Optimization of the Scheduler for the Non-Blocking High-Capacity Router

Miloš Petrović, *Student Member, IEEE,* and Aleksandra Smiljanić, *Member, IEEE*

***Abstract*— The sequential greedy scheduling (SGS) is a scalable maximal matching algorithm that provides non-blocking in an Internet router with input buffers and a cross-bar. In this paper, we will present the FPGA design of the SGS scheduler. We have optimized scheduler components, and we will prove their correct functioning. The scheduler optimization significantly reduces the worst-case packet delay through the router.**

***Index Terms*— Packet switches, scheduling, high-performance, FPGA, circuit optimization.**

I. Introduction

A router with input buffers is the most scalable single-hop architecture [1], [2]. It is non-blocking if appropriate scheduling algorithms are implemented. But, the algorithm must not limit the router scalability either. Scalability of the scheduling algorithm may be estimated, but it is sure only when the algorithm is actually implemented. In this paper, we present the implementation of the sequential greedy scheduling (SGS) algorithm that provides non-blocking in routers with input buffers, and analyze the implementation performance in terms of its speed and scalability.

Maximal matching algorithms provide non-blocking through a cross-bar with the speedup of two [3]-[5]. Consequently, they can provide delay guarantees to sensitive applications, as well as flexible admission control. Sequential greedy scheduling (SGS) [3],[4] is a scalable maximal matching algorithm. In this algorithm, each input port chooses the first available output for which it has packets to send, and forwards the information about the remaining available outputs to the next input port in the chain. It can be implemented using the pipeline technique. In scalable routers packets are typically split into cells which are independently transferred through the switching fabric. When the traffic is policed, it was shown that the pipelined SGS algorithm guarantees the packet delay of $F \cdot T_c$, where F is the size of the policing interval, and T_c the cell duration [4].

Field programmable gate arrays (FPGAs) are convenient for implementation of the SGS algorithm. They are cost-effective, and relatively easy to program. Testing of the FPGA design is significantly simplified compared to the application-specific integrated circuit (ASIC) design, which speeds up the FPGA design and reduces its cost. Thus, we decided to implement the SGS algorithm in FPGA devices. Speed and scalability of the implementation depend on software tools used for synthesis, and placement and routing [6],[7]. Since the cell duration, T_c, should be greater than the output selection time, T_s, in the pipelined SGS algorithm, and the packet delay is proportional to the cell duration, the output selection time should be as low as possible in order to provide low packet delays.

In this paper, we present a design of the scheduler for the non-blocking router based on the SGS algorithm. Next, we propose the optimizations of scheduler components, and prove their correct functioning. Finally, we analyze the performance of the scheduler design in terms of its scalability and speed, and evaluate the optimization gain. We show that the optimized scheduler provides significantly lower output selection times, and so incurs lower packet delay through the router.

II. Design of the Scheduler for the Non-Blocking Router

The router consists of N ports, the cross-bar fabric, and the scheduler. Each port comprises network processor and data memory. The scheduler consists of N control modules, which correspond to the ports. Network processors split packets into cells of fixed length, and send the information about destination output ports of the incoming packets to the scheduler.

The scheduler control module consists of: queue manager, linked list memory, output selector, coder, and output memory. The linked list memory stores virtual queue linked lists (VQLs) and empty queue linked list (EQL). VQL of some output comprises the memory addresses of cells bound for that output, while EQL comprises the memory addresses of empty locations. The queue manager stores pointers to VQLs and EQL, and performs pointer and linked list memory updates whenever a cell arrives, is scheduled by the output selector, or departs the router. The output selector chooses the first available output from the set of outputs for which the given input has cells to send. The result of the scheduling process is a vector in which only one bit, which corresponds to the scheduled queue, is set to logical one. The coder determines the position of a logical one in that vector, and thus the output that has been scheduled for the observed time slot. The scheduled output is forwarded to the queue manager, and also stored in the output memory until the time the cell should be read.

III. Optimization of the Scheduler Design

The structure of the output selector is shown in Fig. 1. D bits contain information about cells of a particular input that participate in the contention process. D_j bit is set to '1' only if there are unscheduled cells for the j-th output port, and the j-th output port was not selected by previous input ports. As a result of the scheduling process, Q_j is set to '1' when the j-th

Manuscript received October 10, 2006. The associate editor coordinating the review of this letter and approving it for publication was Prof. Dr. Samuel Pierre. This work was supported by the Ministry of Science and Enviromental Protection of the Republic of Serbia, and company Pupin Telecom DKTS.

The authors are with the School of Electrical Engineering, Belgrade University, Bulevar Kralja Aleksandra 73, 11000 Belgrade, Serbia. A. Smiljanić is also with Stony Brook University, USA (e-mail: {pmilos, aleks}@ieee.org).

Digital Object Identifier 10.1109/LCOMM.2007.061653.

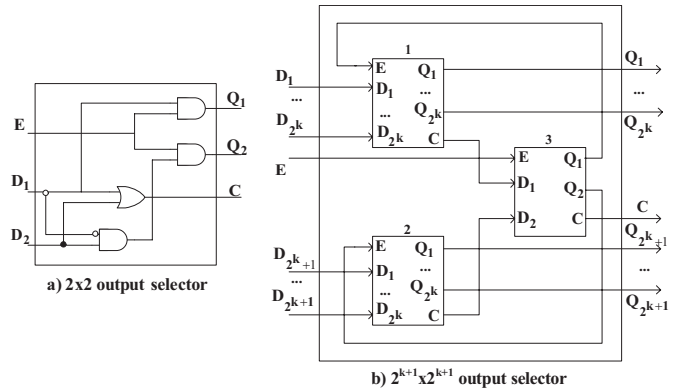

Fig. 1. The output selector structure.

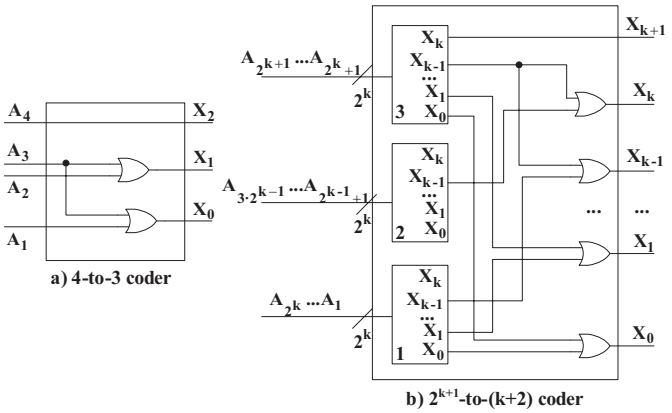

Fig. 2. The coder structure.

output port is chosen by the given input. E bit is the enable signal for the output selector, and C bit is the carry signal.

The two-port output selector is shown in Fig. 1a. The output selector for a router with 2^{k+1} output ports can be built recursively by using two output selectors for 2^k ports and one two-port output selector, as shown in Fig. 1b. A set of 2^{k+1} D bits is divided into two subsets of 2^k bits, and each subset is forwarded to a smaller output selector for 2^k output ports. If any cell is scheduled in one of the subsets, then the carry bit in the corresponding structure is set to '1'. The basic output selector then determines which 2^k-port structure will be selected and sets the corresponding Q bit to '1', which serves as an enable signal for that structure.

Theorem 1: For the $2^k \times 2^k$ output selector the following equations hold:

$$C = \sum_{m=1}^{2^k} D_m, \tag{1}$$

$$Q_i = E \cdot D_i \cdot \prod_{j=1}^{i-1} \overline{D_j}, 1 \le i \le 2^k \tag{2}$$

that is, the output selector chooses the first available output for which an associated input control module has cells to send.

Proof: We will prove the theorem using mathematical induction. From Fig. 1a, it is easy to observe that equations (1), and (2) hold when $k = 1$. Next, let us assume that the the theorem holds for a $2^k \times 2^k$ output selector. We will show that it then holds for a $2^{k+1} \times 2^{k+1}$ output selector.

Let us denote the outputs of the first, the second, and the third output selector with superscripts $^{(1)}$, $^{(2)}$, and $^{(3)}$, respectively. For a $2^{k+1} \times 2^{k+1}$ output selector, from Fig. 1, and inductive assumption (1) it follows that:

$$C = C^{(3)} = C^{(1)} + C^{(2)} = \sum_{m=1}^{2^{k+1}} D_m. \tag{3}$$

Q_i bits are from Fig. 1, and equation (2) equal to:

$$Q_i = \begin{cases} Q_i^{(1)} = EC^{(1)} D_i \prod_{j=1}^{i-1} \overline{D_j}, \ 1 \le i \le 2^k \\ Q_{i-2^k}^{(2)} = E\overline{C^{(1)}} C^{(2)} D_i \prod_{j=2^k+1}^{i-1} \overline{D_j}, \ 2^k < i \le 2^{k+1} \end{cases} \tag{4}$$

It is straightforward to show that from inductive assumption (1), and equation (4) it follows that:

$$Q_i = E \cdot D_i \cdot \prod_{j=1}^{i-1} \overline{D_j}, \ 1 \le i \le 2^{k+1}. \tag{5}$$

From (3), and (5) follows the claim of theorem 1. ■

The output Q bits of the output selector are forwarded to the coder, which structure is given in Fig. 2. When Q_j ='1', then A_j ='1', and the coder returns j coded binary.

The larger structure of a 2^{k+1}-to-$(k+2)$ coder is built recursively by using three 2^k-to-$(k+1)$ coders, as shown in Fig. 2b. First 2^k bits are connected to the first coder, and second 2^k bits are connected to the third coder. If the selected output is among the first $2^k - 1$ outputs the resulting coder has the same outputs as the first coder. Else, if the selected output is among the last $2^{k-1}+1$ outputs, the resulting coder has the same outputs as the third coder plus 2^k. Finally, the second coder determines when the selected output is in between 2^k and $3 \cdot 2^{k-1} - 1$, and the resulting coder outputs are equal to the outputs of the third coder plus 2^k, as well.

Theorem 2: For the 2^k-to-$(k+1)$ coder, when A_x ='1' ($1 \le x \le 2^k$), the outputs of the coder X_i ($0 \le i \le k$) comprise the binary representation of x:

$$x = \sum_{i=0}^{k} X_i \cdot 2^i, \tag{6}$$

and when all the inputs are equal to zero, the coder's output is zero, as well.

Proof: We will again use mathematical induction. From Fig. 2a, it is easy to verify that the claim of theorem 2 holds for $k = 2$. We assume that this claim holds for a 2^k-to-$(k+1)$ coder, and prove the theorem for a 2^{k+1}-to-$(k+2)$ coder.

Let us denote the outputs of the first, the second, and the third coder in Fig. 2b with superscripts $^{(1)}$, $^{(2)}$, and $^{(3)}$, respectively. From inductive assumption (6), and Fig. 2b it follows that if all the inputs of the 2^{k+1}-to-$(k+2)$ coder are equal to zero, so are its outputs. Otherwise, when A_x ='1', it follows that:

$$x^{(1)} = x \cdot \mathbf{1}\{x \in [1, 2^k]\}, \tag{7}$$

$$x^{(2)} = (x - 2^{k-1}) \cdot \mathbf{1}\{x \in [2^{k-1}+1, 3 \cdot 2^{k-1}]\}, \tag{8}$$

$$x^{(3)} = (x - 2^k) \cdot \mathbf{1}\{x \in [2^k + 1, 2^{k+1}]\}, \tag{9}$$

where $\mathbf{1}\{S\}$ is 1 if statement S is true, and 0 otherwise. Inductive assumption (6) is equivalent to the following claims:

$$X_k = \mathbf{1}\{x = 2^k\}, \tag{10}$$

$$X_{k-1} = \mathbf{1}\{x \in [2^{k-1}, 2^k)\}, \tag{11}$$

$$x \ne 2^k \Rightarrow x - X_{k-1} \cdot 2^{k-1} = \sum_{i=0}^{k-2} X_i \cdot 2^i. \tag{12}$$

So, by proving that these claims hold for the 2^{k+1}-to-$(k+2)$ coder, we will prove the theorem. First, we prove (10) when k is replaced by $k+1$:

$$\mathbf{1}\{x = 2^{k+1}\} \overset{(9)}{=} \mathbf{1}\{x^{(3)} = 2^k\} \overset{(10)}{=} X_k^{(3)} \overset{\text{Fig. 2b}}{=} X_{k+1}. \tag{13}$$

```
FOR int IN 1 TO NO_OF_PORTS LOOP
    IF (D(int)='1') THEN
        X<=CONV_STD_LOGIC_VECTOR(int, NO_OF_PORTBITS);
        EXIT;
    ELSIF (int = NO_OF_PORTS) THEN
        X<=CONV_STD_LOGIC_VECTOR(0, NO_OF_PORTBITS);
    END IF;
END LOOP;
```

Fig. 3. The VHDL code for the output selector and coder.

Next, we prove (11) when k is replaced by $k+1$:

$$\begin{aligned}
&\mathbf{1}\{x \in [2^k, 2^{k+1})\} = \\
&= \mathbf{1}\{x \in [2^k, 3\cdot 2^{k-1})\} \vee \mathbf{1}\{x \in [3\cdot 2^{k-1}, 2^{k+1})\} \\
&\overset{(8),\,(9)}{=} \mathbf{1}\{x^{(2)} \in [2^{k-1}, 2^k)\} \vee \mathbf{1}\{x^{(3)} \in [2^{k-1}, 2^k)\} \\
&\overset{(11)}{=} X_{k-1}^{(2)} \vee X_{k-1}^{(3)} \overset{\text{Fig. 2b}}{=} X_k. \qquad (14)
\end{aligned}$$

Now, we prove (12) when k is replaced by $k+1$. From (6), (7), and (9) it follows that $X_i^{(1)} \vee X_i^{(3)}$ is equal to $X_i^{(1)} + X_i^{(3)}$, for all i, so if $x \neq 2^{k+1}$:

$$\begin{aligned}
&\textstyle\sum_{i=0}^{k-1} X_i \cdot 2^i \overset{\text{Fig. 2b}}{=} \sum_{i=0}^{k-1} (X_i^{(1)} + X_i^{(3)}) \cdot 2^i = \\
&\overset{(7),(9),(12)}{=} x \cdot \mathbf{1}\{x \in [1, 2^k)\} + (x - 2^k) \cdot \mathbf{1}\{x \in (2^k, 2^{k+1})\} \\
&\overset{(14)}{=} x \cdot \mathbf{1}\{x \neq 2^k\} - X_k \cdot 2^k + 2^k \cdot \mathbf{1}\{x = 2^k\} \\
&= x - X_k \cdot 2^k. \qquad (15)
\end{aligned}$$

Finally, from (13)-(15) follows the claim of theorem 2. ■

IV. The Performance Analysis

We have implemented both the optimized design described in the previous section, and the VHDL design on low-cost high-performance Altera Cyclone FPGA device of the highest capacity [8]. Both designs were verified by simulation, and successfully tested for different router sizes, when the traffic corresponds to Bernoulli random process, using the testing software that we developed [7]. The VHDL design of the output selector and coder is given by a loop shown in Fig. 3. The loop pointer (integer int) goes through the set of D bits, and it is checked if there are cells that can be scheduled. If such cell is found, then the loop breaks, and the binary coded number of the scheduled output is returned. Otherwise, zero is returned (i.e. no cells are scheduled), and the loop ends.

Performance of the implementations can be measured in terms of minimum output selection time (T_s), and maximum number of control modules that can fit a single chip (N_P). Table I corresponds to the case when the optimized design is used, and table II to the case when the VHDL design is used. In both cases, the pointers to VQLs are stored in memory blocks of the FPGA device. Also, the state machine of the queue manager is optimized, while the pins' speed is doubled in order to increase the scalability [6]. It can be seen from tables I and II that the router with 256 ports can be controlled by the designed scheduler modules. If we assume the router with 10Gb/s ports, the scheduler modules designed on one low-cost Altera FPGA may control a router with hundreds of ports, i.e. with terabit capacity, which confirms anticipated scalability of the SGS algorithm. The maximum number of input modules is limited by either the number of

TABLE I

Resource utilization and timing characteristics, $F = 16N$

N	N_P	LE	Mem [Kbits]	$pins$	lim	T_s [ns]
32	12	7912	77.9	228	Mem	52.2
64	9	8699	130.4	221	Mem	53.5
128	6	9375	191.9	252	Mem	54.9
256	1	2967	69.8	281	Pins	60.5

TABLE II

Resource utilization and timing characteristics, $F = 16N$

N	N_P	LE	Mem [Kbits]	$pins$	lim	T_s [ns]
32	12	7719	77.9	228	Mem	52.6
64	9	8381	130.4	221	Mem	64.5
128	6	8789	191.9	252	Mem	79.8
256	1	2660	69.8	281	Pins	90

M4K memory blocks, or the number of pins, and these do not depend on the design option. However, the minimum output selection time differs significantly. The optimized scheduler design outperforms the VHDL design in all the considered cases. The gain of the optimization increases with the increase in router size. It rises from 0.7% for $N = 32$ to 48.7% for $N = 256$. Minimum output selection times below 60.5ns are obtained for the optimized scheduler, so low packet delays can be guaranteed.

V. Conclusion

In this paper we have presented a scalable design of the scheduler based on the SGS algorithm. We have optimized the scheduler components, so as to obtain low output selection times. It was proven that the proposed structures of the output selector and the coder function correctly, and that they significantly outperform the corresponding VHDL design. The gain of the optimized structures is up to 48.7% for the observed cases, so the total packet delay is significantly reduced. This gain is expected to further increase with the router size. Thus, the optimized scheduler will improve performance of high-capacity non-blocking packet routers that ought to provide rate and delay guarantees even to the most sensitive applications.

References

[1] N. McKeown *et al.*, "The Tiny Tera: a packet switch core," *IEEE Micro*, vol. 17, no. 1, pp. 26-33, Jan.-Feb. 1997.

[2] H. J. Chao, "Saturn: a terabit packet switch using dual round-robin," *IEEE Commun. Mag.,* vol. 38, no. 12, pp. 78-84, Dec. 2000.

[3] A. Smiljanić, "Flexible bandwidth allocation in terabit packet switches," in *Proc. IEEE HPSR 2000,* June 2000, pp. 233-241.

[4] A. Smiljanić, "Flexible bandwidth allocation in high-capacity packet switches," *IEEE/ACM Trans. Networking,* vol. 10, no. 2, pp. 287-293, April 2002.

[5] G. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," in *Proc. IEEE INFOCOM 2000,* pp. 556-564.

[6] M. Petrović and A. Smiljanić, "Design of the scheduler for the high-capacity non-blocking packet switch," in *Proc. IEEE HPSR 2006,* pp. 397-402.

[7] M. Petrović, M. Blagojević, V. Joković, and A. Smiljanić, "Design, implementation, and testing of the controller for the terabit packet switch," in *Proc. IEEE ICCCAS 2006,* vol. 3, pp. 1701-1705.

[8] P. Leventis, *et al.*, "Cyclone^{TM}: a low-cost, high-performance FPGA," in *Proc. IEEE CICC 2003,* pp. 49-52.