Hardware Scheduling in High-Speed, High-Capacity IP Routers

S. Q. Zheng †, Mei Yang †, and Francesco Masetti‡

† Department of Computer Science
University of Texas at Dallas, Richardson, TX 75083-0688, USA
{sizheng, meiyang}@utdallas.edu
‡ Research & Innovation, Alcatel USA, Plano, Texas 75075, USA
Francesco.Masetti@alcatel.com

Abstract

The key to the design of high speed, high capacity IP switches/routers with multiple servers in an input/output module is a fast scheduling scheme resolving input and output contentions. Such a scheduling scheme is a typical application of the multi-requester, multi-server (MRMS) problem. To efficiently solve the MRMS problem and provide fair services to all requesters, we introduce four programmable k-selectors designs in this paper. Simulations on Altera's CPLD (FPGA) demonstrate that our designs achieve significant performance improvement over the design using programmable priority encoders. Programmable k-selectors are very useful to construct hardware Request-Grant or Request-Grant-Accept schedulers for high-speed, high-capacity multi-server switches/routers.

Keywords
MRMS problem, Multi-server switch architecture, Programmable k-selector, Programmable prefix sums

1 Introduction

The exponential growth of Internet multimedia traffic demands high-speed, high-capacity IP switches/routers. In general, an IP switch/router consists of a number of input/output (I/O) modules that are interconnected by a switching matrix. Typical tasks assigned to an I/O module include IP packet buffering, routing table lookup, IP packet segmentation, packet filtering, queue management, etc. As these tasks being carried out by hardware, IP packet switching becomes the bottleneck of router performance. There are two major challenges in the design of high speed, high capacity IP switches/routers. (1) How to build a large capacity switching matrix? (2) How to design a fast scheduling scheme that resolves output contention and schedules packet transmission between I/O modules within stringent time constraint while achieving high switching throughput?

Combining the strength of both optical and electronic technologies, an opto-electronic cell-based IP switch/router architecture with multiple servers [1, 2] is proposed to be a competitive solution for (1). In this switch architecture, variable-length IP packets are segmented into fix-sized cells as they arrive, transferred across the switching matrix (SM), and reassembled again into IP packets before they depart. Each input port maintains N virtual output queues (VOQs) [3], each being associated with a destination output port. There are multiple connections, using space division multiplexing or wavelength division multiplexing, between a port and the SM. One or more connections are abstracted as a server. These servers may provide necessary switch speedup for desired performance. With speedup of these servers, combined input/output queuing (CIOQ) is used to take both the advantage of output queuing that supports fair bandwidth sharing and provides delay bounds for regulated traffic, and the advantage of input queuing that is more scalable and easy to implement. Figure 1 demonstrates the multi-server switch architecture for an $N \times N$ switch with k servers on each input/output port, where $k \leq N$. During each switch slot, there may be up to N requests to k servers at each input and output port.

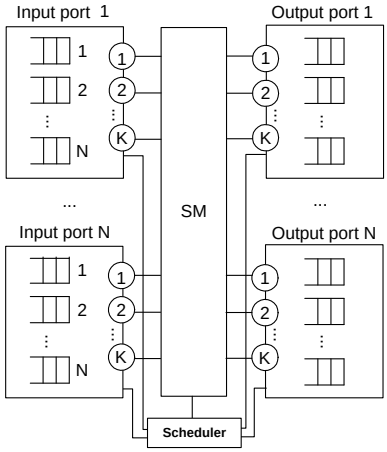

Figure 1. Block diagram of a multi-server switch architecture.

The key to the design of the multi-server switch architecture is an efficient and fast switch scheduling algorithm built in hardware to resolve input and output port contentions. Representing each port by a node and each I/O request by an edge, we obtain a bipartite graph. The switch scheduling problem is thus reduced to finding a *maximum k-matching* in this bipartite graph. A k-matching is a subgraph in which each node's degree is no more than k. A maximum k-matching is a k-matching with maximum number of edges. The maximum k-matching problem is a special case of b-matching problem [4]. All known sequential and parallel algorithms for the maximum k-matching problem are too complex to implement. As many known maximal-matching algorithms for input-buffered switches, such as PIM [5], iSLIP [6], DDR [7] and their variations, several approximation algorithms have been proposed [1, 2, 8].

Such an algorithm may take multiple iterations to stop. Each iteration consists of either two steps, Request and Grant (RG), or

three steps, Request, Grant and Accept (RGA). For example, the three steps of a RGA algorithm are as follows.

***Request*:** Any input port with unmatched servers send requests to every output port for which it has a request.

***Grant*:** An output port with unmatched servers makes up to the number of available servers grants, starting from the highest priority element and sends them back to their corresponding input ports.

***Accept*:** An input port with unmatched servers accepts up to the number of available servers grants, starting from the highest priority element.

Figure 2 gives a high-level block diagram of such a scheduler implemented in hardware. The *Grant* step is implemented by the grant arbitration component, as the *Accept* step by the accept arbitration component. Clearly, the delay through the grant arbitration component and the accept arbitration component directly affects the speed of the scheduling algorithm.

The function of an arbitration component in a multi-server switch is one of many possible applications of the multi-requester, multi-server (MRMS) problem, which is defined as follows. There are N requesters and k servers, where $k \leq N$. Given N binary input requests, R_i's, where $0 \leq i \leq N-1$, with $R_i = 1$ representing requester i having a request, select $\min\{k, \sum_{i=0}^{N-1} R_i\}$ requesters such that $R_i = 1$. All servers are functionally equivalent. At any time, at most one requester can be served by a server. An implementation of this k-selection function is a *k-selector*. The 1-selector, an arbiter, is commonly used in constructing schedulers for input-buffered switches. Several designs of high-speed 1-selector (e.g. [9, 10, 11]) have been reported.

To achieve fairness, some "intelligence" should be built into a k-selector. A *programmable k-selector* is one whose starting selection point can be dynamically changed to ensure fairness. A programmable k-selector built with an integrated circuit (IC), such as FPGA or ASIC, is extremely attractive due to its high speed and its easy integration into the interface between requesters and servers. One possible design of a programmable k-selector is employing an arbiter, such as the programmable priority encoder (PPE) proposed in [10]. Since the PPE can only make one selection each time, we have to run PPE up to k times to make all grants. Such an implementation may not be fast enough to satisfy the requirements of some real-time MRMS applications, such as scheduling in the multi-server switch architecture. Thus, a more efficient hardware solution is needed.

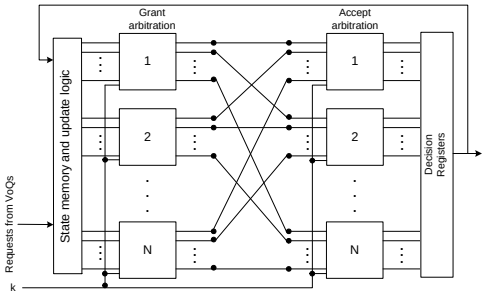

Figure 2. Block diagram of a scheduler for an $N \times N$ multi-server switch.

In this paper, we propose several efficient hardware designs of programmable k-selectors. To the best of our knowledge, our programmable k-selector designs are the first hardware designs that could make k selections out of n requesters. The programmable k-selectors have been used in constructing multi-server switch schedulers based on RG/RGA scheduling algorithms, such as the kDRR scheduling algorithm proposed in [8]. Due to the fact that the core of our designs is an $O(\log N)$-level combinational circuit, our programmable k-selectors are extremely useful for the switch control/scheduling in high-speed, high-capacity IP routers [12].

In the following of the paper, we first reduce the programmable k-selection function to a programmable prefix sums operation and propose three different programmable prefix sums circuit designs based on a simple parallel prefix sums circuit. We further present four next reference point generation circuit designs, each one is associated with a different programmable k-selector design. With simulations on Altera's CPLD (FPGA) implementations of programmable k-selectors, we show that our designs achieve significant speed improvement over the PPE design with acceptable additional cost in terms of the number of logical cells.

2 Programmable k-Selectors

The function of a *programmable k-selector* is defined as follows. Given N binary requests R_i's, where $0 \leq i \leq N-1$, and two integers x and k such that $0 \leq x \leq N-1$, $1 \leq k \leq N$, select first $\min\{k, \sum_{i=0}^{N-1} R_i\}$ input such that $R_i = 1$ starting from position x in a circular manner. If R_i is selected, then the output grant signal $G_i = 1$; if $R_i = 0$, or $R_i = 1$ but it is not selected (in such a case, $\sum_{i=0}^{N-1} R_i > k$), then $G_i = 0$. This k-selector is considered programmable because of the parameter x, which is specified dynamically each time the device is invoked, is used to determine the starting selection position. This parameter is useful for history sensitive applications.

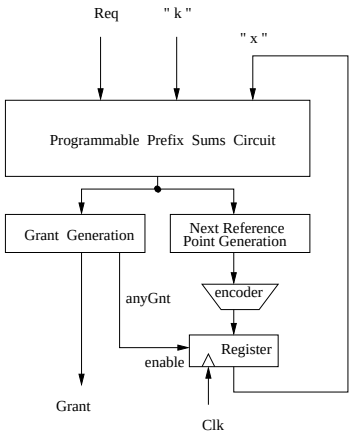

Figure 3. The block diagram of a programmable k-selector.

The programmable k-selector function can be reduced to the following programmable prefix sums operation. Given N binary requests $\mathbf{R} = (R_0, R_1, \cdots, R_{N-1})$, and an integer x such that $0 \leq x \leq N-1$, compute the prefix sums $Sum_i = R_x + R_{(x+1) \bmod N} + \cdots + R_i$ for $0 \leq i \leq N-1$. After performing this prefix sums operation, request R_i is granted if and only if $R_i = 1$ and $Sum_i \leq k$. For example, consider the case that $N = 8$, $k = 3$, $x = 2$, and $\mathbf{R} = (0, 1, 0, 1, 1, 0, 1, 1)$. Thus, $(Sum_0, Sum_1, \cdots, Sum_7) = (4, 5, 0, 1, 2, 2, 3, 4)$, and the selected requests are R_3, R_4 and R_5.

The block diagram of a generic programmable k-selector is shown in Figure 3. It is the combination of a programmable prefix sums circuit, a grant generation circuit, and a next reference point generation circuit. In the following, we first introduce our programmable prefix sums circuits.

3 Programmable Prefix Sums Circuit Designs

3.1 A Parallel Prefix Sums Circuit

We first study a special case with starting point $x = 0$. This prefix sums operation is defined as: given a sequence of integers $\mathbf{R} = (R_0, R_1, \cdots, R_{N-1})$, compute the prefix sums $Sum_i = R_0 + R_1 + \cdots + R_i$ for $0 \leq i \leq N-1$.

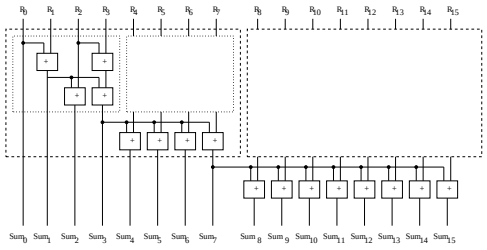

Figure 4. A parallel prefix sums circuit using full adders.

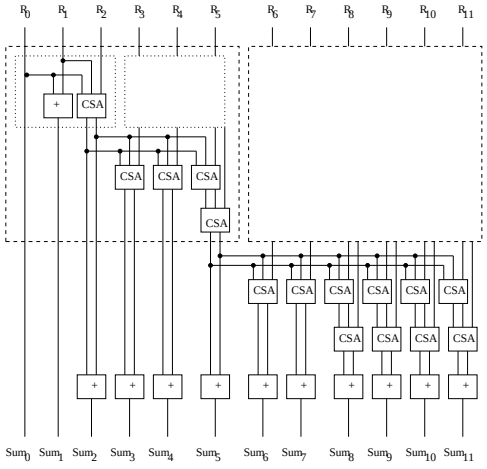

Figure 5. A parallel prefix sums circuit using carry save adders and full adders.

We need to use adders to implement parallel prefix sums operation. Figure 4 displays the parallel prefix sum circuit for input size of $N = 16$ using full adders. Dotted blocks are used to show the recursive construction schemes of this circuit. Due to ripple carries, such a design may not be the fastest one. Basically, there are two speedup methods. One is to use carry-lookahead technique, and the other is to use carry save adders to form a well-known Wallace tree, as shown in Figure 5. Using these techniques, the time complexity of this parallel prefix sums circuit is $O(\log N)$ gates delay. We name this special-case prefix sums circuit (and its improved variations) SIMPLE_PS.

Consider N requests $\mathbf{R} = (R_0, R_1, \cdots, R_{N-1})$ as an ordered circular queue. We define the prefix sums problem with reference point of position x as follows: Given a sequence of integers $\mathbf{R} = (R_0, R_1, \cdots, R_{N-1})$ and an integer variable $0 \leq x \leq N-1$, compute the prefix sums $Sum_i = R_x + R_{(x+1) \bmod N} + \cdots + R_i$ for $0 \leq i \leq N-1$. Here, x is the *reference point* of the computation. Since x is a variable, this generalized problem is called *programmable prefix sums* (PPS) computation. In the following, we will present our three programmable prefix sums circuit designs.

3.2 Design SHIFT_PPS

Conceptually, this is the simplest design. The block diagram of this design is shown in Figure 6. One SIMPLE_PS and two barrel shifters are used. Before prefix sums operation, R_i's are circular shifted x positions to the left. After prefix sums are computed, the sums are circular shifted x positions to the right. There is another output $Sum = \sum_{i=0}^{N-1} R_i$, which will be used in ROUNDROBIN_SELECT design.

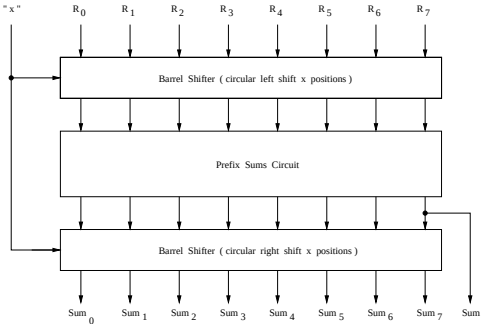

Figure 6. A programmable prefix sums circuit using barrel shifters.

A N-bit barrel shifter can be implemented by $\log_2 N$ stages of N 2 – 1 multiplexers with $O(\log_2 N)$ gates delay [13]. It should be noticed that the second barrel shifter is more expensive because each Sum_i contains $\log_2 N$ bits. When used in a k-selector, we can use the comparators to obtain G_i's, as in Figure 10, before the grant signals are circular shifted right. This will significantly save circuitry of the second shifter.

3.3 Design DOUBLE_PPS

This design employs two copies of SIMPLE_PS circuit. One is used for computing prefix sums of the bits preceding the x-th position, and the other is for calculating prefix sums of the remaining bits in the circular fashion. The two parts of prefix sums are merged into the final prefix sums. We use an N-bit boolean vector $\mathbf{T}$ to separate the two parts.

$\mathbf{T}$ is obtained by a thermometer encoding of a $\log_2 N$-bit-wide vector $\mathbf{x}$ with the following transformation equation:

$$T[i] = 0 \text{ if and only if } i < value(\mathbf{x}) \text{ for } 0 \leq i \leq N-1$$

Table 1 shows the truth table and logic equations for the thermometer logic of $N = 8$. Given, a $\log_2 N$-bit $\mathbf{x}$, we use a thermometer encoding circuit to generate $T_0 T_1 \cdots T_{N-1}$, which has $\log_2 N$ gates delay. Figure 7 shows the thermometer encoder for $N = 8$. For example, if x = value($\mathbf{x}$) = 3, then $T_0 T_1 \cdots T_7 =$ 00011111.

We then use $\mathbf{T}$ to separate the request inputs into two parts at position x. The first part has $(R_0, R_1, \cdots, R_{x-1}, 0, \cdots, 0)$,

x(2..0)	T(7..0)	Logic equations
000	11111111	$T_7 = 1$
001	11111110	$T_6 = \overline{x_2 \cdot x_1 \cdot x_0}$
010	11111100	$T_5 = \overline{x_2 \cdot x_1}$
011	11111000	$T_4 = \overline{x_2 \cdot (x_1 + x_0)}$
100	11110000	$T_3 = \overline{x_2}$
101	11100000	$T_2 = \overline{x_2 + x_1 \cdot x_0}$
110	11000000	$T_1 = \overline{x_2 + x_1}$
111	10000000	$T_0 = \overline{x_2 + x_1 + x_0}$

Table 1. Truth table and logic equations for the thermometer logic of $N = 8$.

which is extracted out by AND of $\mathbf{R}$ and $\overline{\mathbf{T}}$. The second part has $(0, \cdots, 0, R_x, \cdots, R_{N-1})$, which is the result of AND of $\mathbf{R}$ and $\mathbf{T}$. On each part, a SIMPLE_PS performs prefix sums computations concurrently. Then the prefix sums of the second part are added to the prefix sums of the first part. The final prefix sums are obtained by merging the prefix sums of two parts. The logic of DOUBLE_PPS for $N = 4$ is shown in Figure 8.

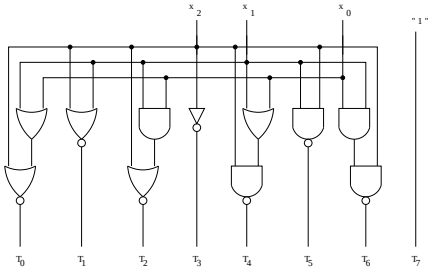

Figure 7. A thermometer encoder.

3.4 Design CONVERT_PPS

Compared with Design SHIFT_PPS, Design DOUBLE_PPS does not use barrel shifters, whereby reducing delay caused by barrel shifters. But an additional SIMPLE_PS circuit is needed. Can we save some circuitry by removing the second SIMPLE_PS circuit? We present another design, named Design PS_CONVERT, which uses one SIMPLE_PS but does not require barrel shifters.

Let Sum'_i's be the prefix sums with reference point of location 0. We observe that for the programmable prefix sums Sum_i's, $0 \leq i \leq N-1$, with reference point of location x can be obtained as follows.

- Case 1: If $T_0 = 1$, then $T_i = 1$ and $Sum_i = Sum'_i$ for $0 \leq i \leq N-1$.
- Case 2: If $T_0 = 0$, then $T_i = 0$ for $0 \leq i \leq x-1$ and $T_i = 1$ for $x \leq i \leq N-1$. Thus, $Sum_i = Sum'_i - Sum'_{x-1}$ for $x \leq i \leq N-1$, and $Sum_i = Sum'_i + Sum_{N-1}$ for $0 \leq i \leq x-1$.

Based on this observation, we construct a circuit named PS_CONVERT, as shown in Figure 9. Design CONVERT_PPS is composed of three combinational circuits: an N-bit SIMPLE_PS, an N-bit thermometer encoder, and an N-bit PS_CONVERT.

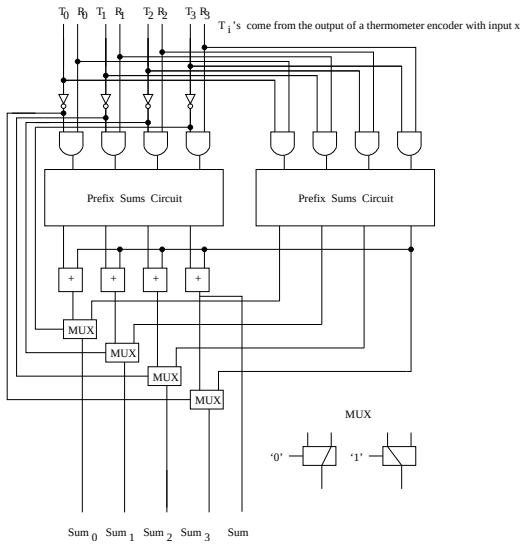

Figure 8. The Double_PPS circuit for $N = 4$.

4 Programmable k-Selector Designs

In this section, we will focus on designs of the other two major parts of a programmable k-selector, the grant generation circuit and the next reference point generation circuit.

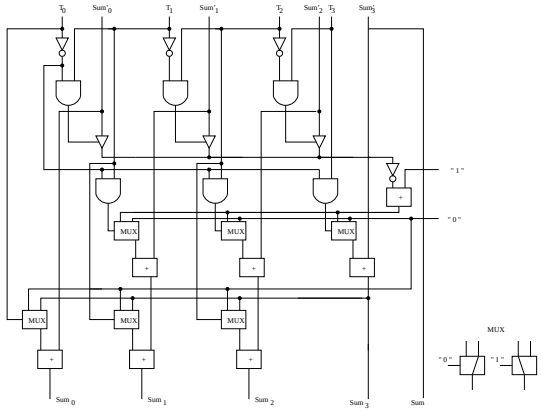

Figure 9. The PS_CONVERT circuit for $N = 4$.

4.1 Grant Generation Circuit

The grant generation signals are generated as follows: Grant signal $G_i = 1$ if and only if $R_i = 1$ and $Sum_i \leq k$. Figure 10 shows the grant generation circuit for $N = 8$. The signal $anyGnt$ in Figure 3 is obtained by logical OR operation on all G_i's.

4.2 Next Reference Point Generation Circuits

A programmable k-selector is used to schedule serving the requests in iterations. In each iteration, $\min\{k, \sum_{i=0}^{N-1} R_i\}$ requests are selected. A new reference point (NRP) is determined before a new

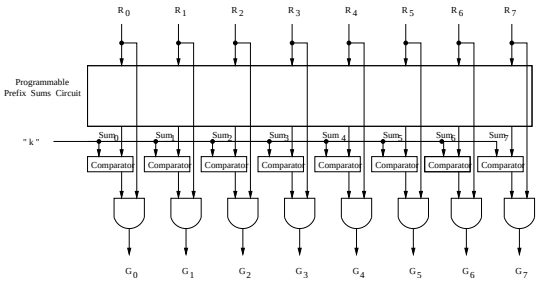

Figure 10. Grant generation circuit.

iteration. An important issue in determining NRP is to avoid request starvation and ensure fairness to all requests. With the programmable prefix sums circuit and the grant generation circuit in place, various programmable k-selectors can be designed by using different NRP generation circuits.

We use $current\_ref\_pt$ and $next\_ref\_pt$ to denote the reference point of the current and next prefix sums operation, respectively. In the following, we will discuss four next reference point generation circuits.

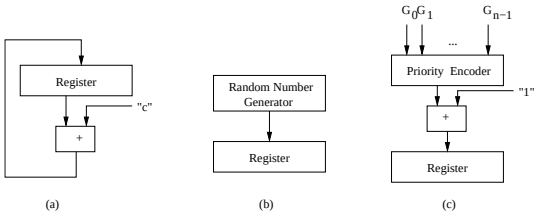

Figure 11. Three NRP generation circuits

- REGULAR_SELECT: Our first programmable k-selector design is based on the following NRP generation method: $next\_ref\_pt = (current\_ref\_pt + c) \bmod N$, where c is a constant. The circuit of REGULAR_SELECT is shown in Figure 11(a). The advantage of this design is its simplicity.
- RANDOM_SELECT: The NRP in this design is generated by a random generator, as shown in Figure 11(b). The advantage of this design is that it is conceptually simple. However, a truly random number is hard to generate, especially using hardware.
- PRIORITY_SELECT: Let the current grant signals be $G_0, G_1, \cdots, G_{N-1}$, and let $j = \max\{i \mid G_i = 1, 0 \leq i \leq N-1\}$. This design generates $next\_ref\_pt = (j+1) \bmod N$ using a priority encoder. For example, for $N = 8$, $k = 3$ and the current grant signals being $(G_0, G_1, G_2, G_3, G_4, G_5, G_6, G_7) = (0, 1, 0, 1, 1, 0, 1, 0)$, $current\_ref\_pt = 4$ and $next\_ref\_pt = 5_{10} = 101_2$. The block diagram of this circuit is shown in Figure 11(c). We omit the construction of a priority encoder since it is a well-known logic component.
- ROUNDROBIN_SELECT: The NRP of the Round Robin scheme is computed as follows. Let the current output of the programmable prefix sums circuit be $Sum_0, Sum_1, \cdots, Sum_{N-1}$, and let $Sum = \sum_{i=0}^{N-1} R_i$, which can be easily generated from programmable prefix sums circuits, as shown in Figures 6, 8, and 9. If $Sum \leq k$, $next\_ref\_pt = j$ such that $Sum_{(j-1) \bmod N} = Sum$ and $R_{(j-1) \bmod N} = 1$; otherwise, $next\_ref\_pt = j$ such that $Sum_{(j-1) \bmod N} = k$ and $R_{(j-1) \bmod N} = 1$. Figure 12 shows the circuit for ROUNDROBIN_SELECT. The output signal $F_j = 1$ if and only if $next\_ref\_pt = j$. This design ensures fairness to all requests.

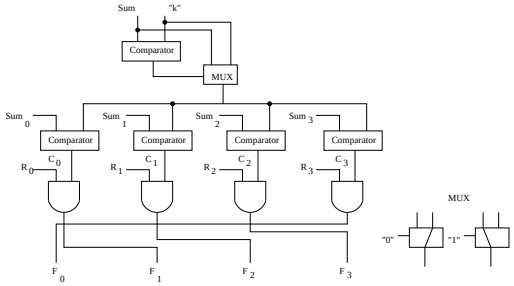

Figure 12. Circuit for generating NRP in design ROUNDROBIN_SELECT.

5 Simulation Results and Comparisons

In this section, we present the simulation results of various designs of programmable k-selectors with Altera Max+plus II using its CPLD (FPGA) family ACEX 1k [14]. Since the next reference point generation circuit is generally simple and can be performed in parallel with the grant generation circuit, we focus on minimizing the delay from requests **R** to grants **G** in our simulations. Table 2 lists the timing results in terms of ns and Table 3 compares the area cost in terms of the number of logic cells (LCs). All these designs are optimized under the same operating conditions and the tool is directed to achieve the fastest implementation of each design.

Any eligible value of k can be used in our k-selector designs, while Design PPE (PPE_only_smpls in [10]) is only for $k = 1$. Each of our designs is much faster than the design of PPE when k is over a certain value. For example, when $N = 64, k = 32$, Design SHIFT_PPS only takes 66.6 ns to make 32 grants while PPE takes $39.9 \times 32 = 1276.8$ ns to pick up 32 grants. The bottom row of Table 2 shows the improvement percentage of SHIFT_PPS over PPE when $k = N/2$. Compared to the timing improvements our designs have achieved, the tradeoff of more cost of our designs is acceptable.

Among our three designs, Design DOUBLE_PPS achieves the best timing results with the most cost of logic cells (LCs). The performances of SHIFT_PPS and CONVERT_PPS are comparable, but CONVERT_PPS has more cost than SHIFT_PPS in terms of the number of LCs. It is important to note that the number of LCs does not imply the chip area used, since interconnections take significant area. Design SHIFT_PPS uses the least number of LCs, but its two shifters consume significant wire routing resources. It is very likely that in ASIC implementations, SHIFT_PPS uses more chip area than the other two designs. Although our simulations depend on the CPLD chip used, these results are instructive. We expect that the performance of our designs to be much better if they are implemented by ASICs.

Design	N=8	N=16	N=32	N=64
PPE ($k = 1$)	13.8	17.5	30.0	39.9
SHIFT_PPS	20.6	33.5	49.0	66.6
DOUBLE_PPS	22.3	29.3	46.5	65.5
CONVERT_PPS	25.3	31.7	42.5	67.0
IMPROVEMENT OF SHIFT_PPS OVER PPE WHEN $k = N/2$	62.7%	76.1%	89.8%	94.8%

Table 2. Timing results for various programmable k-selector designs in terms of *ns*.

Design	N=8	N=16	N=32	N=64
PPE ($k = 1$)	41	137	676	2600
SHIFT_PPS	105	316	918	2371
DOUBLE_PPS	181	531	1678	4882
CONVERT_PPS	164	517	1395	3663

Table 3. Area results for various programmable k-selector designs in terms of number of logic cells (LCs).

6 Concluding Remarks

In this paper, we defined k-selectors for MRMS problems, introduced programmable k-selectors and proposed several programmable k-selector circuit designs. Due to their high performance, programmable k-selectors are extremely useful for switch control/scheduling in high-speed, high capacity IP switches/routers, such as constructing the multi-server switch scheduler [8] and the scheduler for the SDM CIOQ switch, and the switch control for group connectors [15]. Programmable k-selectors are also useful for other real-time MRMS applications, such as the control of an $p \times q$ concentrator [16, 17], and the control of a shared multi-bus system.

One possible extension of this work is to incorporate more "intelligence" into programmable k-selectors. For example, requests can be assigned priorities. A prioritized programmable k-selector can grant requests according to their priorities, favoring the requests with higher priorities.

References

[1] J. Blanton, H. Badt, G. Damm, and P. Golla, Iterative scheduling algorithms for optical packet switches, *ICC 2001 Workshop*, Helsinki, June 2001.

[2] G. Damm, J. Blanton, P. Golla, D. Verchere, and M. Yang, Fast scheduler solutions to the problems of priorities for polarized data traffic, *Proc. of International Symposium on Telecommunications* (IST'01), Tehran, Iran, Sept. 2001.

[3] M. J. Karol, M. G. Hluchyj and S. P. Morgan, Input vs. output queueing on a space-division packet switch, *IEEE Transaction on Communications*, Vol. 35, No. 12, pp. 1347-1356, 1987.

[4] W. J. Cook, W. R. Pulleyblank, A. S., and W. H. Cunningham, *Combinatorial Optimization*, Wiley John & Sons Inc., Nov. 1997.

[5] T. Anderson, S. Owicki, J. Saxe and C. Thacker, High-speed switch scheduling for local-area networks, *ACM Transactions on Computer Systems*, vol. 1, no. 4, pp. 319-352, 1993.

[6] N. McKeown, The iSLIP Scheduling Algorithm for Input-Queued Switches, *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 188-201, 1999.

[7] J. Chao, Saturn: a terabit packet switch using dual round robin, *IEEE Communications Magazine*, vol. 38, no. 12, pp. 78-84, 2000.

[8] M. Yang, S. Q. Zheng and D. Verchere, The kDDR scheduling algorithms for multi-server packet switches, to appear in *Proc. ISCA 15th International Conference on PDCS*, 2002.

[9] H. J. Chao, C. H. Lam, and X. Guo, A fast arbitration scheme for terabit packet switches, *Proc. of Globecom 1999*, pp. 1236-1243, 1999.

[10] P. Gupta, N. McKeown, Designing and implementing a fast crossbar scheduler, *IEEE Microelectronics*, Vol. 19, No. 1, pp. 20-29, 1999.

[11] S. Q. Zheng, M. Yang, J. Blanton, P. Golla and D. Verchere, A parallel round-robin arbiter for switch control, *Proc. of IEEE Midwest Symposium on Circuits and Systems*, Aug. 2002.

[12] M. Yang and S. Q. Zheng, Efficient scheduling for CIOQ switches with space-division multiplexing expansion and grouped inputs/outputs, submitted for publication.

[13] V. P. Heuring, H. F. Jordan, *Computer systems design and architecture*, Addison-Wesley, 1996.

[14] Altera, *ACEX 1K Programmable Logic Device Family Data Sheet*, Sept. 2001.

[15] Y. Yang, S. Q. Zheng, and D. Verchere, Group switching for DWDM networks, submitted for publication.

[16] M. Garey, F. Hwang and G. Richards, Asymptotic results for partial concentrators," *IEEE Transactions on Communications*, vol. 36, no. 2, pp. 214-217, February 1988.

[17] N. Pippenger, Superconcentrators, *SIAM Journal on Computing*, vol. 6, pp. 298-304, 1977.