffer size.

A. Analysis for uniform traffic

Diagram of average cell latency over the offered load (in further text labelled by p) for both algorithms under uniform traffic and unlimited buffer size is shown on Fig. 5. Simulations are performed on the hundred million time slots. From this diagram it can be noticed that LIPFwTM has smaller average cell latency for any offered load less than 0.99. At the range of offered load between 0.9 and 0.98 LIPFwTM has twice less latency then iLPF.

Maximum values of port lengths for previous simulations are shown on Fig. 6. As can be seen from this Figure, LIPFwTM has better performance, requiring lower maximum port lengths over offered load.

After these simulations we evaluated system behaviour under limited buffer size, i.e. limited number of wires that transfer port lengths information to the scheduler. Results of this evaluation are shown on Figures 7 and 8.

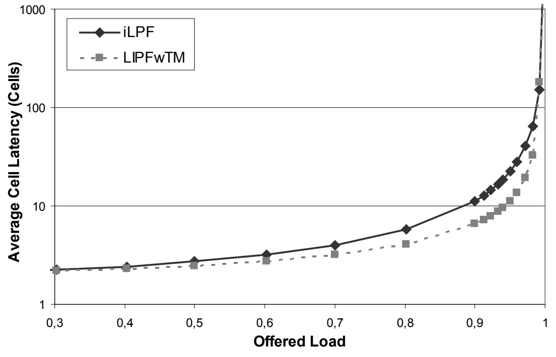

Fig. 5 - Average Cell Latency for uniform traffic

Maximum allowed value for port length in the diagram is labelled by letter "L". For example, L=3 means that maximum occupation of port is three. In that case, two wires are provided for each port to transfer length information toward the scheduler. Ten wires would be provided for L=1023. LIPFwTM algorithm is labelled by letter "K" (Figure 8.) to be distinguished from iLPF algorithm.

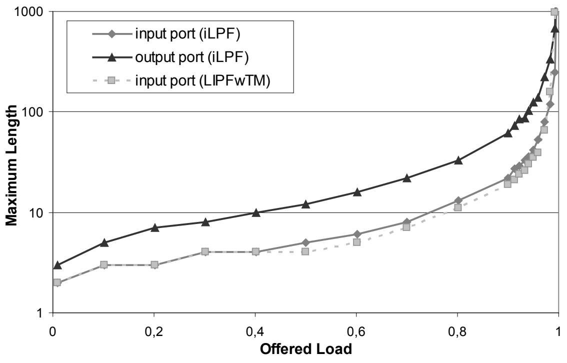

Fig. 6 - Maximum port lengths

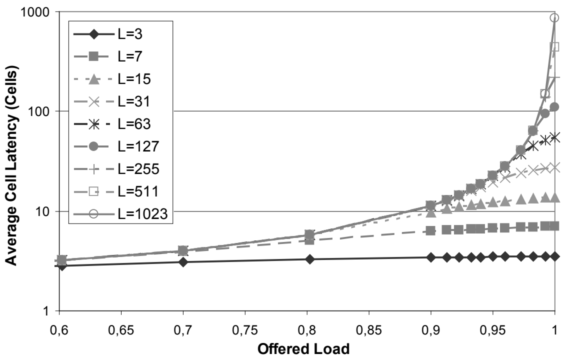

Fig. 7 - Average Cell Latency for iLPF algorithm with limited port lengths

Similar results and the same trends for both algorithms can be noticed by observing those diagrams. As expected, with larger buffer (more input wires) diagrams look more like those on Fig. 5.

For better comparison of two algorithms we isolated diagrams for several values of parameter "L" on the same chart (Fig. 9). From this chart it is obvious that LIPFwTM has better performance, except for very small value of parameter "L" (L=3).

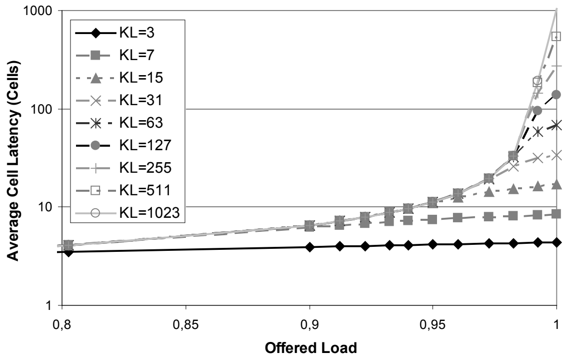

Fig. 8 - Average Cell Latency for LIPFwTM algorithm with limited port lengths

Lightly observing diagrams on Figures 7, 8 and 9, one can conclude that system with shortest buffer length provide the best performance. Indeed, in that case there is smallest average cell latency. However, since not all incoming cells will be accepted, we must observe how many cells will be rejected due to the buffer overflow. Loss probability diagram, under different offered loads and input buffer length limitations for iLPF algorithm, is shown on Fig. 10. The similar diagram for LIPFwTM algorithm is shown on Fig. 11.

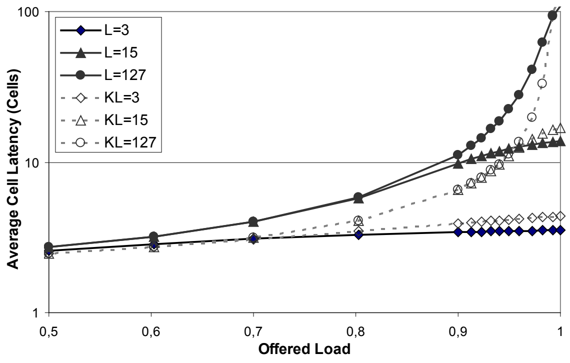

Fig. 9 - iLPF and LIPFwTM algorithm diagrams on the same chart for some values of "L"

For better comparison of two algorithms distinctive diagrams from Fig. 10 and 11 are presented on the same chart (Fig. 12). From this chart it is obvious that LIPFwTM has lower loss probability for a broad range of traffic load and queue lengths. This is not the case only for very high offered load (higher than 0.98). For offered load less then 0.98 and queue lengths longer or equal to 127, we did not observe any lost cells for both algorithms.

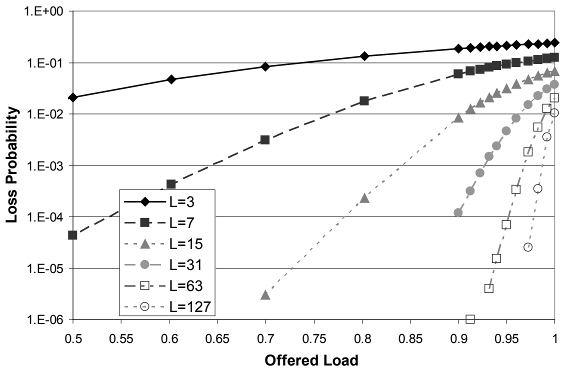

Fig. 10 - Loss probability for iLPF algorithm with limited port lengths

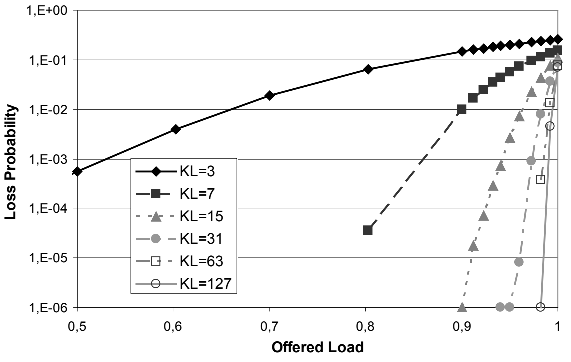

Fig. 11 - Loss probability for LIPFwTM algorithm with limited port lengths

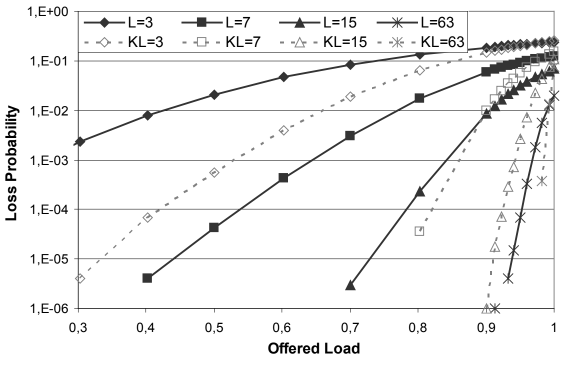

Fig. 12 - Loss probability comparison

From all mentioned above, it can be concluded that scheduler with LIPFwTM algorithm shows better performance, under uniform traffic with offered load less then 0.98, than scheduler with iLPF algorithm.

B. Analysis for IBP traffic model

Interrupted Bernoulli Process (IBP) is more realistic for network traffic modelling. Explanation of this model and traffic generation process can be found in [6]. We observed the same performances as previously, for a range of offered load and different burst parameter (Bs), on million time slots. In this paper, we are presenting simulated system behaviour for several burst values: 2, 4, 8, 16, 32 and 64. Burst values larger then 64 do not have too much sense for 4x4 switching fabric.

For unlimited buffer size, Fig. 13 illustrates dependence of average cell latency over offered load and parameter burst. Label "K" is used for LIPFwTM algorithm results, again. LIPFwTM algorithm has slightly lower values of cell latency then iLPF algorithm for any offered load except p=1.

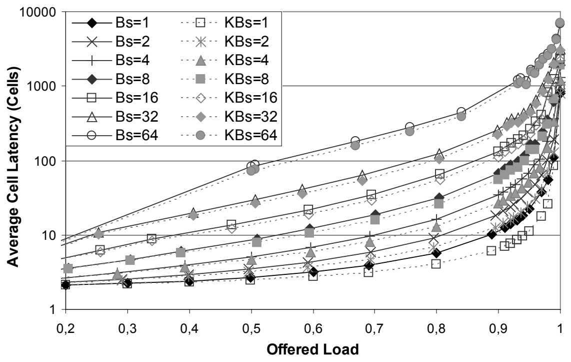

Fig. 13 - IBP traffic model

Maximum input and output port lengths, for different burst values (Bs) with iLPF algorithm are depicted on Fig. 14. For lower burst values (2, 4, 8) maximum port lengths are relatively small, especially for offered load less then 0.9. Burst values of 32 and 64 are too large for 4x4 switching fabric, anyway. These are presented for completeness of evaluation. Also, Bs=1 denotes uniform traffic.

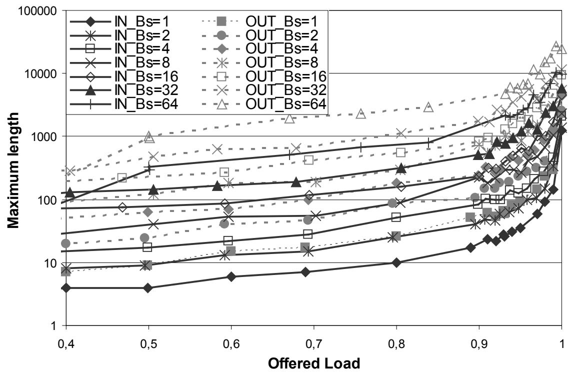

Fig. 14 - Maximum port lengths for iLPF under IBP traffic

Required input port lengths for LIPFwTM algorithm, with various burst values are presented on Fig. 15. Generally, LIPFwTM algorithm requires smaller queues then iLPF, under the same traffic conditions. IBP traffic with Bs≤8 and offered load up to 0.9 can be totally served with queue length of only 127 cells (7 wires per port).

We considered limited buffer size, similarly as in the case of uniform traffic. As an illustration of system behaviour, average cell latency diagrams for burst values of 2, 8 and 32 are presented on Fig. 16, 17 and 18, respectively.

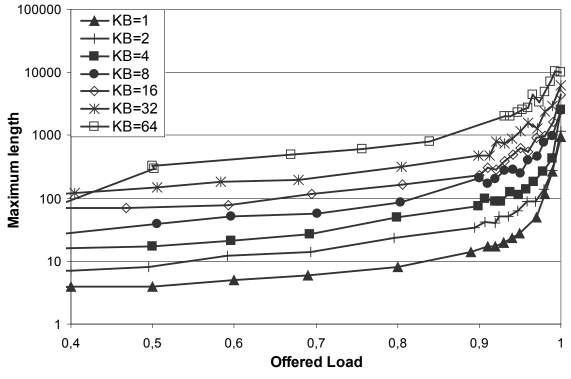

Fig. 15 - Maximum port lengths for LIPFwTM under IBP traffic

According to already mentioned, loss probability has to be observed. Due to the limited space, there will be only presented charts for the same values of burst as in the above diagrams: 2, 8 and 32 (Fig. 19, 20 and 21, respectively).

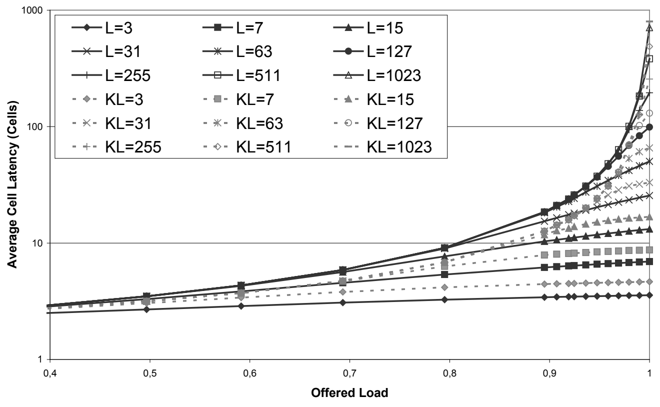

Fig. 16 - Average cell latency for Bs=2

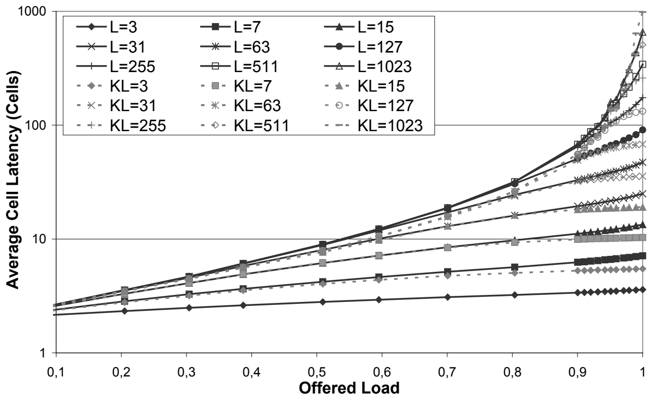

Fig. 17 - Average cell latency for Bs=8

From these diagrams one can notice that for small burst (Bs) everything behaves very similar to uniform traffic. However, for larger burst values system behaviour depends on buffer size limit. For more restrictive limits iLPF algorithm has an advantage. For instance, if Bs=32 (Fig. 21) with queue length (L) of 3, 7 and 15, iLPF algorithm has smaller loss probability. But, with less restrictive limit (L=63 or more) LIPFwTM algorithm is a better choice.

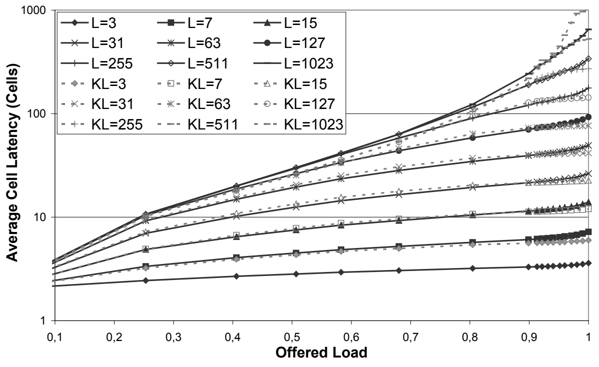

Fig. 18 - Average cell latency for Bs=32

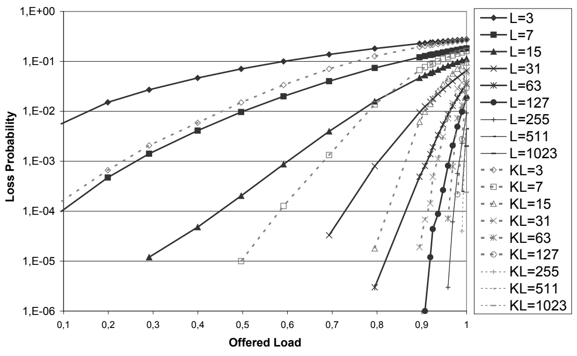

Fig. 19 - Loss probability for Bs=2

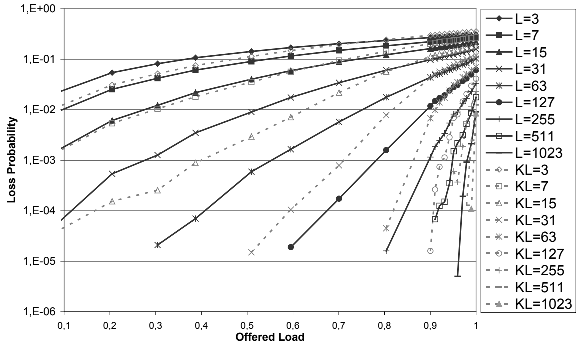

Fig. 20 - Loss probability for Bs=8

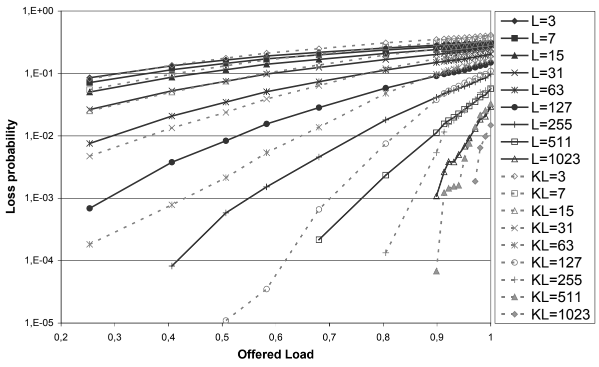

Fig. 21 - Loss probability for Bs=32

C. *Analysis for unbalanced traffic model*

For unbalanced traffic model [4] we also considered average cell latency, but in some different circumstances. At the beginning, we did not simulate system's behaviour for a range of traffic load, but only for offered load equals to one. In this case, we varied value of unbalanced probability (w). Average cell latency diagram for unlimited buffer size, obtained in simulation with ten million time slots, is shown on Fig. 22.

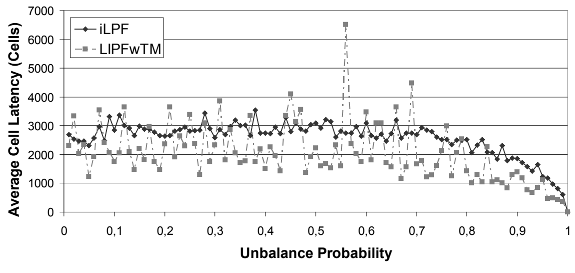

Fig. 22 - Average cell latency for unbalanced traffic with p=1

Although LIPFwTM algorithm shows trend of lower cell latency, it strongly depends on actual traffic pattern along with unbalanced probability. By observing diagram on Fig. 22, suspicion of system with LIPFwTM algorithm instability occurs. For that reason, we evaluated scheduler behaviour under several more offered loads. Results of these simulations are presented on Fig. 23. Prefix "K" labels LIPFwTM algorithm.

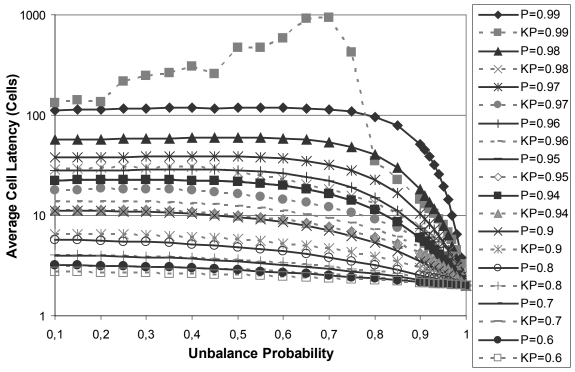

Fig. 23 - Average cell latency for unbalanced traffic with different offered loads

LIPFwTM algorithm still shows sign of instability for offered load of 0.99. Under less offered load (p≤0.98) scheduler shows stable behaviour, with lower latency for higher unbalance.

Better observation can be made by looking isolated diagram for some selected offered load, like on Fig. 24. LIPFwTM algorithm has twice less latency then iLPF for a broad range of unbalance probability, with offer load of 0.97. Generally, LIPFwTM has lower cell latency for all unbalanced traffic conditions, although that difference decrease for smaller traffic load. Exception is a very high offered load of 0.99, where LIPFwTM shows some unpredictable behaviour.

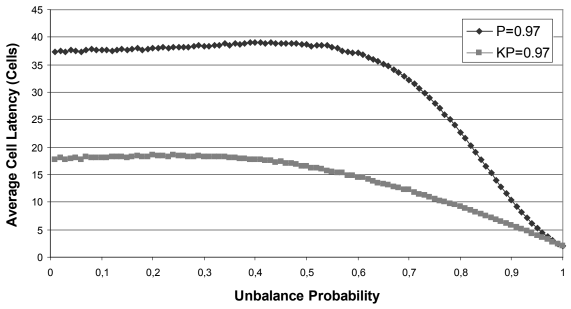

Fig. 24 - Average cell latency for unbalanced traffic with offered load of 0.97

Analysis of maximum queue lengths has also begun with offered load of 1. Results of this simulation are presented on Fig. 25. Both algorithms show strong dependence on actual traffic pattern, rather then just unbalance probability. Hence, conclusion about instability intrudes again, although LIPFwTM algorithm requires smaller queues.

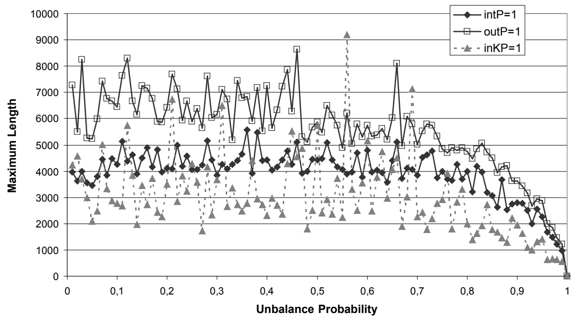

Fig. 25 - Maximum queue lengths for unbalanced traffic with p=1

Further evaluation, with more traffic loads, has been implemented (Fig. 26 and 27). For p=0.99 system still behaves unstable, especially with LIPFwTM algorithm.

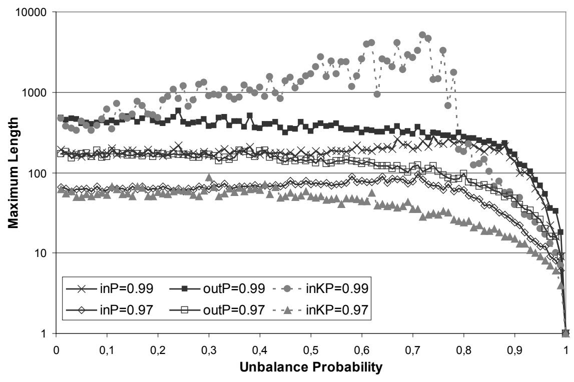

Fig. 26 - Maximum queue lengths for unbalanced traffic with p=0.99 and p=0.97

Maximum input queue length rapidly changes value for different unbalance probability. For load of 0.97 or less system shows obvious trend: LIPFwTM algorithm can be implemented with 128 cells queue length, without its overflow. For the same queue length with iLPF algorithm overflow will be omitted with offered load of 0.95 or less.

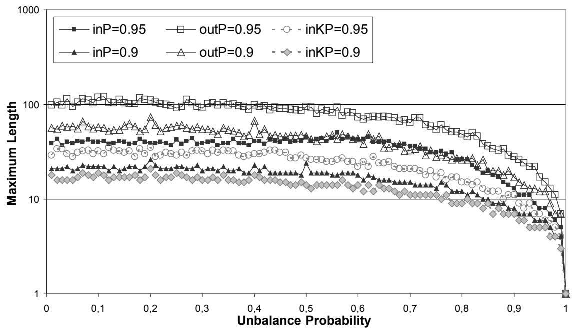

Fig. 27 - Maximum queue lengths for unbalanced traffic with p=0.95 and p=0.9

Simulation results of average cell latency with both algorithms for p=1, considering limited queue lengths, are shown on Fig. 28. Similar as with other traffic models, cell latency increases for larger queue (port) lengths. LIPFwTM algorithm has larger latency then iLPF, with almost all port lengths, for p=1.

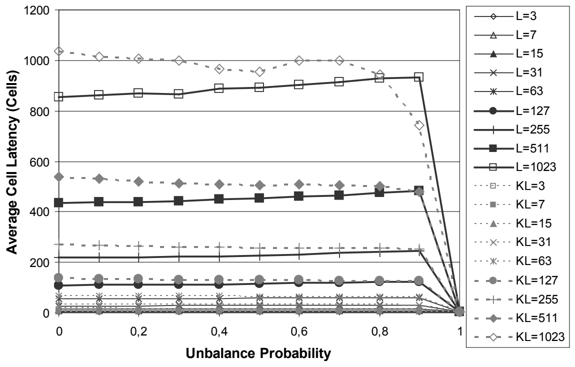

Fig. 28 – Average cell latency for unbalanced traffic with p=1

If we consider few more traffic loads, diagrams on Figures 29, 30 and 31 will be generated. Scheduler instability for LIPFwTM algorithm can be noticed from Fig. 29, for longer queues and for p=0.99. For shorter queues (length less or equal of 127) scheduler shows stable behaviour with small latency. With offered load equals to 0.98 this algorithm shows stable behaviour with small latency for any queue size (Fig. 30). Further diagrams for lower traffic load are not presented, due to space limitation, but show the same trends.

Scheduler with iLPF algorithm behaves stable for all offered loads and queue lengths (Fig. 31). Other diagrams, with lower loads, are also omitted but have identical shape and smaller latencies.

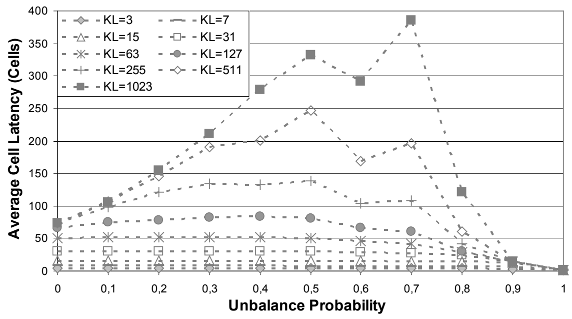

Fig. 29 - Average cell latency for LIPFwTM with unbalanced traffic and p=0.99

Loss probability for unbalanced traffic with offered load equals to 1 and different queue lengths is shown on Fig. 32. For very short queues, loss probability is relatively high and mostly falling with higher unbalance probability.

Loss probability diagrams for lower traffic loads are omitted, but have similar form as one on Fig. 32. Values of loss probability decrease with lower traffic loads. This is expected behaviour from earlier discussion on maximum required queue lengths (Figures 26 and 27). For queue lengths of 127 or more with LIPFwTM algorithm we did not observe any lost cells under offered load lower or equal to 0.97.

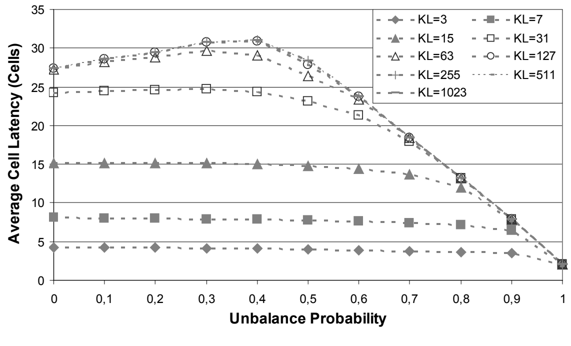

Fig. 30 - Average cell latency for LIPFwTM with unbalanced traffic and p=0.98

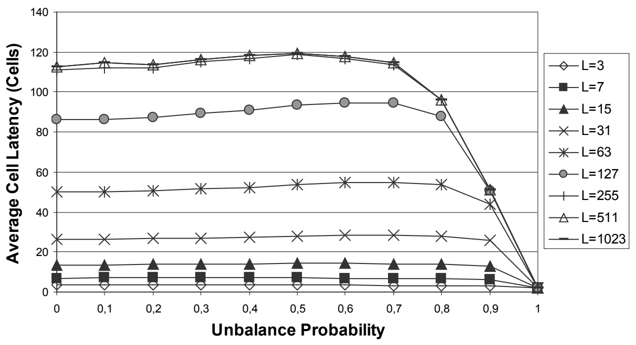

Fig. 31 - Average cell latency for iLPF with unbalanced traffic and p=0.99

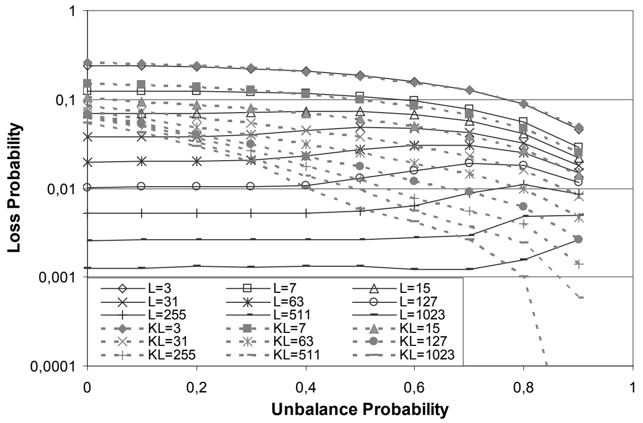

Fig. 32 - Loss probability for unbalanced traffic with p=1

From Figures 22 and 25, 4x4 crossbar switch instability, for offered load of one (p=1) with unbalanced traffic and unlimited buffer size, can be noticed. It would be interesting to observe system behaviour for different number of ports.

As an example, we analysed 32x32 crossbar switch under unbalanced traffic with p=1. Diagram of average cell latency for such a case is shown on Fig. 33. From this diagram can be noticed that there is not any sign of instability for 32x32 crossbar switch. Also, iLPF algorithm has far more latency then LIPFwTM algorithm, in this case.

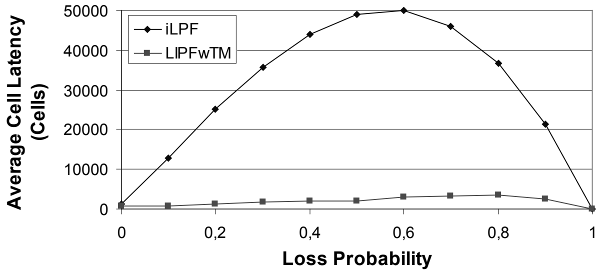

Fig. 33 - Average cell latency for unbalanced traffic with p=1 and 32x32 crossbar switch

Another interesting parameter toward instability is maximum queue length for unbalanced traffic with offered load p=1. These diagrams, for both algorithms with 32x32 crossbar switch, are presented on Fig. 34.

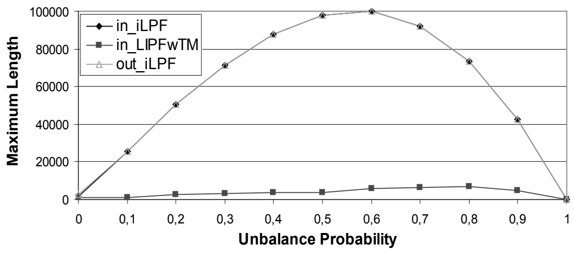

Fig. 34 – Maximum queue lengths for unbalanced traffic with p=1 and 32x32 crossbar switch

Maximum queue lengths for LIPFwTM algorithm have significantly less values than lengths for iLPF algorithm. It means that LIPFwTM algorithm has even more advantage over iLPF for larger switch with unbalanced traffic.

It is interesting to notice that input and output port lengths for iLPF algorithm have almost identical values for whole range of unbalance probability. Also, there are not any sign of instability on diagrams from Fig. 34.

IV. SCHEDULER IMPLEMENTATION

After analysis with three traffic models, it is shown that LIPFwTM algorithm has better performance then iLPF, for most analyzed traffic conditions. Beside, according to analysis results, optimum decision would be to implement scheduler for port length limit of 127 (L=127). It means that seven wires would transfer information about buffer occupancy for each port. Such a system gives very good performance for any traffic model observed in this paper.

However, for the reason of simplicity, we present here implementation of scheduler with queue length of 15 cells (4 input wires per port). Due to results verification, behavioural simulation was performed under the very same traffic conditions like in simulations.

Diagram of some signals at the end of behavioural simulation under uniform traffic on ten thousand time slots, for offered load of 0.8 (p=0.8), is shown on Fig. 35. Request signal is actually request matrix based on buffer occupancy. It is arbiter input (Fig. 1). Signals *port(0,1,2,3)duzina* are queue lengths (input and output port lengths) transferred to the scheduler. So, these signals are sorter inputs (Fig. 1). Signals *duzina(0,1,2,3)index* are produced by sorter and connected to arbiter. These are indexes of input ports ordered according to their port lengths. Signal *grant_valid* signalize that current arbitration is finished and grant matrix (signal *grant*) is produced.

V. CONCLUSIONS

Analysis presented in this paper approved our approach to crossbar scheduler design and implementation. Results obtained for scheduler with seven input wires per port (queue length of 127) show very good performance under all three observed traffic models in a broad range of offered load (p≤0.98). For that reason, its hardware implementation is our next task.

It is also shown that LIPFwTM algorithm is better choice than iLPF for most traffic conditions. With all three observed traffic models LIPFwTM has lower average cell latency and loss probability for both limited and unlimited buffer size. Only exceptions are cases with very small buffer size for IBP and very high offered load for unbalanced traffic (p>0.98). We believe that LIPFwTM is unstable in these circumstances.

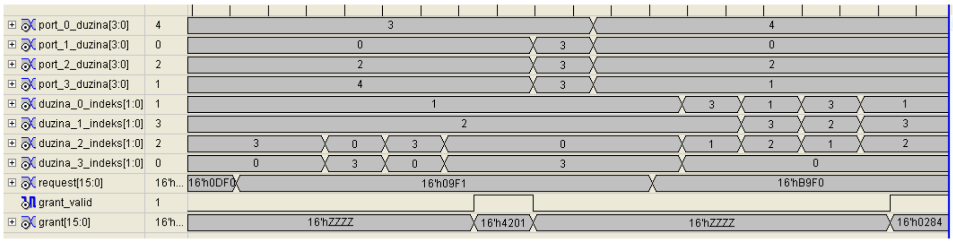
port_0_duzina[3:0] 4 3 4
port_1_duzina[3:0] 0 0 3 0
port_2_duzina[3:0] 2 2 3 2
port_3_duzina[3:0] 1 4 3 1
duzina_0_indeks[1:0] 1 1 3 1 3 1
duzina_1_indeks[1:0] 3 2 3 2 3
duzina_2_indeks[1:0] 2 3 0 3 0 1 2 1 2
duzina_3_indeks[1:0] 0 0 3 0 3 0
request[15:0] 16'h... 16'h0DF0 16'h09F1 16'hB9F0
grant_valid 1
grant[15:0] 16'h... 16'hZZZZ 16'h4201 16'hZZZZ 16'h0284

Fig. 35 - Some signals after behavioural simulation

Our further research in this area will focus on similar performance analysis for tested switches with more input and output ports. Special interest will be paid to find solution for detected instability of LIPFwTM algorithm.

REFERENCES

[1] A. Mekkittikul, "Scheduling non-uniform traffic in high speed packet switches and routers", PhD. Thesis, Stanford University, 1998.

[2] N. H. Liu, K. L. Yeung, D. C. W. Pao, "Scheduling Algorithms for Input-queued Switches with Virtual Output Queueing", Proc. Of ICC'2001, Helsinki, Finland, June 2001.

[3] A. Mekkittikul, N. McKeown, "A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches", IEEE Infocom 98, San Francisco, April 1998, Vol 2, pp. 792-799.

[4] I. Radusinovic, Z. Veljovic, "Exhaustive limited service round robin matching algorithm for buffered crossbar switches", Proc. of Telsiks 2005, Nis, Serbia and Montenegro, September 2005.

[5] K. E. Batcher, "Sorting networks and their applications", Spring Joint Computer Conference, AFIPS Proc. vol. 32, pp 307-314, 1968.

[6] I. Radusinovic, Z. Veljovic, M. Pejanovic, Z. Petrovic, "Impact of Scheduling Algorithms of Performances of Buffered Crossbar Switch Fabric", IEEE ICC 2002, New York, USA, April 2002.

[7] www.xilinx.com