CONSTRUCTING SCHEDULERS FOR HIGH-SPEED, HIGH-CAPACITY SWITCHES/ROUTERS

S. Q. Zheng †, M. Yang †, and F. Masetti ‡

† Department of Computer Science

Box 830688, MS EC 31, University of Texas at Dallas, Richardson, TX 75083-0688, USA

{sizheng, meiyang}@utdallas.edu

‡ Research & Innovation, Alcatel USA, 3400 Plano Parkway, Plano, TX 75075, USA

Francesco.Masetti@alcatel.com

Abstract

The key to the design of CIOQ switches with space division multiplexing and grouped inputs/outpus (SDMG CIOQ switches for short) 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 programmable k-selectors which can make k grants out of N requests in $O(\log N)$ time. We first show that the function of a programmable k-selector can be reduced to a programmable prefix sums operation. Based on a simple prefix sums circuit, we propose three programmable prefix sums circuit designs. We further propose four different programmable k-selector designs. Simulations on Synopsys's design_analyzer demonstrate that our designs achieve significant performance improvement over the design using programmable priority encoders. Due to their high performance, programmable k-selectors are very useful for constructing schedulers for high-speed, high-capacity switches/routers, such as, SDMG CIOQ switches and multi-server switches.

Keywords: Switch, scheduler, programmable k-selector, programmable prefix sums.

1. Introduction

The exponential growth of Internet 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 to improve the switching capacity? (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?

To achieve high switching capacity, people have proposed combined input and output queueing (CIOQ) switches, which take the advantage of input queueing (IQ) switches and output queueing (OQ) switches [1]. In a CIOQ switch, the switching matrix needs to run k times faster than the line rate (referred as speedup of k) to realize k times switching capacity. However, for CIOQ switches with high speed links, it may not always possible to realize speedup of k.

To remove the speedup requirement of the switching matrix, we have proposed a new CIOQ switch architecture which features in space division multiplexing expansion and grouped inputs/outputs (SDMG CIOQ switch for short) in [2]. To achieve the same switching capacity as of an $N \times N$ CIOQ switch with speedup of k, an SDMG CIOQ switch employs an $Nk \times Nk$ switching matrix. As shown in Figure 1, for an $N \times N$ SDMG CIOQ switch, there are k connections between each input/output port and the switching matrix. To remove head-of-line blocking [3], each input port maintains N virtual output queues (VOQs) [3], each of which is associated with a destination output port.

We assume that an SDMG CIOQ switch is cell based. In such a switch, 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. Time is divided into cell slots and one cell slot equals to the

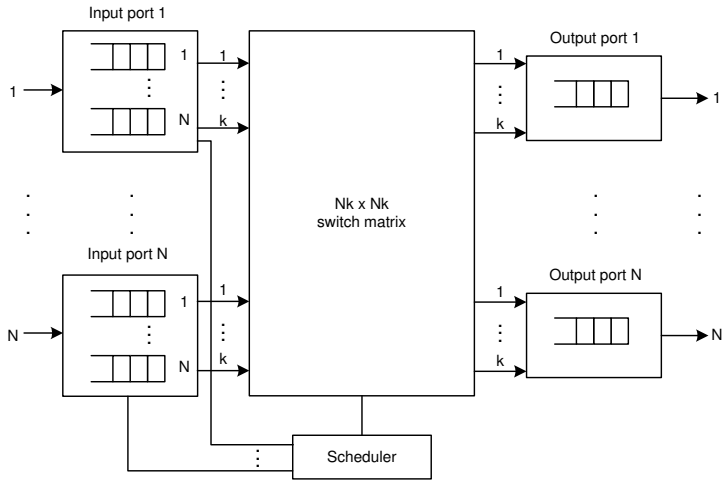

Fig. 1. An SDMG CIOQ switch.

transmission time of a cell. At the start of each cell slot, a scheduling algorithm needs to decide which cells can be transferred from (to) each input (output) since there may be up to N requests to k connections at each input/output port.

The key to the design of the SDMG CIOQ switch architecture is an efficient and fast cell scheduling algorithm to resolve input and output contentions. The cell scheduling problem on the SDMG CIOQ switch can be abstracted as a *maximum k-matching* problem on the bipartite graph composed of nodes of input/output ports [2] and edges of requests from input ports to output ports. Due to the high time complexity of optimal maximum k-matching algorithms, we have proposed an efficient approximation algorithm, kFRR in [2], based on FIRM [4]. As many known scheduling algorithms for input-buffered switches, such as PIM [5], *i*SLIP [6], DDR [7], FIRM [4], kFRR is also an iterative algorithm with each iteration consisting of the following three steps.

Step 1: **Request**. Any input port with available connections sends requests to every output port for which it has a request.

Step 2: **Grant**. An output port with available connections grants up to the number of available connections, starting from the highest priority input, and sends these grants back to their corresponding input ports.

Step 3: **Accept**. An input port with available connections accepts grants up to the number of available

connections, starting from the highest priority output.

For brevity, we omit the details of how kFRR updates the highest priority input/output pointers. Figure 2 shows the high-level block diagram of the kFRR scheduler implemented in hardware. The *Grant* step is implemented by a set of N grant arbitration components, and the *Accept* step is implemented by a set of N accept arbitration components. Each arbitration component is responsible for selecting k out of N requests. Clearly, the delay through a grant arbitration component and an accept arbitration component directly affects the speed of the kFRR scheduling algorithm.

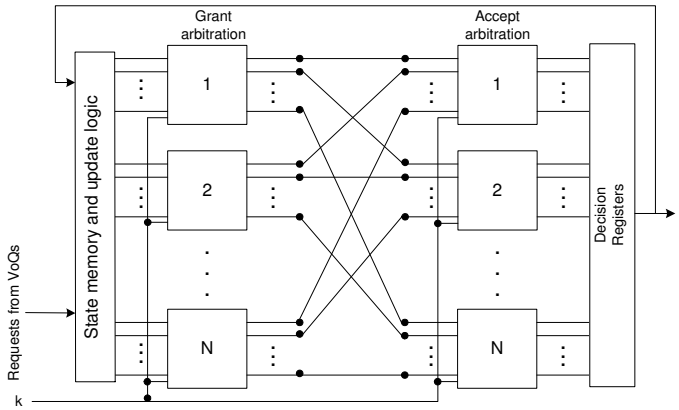

Fig. 2. Block diagram of the kFRR scheduler for an $N \times N$ SDMG CIOQ switch.

The function of an arbitration component used in the kFRR scheduler of an SDMG CIOQ 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 first $\min\{k, \sum_{i=0}^{N-1} R_i\}$ requesters such that $R_i = 1$. All servers are functionally equivalent. At any time, at most one requester can be served by a server. We define the implementation of this k-selection function as 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. [8], [9], [10]) have been reported.

A *programmable k-selector* is one whose selection starting point can be dynamically changed to ensure fairness. One possible design of a programmable k-selector is employing an arbiter, such as the pro-

grammable priority encoder (PPE) proposed in [9]. Since the PPE can only make one grant each time, we have to run PPE up to k times to make k grants. The time complexity of this design is $O(k \log N)$, which is not fast enough to satisfy the requirements of some real-time MRMS applications, such as scheduling on the SDMG CIOQ switch architecture. Thus, a more efficient hardware solution is needed.

In this paper, we propose several hardware designs of programmable k-selectors, all having the time complexity of $O(\log N)$. To the best of our knowledge, our programmable k-selector designs are the first hardware designs that could make k grants out of N requests simultaneously. Due to their high performance, programmable k-selectors are very useful for constructing the kFRR scheduler for SDMG CIOQ switches, implementing schedulers for multi-server switches [11], [12], [13], and switch control/scheduling for high-speed, high-capacity switches/routers.

The rest of the paper is organized as follows. In Section 2, we define the function of programmable k-selectors. In Section 3, we 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. In Section 4, we present four different programmable k-selector designs, each one is associated with a different next reference point generation circuit. In Section 5, we discuss and compare simulation results of different programmable k-selector designs on Synopsys's *design_analyzer* [15]. Section 6 concludes the paper.

2. Function of 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. x is named as the *reference point* which indicates the selection starting position.

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 $\mathbf{R} = (0, 1, 0, 1, 1, 0, 1, 1)$, we have $(Sum_0, Sum_1, \cdots, Sum_7) = (4, 5, 0, 1, 2, 2, 3, 4)$, and the selected requests are R_3, R_4 and R_6.

The block diagram of a general 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 programmable prefix sums circuit designs.

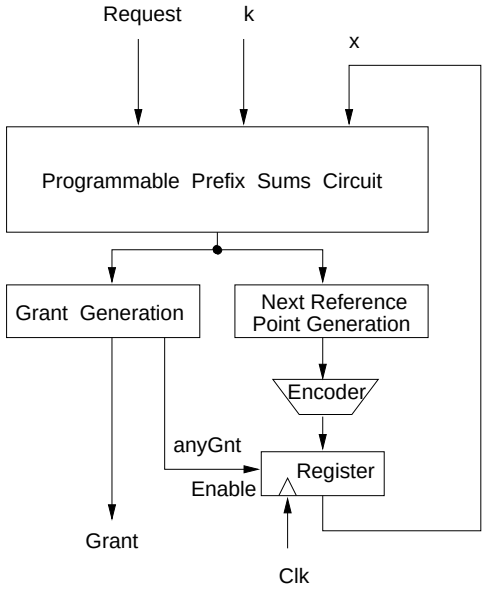

Fig. 3. The block diagram of a programmable k-selector.

3. Programmable Prefix Sums Circuit Designs

3.1 A Parallel Prefix Sums Circuit

We first discuss the design of a special case programmable prefix sums circuit 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 $0 \leq i \leq N-1$.

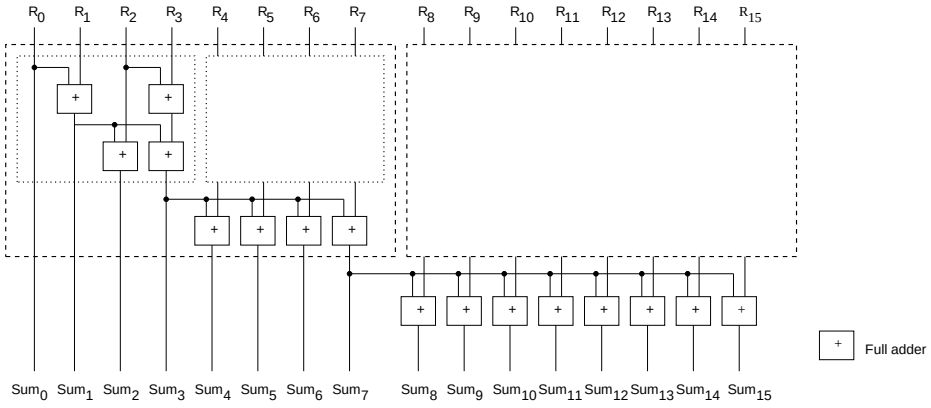

Fig. 4. A parallel prefix sums circuit using full adders.

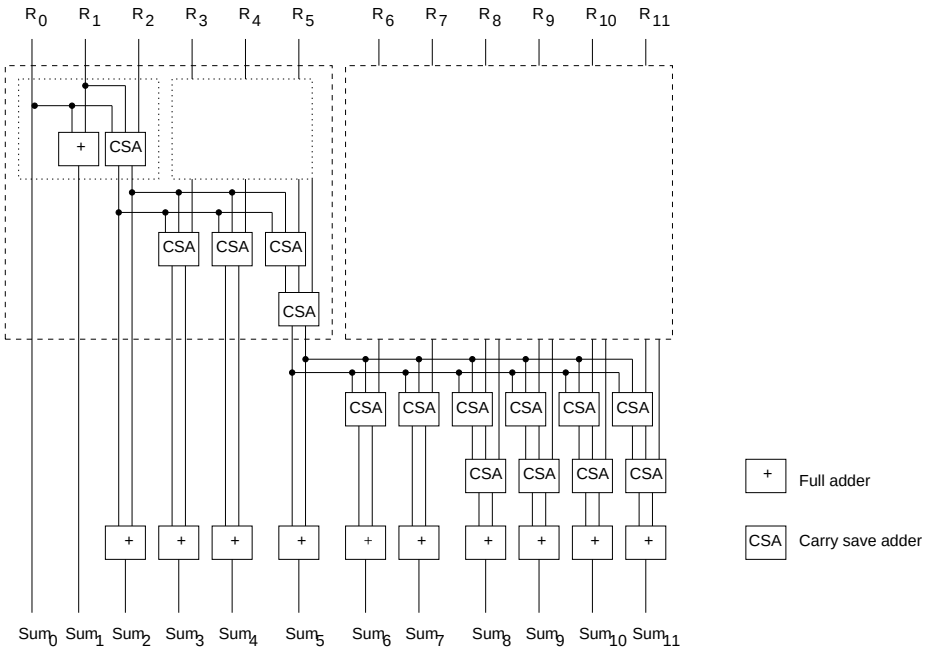

Fig. 5. A parallel prefix sums circuit using carry save adders and full adders.

We use adders to construct the parallel prefix sums circuit. Figure 4 displays the parallel prefix sum circuit for $N = 16$ using full adders. Dotted blocks depict the recursive construction scheme of this circuit. Due to ripple carries, such a design may not be the fastest one. Basically, there are two speedup methods. One is using carry-look-ahead technique, and the other is using carry save adders to form a well-known Wallace tree, as shown in Figure 5. Using carry save adders, the time complexity of this parallel prefix sums circuit is $O(\log N)$ gate 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 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 $0 \leq i \leq N-1$. Since x is a variable, this generalized problem is called *programmable prefix sums* (PPS) computation. In the following, we will present 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 an additional output $Sum = \sum_{i=0}^{N-1} R_i$, which will be used in the ROUNDROBIN_SELECT design.

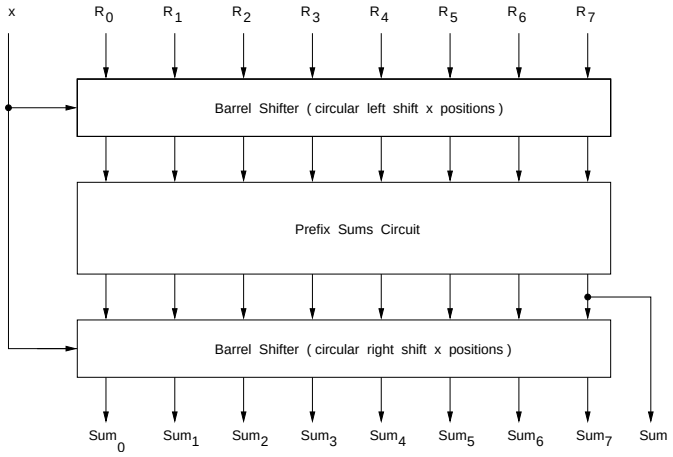

Fig. 6. A programmable prefix sums circuit using barrel shifters.

An N-bit barrel shifter can be implemented by $\log N$ stages of N 2×1 multiplexers with $O(\log N)$ gate delay [14]. It should be noticed that the second barrel shifter is more expensive because each Sum_i contains $\log N$ bits. When implementing a programmable k-selector, we can first use the comparators to obtain G_i's, as shown in Figure 11, before the grant signals are circular shifted right. This will significantly save circuitry of the second barrel shifter.

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 I

TRUTH TABLE AND LOGIC EQUATIONS FOR THE THERMOMETER LOGIC OF $N = 8$.

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. The two parts of prefix sums are merged into the final prefix sums.

We use an N-bit boolean vector **T** to separate the two parts. **T** is obtained by a thermometer encoding of a $(\log N)$-bit boolean vector **x** (the binary representation of x) with the following transformation equation:

$$T_i = \begin{cases} 0 & \text{if and only if } i < x, \\ 1 & \text{otherwise,} \end{cases}$$

where $0 \leq i \leq N - 1$. Table I shows the truth table and logic equations for the thermometer logic of $N = 8$. Given, a $(\log N)$-bit **x**, we use a thermometer encoding circuit to generate $T_0, T_1, \cdots, T_{N-1}$, which has $\log N$ gate delay. The sketch of a recursive algorithm for generating the boolean equations of T for $N = 2^{width}$, where $width \leq 0$ is shown in Figure 8. Thereafter, the N-bit thermometer encoder circuit can be constructed. Figure 7 shows the thermometer encoder circuit for $N = 8$. For example, if $x = 3$, then

$T_0T_1 \cdots T_7 = 00011111$.

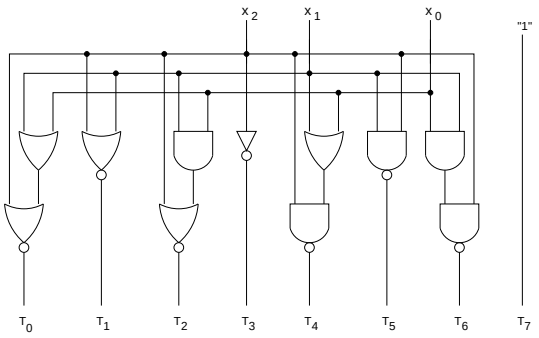

Fig. 7. A thermometer encoder.

```
width := log N;
power2 := 1;
T 0 := 0;
for i = 0 to (width - 1) do
    for j = 0 to (power2 - 1) do
       tmp := T j ;
        T j  := tmp OR x i ;
         T (j + power2) := tmp AND x i ;
    endfor
   power2 := power2 + power2;
endfor
for i = 0 to (N - 1) do
    T i  := NOT(T i );
endfor
```

Fig. 8. An algorithm for obtaining T for $N = 2^{width}$.

T is used to separate the requests into two parts at position x. The first part has $(R_0, R_1, \cdots, R_{x-1}, 0, \cdots, 0)$, which is extracted out by AND of **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 **R** and **T**. For each part, we use a SIMPLE_PS to compute prefix sums. Then we add the prefix sums of the second part to the prefix sums of the first part. The final prefix sums are obtained by merging the prefix sums of two parts. The circuit of DOUBLE_PPS for $N = 4$ is shown in Figure 9.

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

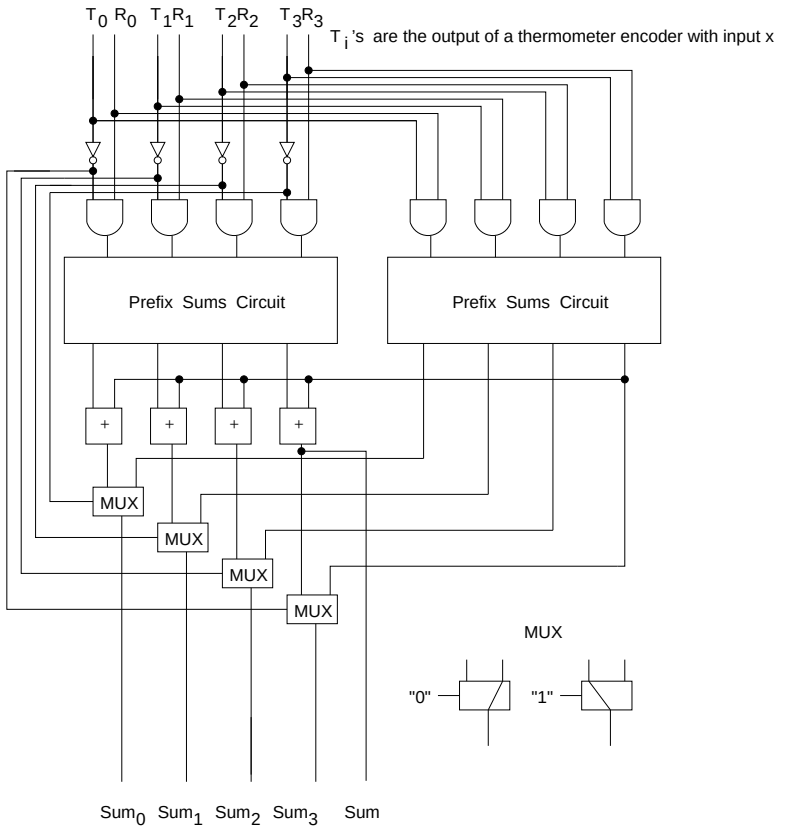

Fig. 9. The Double_PPS circuit for $N = 4$.

by removing the second SIMPLE_PS circuit? We present another design, named Design CONVERT_PS, which uses one SIMPLE_PS but does not require barrel shifters.

Let Sum'_i's, $0 \leq i \leq N-1$, be the prefix sums with $x = 0$. We observe that the programmable prefix sums Sum_i's with $0 \leq x \leq N-1$, 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, $Sum_i = Sum'_i + Sum_{N-1}$ for $0 \leq i \leq x-1$, and $Sum_i = Sum'_i - Sum'_{x-1}$ for $x \leq i \leq N-1$.

Based on this observation, we construct a circuit named PS_CONVERT, as shown in Figure 10. 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.

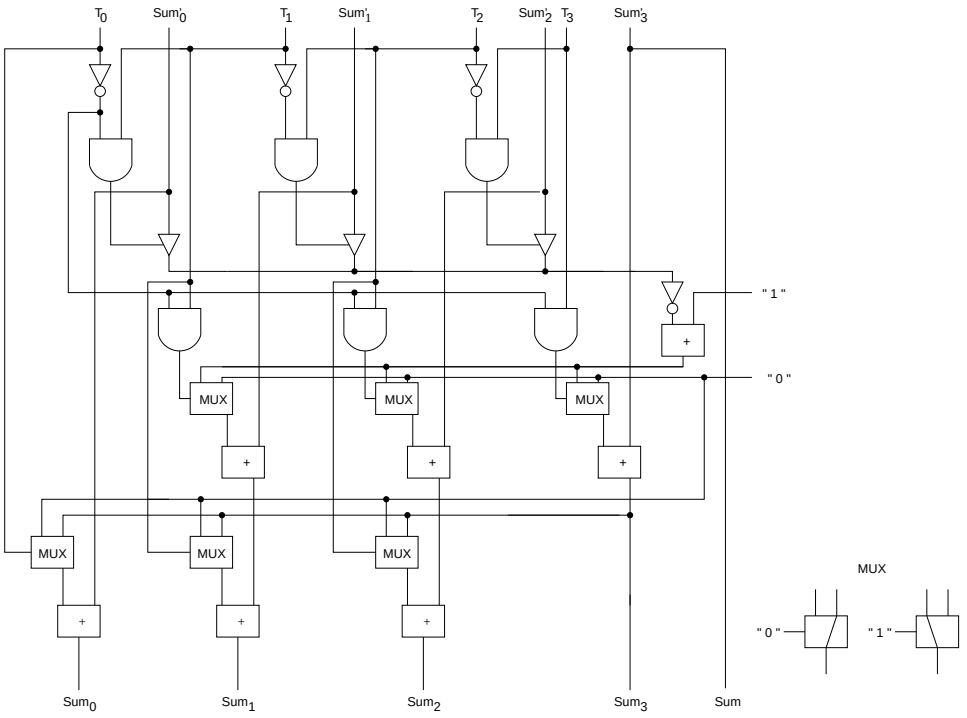

Fig. 10. The PS_CONVERT circuit for $N = 4$.

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.

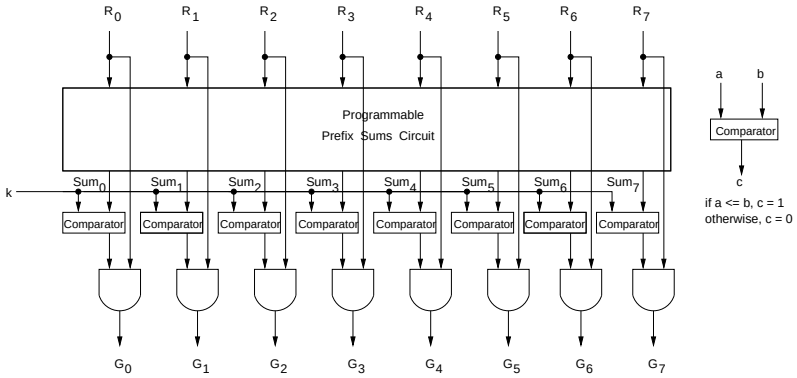

Fig. 11. Grant generation circuit for $N = 8$.

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 11 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 serve all requesters iteratively. In each iteration, $\min\{k, \sum_{i=0}^{N-1} R_i\}$ requests are selected. The next reference point (NRP) is determined after each iteration. An important issue in determining NRP is to avoid request starvation and ensure fairness to all requesters. 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 NRP generation circuits.

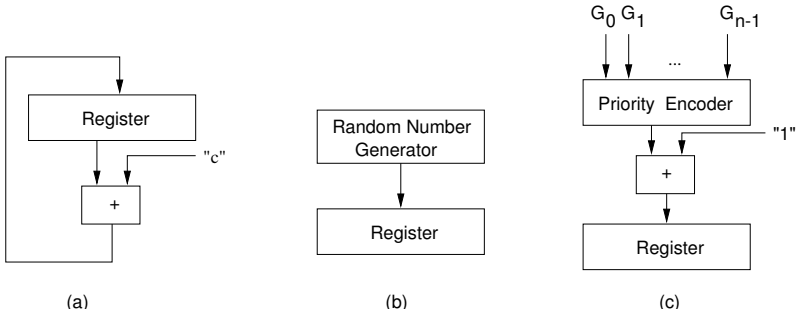

Fig. 12. Three NRP generation circuits.

- 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 12(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 12(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 $(G_0, G_1, G_2, G_3, G_4, G_5, G_6, G_7) = (0, 1, 0, 0, 1, 0, 1, 0)$, $current\_ref\_pt = 4$ and $next\_ref\_pt = 7$. The block diagram of this circuit is shown in Figure 12(c). We omit the circuit of a priority encoder here 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, 9, and 10. If $Sum \leq k$, $next\_ref\_pt = (j + 1) \bmod N$ such that $Sum_j = Sum$ and $R_j = 1$; otherwise, $next\_ref\_pt = (j + 1) \bmod N$ such that $Sum_j = k$ and $R_j = 1$. Figure 13 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 requesters.

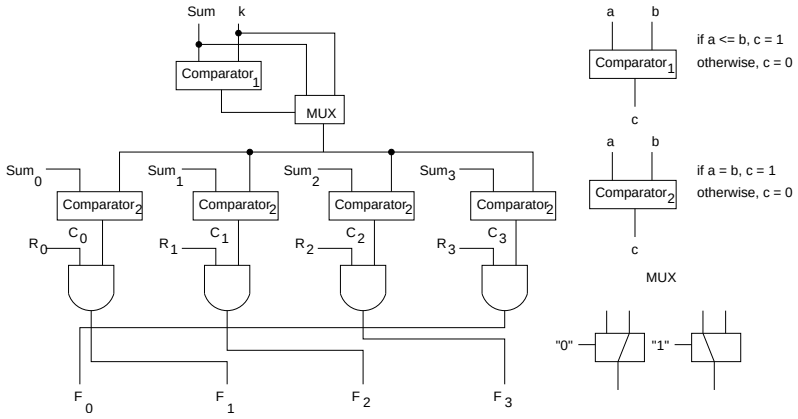

Fig. 13. Circuit for generating NRP in design ROUNDROBIN _SELECT.

5. Simulation Results and Comparisons

In this section, we present the simulation results of various designs of programmable k-selectors with Synopsys's *design_analyzer* using its library *lsi_10k* [15]. 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. We have written the Verilog HDL [16] codes for the design of PPE, Design SHIFT_PPS, Design DOUBLE_PPS, and CONVERT_PPS, and compiled them on

Design	**N=8**	**N=16**	**N=32**	**N=64**
PPE ($k = 1$)	8.91	12.41	17.58	25.93
SHIFT_PPS	30.39	39.80	59.40	92.18
DOUBLE_PPS	27.15	37.35	56.78	89.79
CONVERT_PPS	28.35	40.82	60.39	86.87
IMPROVEMENT	23.8%	62.5%	79.8%	89.2%

TABLE II

TIMING RESULTS FOR VARIOUS PROGRAMMABLE k-SELECTOR DESIGNS IN TERMS OF *ns*.

Design	**N=8**	**N=16**	**N=32**	**N=64**
PPE ($k = 1$)	192	468	1088	2457
SHIFT_PPS	413	1421	4816	16586
DOUBLE_PPS	597	1696	4474	14091
CONVERT_PPS	588	1752	4799	13970

TABLE III

AREA RESULTS FOR VARIOUS PROGRAMMABLE k-SELECTOR DESIGNS IN TERM OF THE NUMBER OF 2-INPUT NAND GATE.

the *design_analyzer*. Table II lists the timing results in terms of *ns* and Table III lists the area cost of these designs in terms of the number of 2-input NAND gates for $N = 8, 16, 32, 64$. Although not shown here, we have done the simulations for $N = 128$. However, due to the limitation of the library, some designs of $N = 128$ cannot fit completely. All these designs are optimized under the same operating conditions and the tool is directed to optimize area cost of each design.

The performance of our proposed three designs is independent of k, while the design of PPE (PPE_only_smpls) 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 DOUBLE_PPS only takes 89.79 *ns* to make 32 grants while the design of PPE takes at least $25.93 \times 32 = 829.76$ *ns* to make 32 grants. The bottom row of Table II shows the timing improvement percentage of DOUBLE_PPS over the design of PPE when $k = N/2$.

Among our three designs, Design DOUBLE_PPS achieves the best timing results with moderate area cost. The performances of Design SHIFT_PPS and Design CONVERT_PPS are comparable, while Design SHIFT_PPS consumes more area than Design CONVERT_PPS with N is increasing since the barrel shifter needs a large amount of interconnection resources. As N increases, the area cost of Design CONVERT_PPS tends to be the least one among the three designs. This is consistent with our analysis.

6. Concluding Remarks

In this paper, we defined programmable k-selectors for MRMS problems and proposed several programmable k-selector circuit designs. A programmable k-selector built with an integrated circuit (IC), such as FPGA or ASIC, is very attractive due to its high speed and its easy integration into the interface between requesters and servers. With the time complexity of $O(\log N)$, programmable k-selectors are very useful for switch control/scheduling in high-speed, high capacity IP switches/routers, such as constructing schedulers for SDMG CIOQ switches [2] and multi-server switches [11], [12], [13]. Programmable k-selectors are also useful for other real-time MRMS applications, such as the control of an $p \times q$ concentrator [17], [18], 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 and favoring the requests with higher priorities.

References

[1] S. T. Chuang, A. Goel, N. Mckeown, B. Prabhakar, Matching output queueing with a combined input output queued switch, *IEEE Journal on Selected Areas in Communications*, Vol. 17, No. 6, Jun. 1999, 1030-1039.

[2] M. Yang and S. Q. Zheng, An efficient scheduling algorithm for CIOQ switches with space division multiplexing expansion, to be presented on *IEEE Infocom2003*, San Francisco, Apr. 2003.

[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, 35(12)*, 1987, 1347-1356.

[4] D. N. Serpanos and P. I. Antoniadis, FIRM: A class of distributed scheduling algorithms for high-speed ATM switches with multiple input queues, *Proc. of IEEE Infocom2000*, 2000, 548-555.

[5] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, High-speed switch scheduling for local-area networks, *ACM Transactions on Computer Systems, 1(4)*, 1993, 319-352.

[6] N. McKeown, The *i*SLIP Scheduling Algorithm for Input-Queued Switches, *IEEE/ACM Transactions on Networking, 7(2)*, 1999, 188-201.

[7] J. Chao, Saturn: a terabit packet switch using dual round robin, *IEEE Communications Magazine, 38(12)*, 2000, 78-84.

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

[9] P. Gupta, N. McKeown, Designing and implementing a fast crossbar scheduler, *IEEE Microelectronics, 19(1)*, 20-29, 1999.

[10] 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.

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

[12] 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.

[13] M. Yang, S. Q. Zheng and D. Verchere, The *k*DDR scheduling algorithms for multi-server packet switches, to appear in *Proc. ISCA 15th International Conference on PDCS*, 2002, 78-83.

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

[15] Synopsys Design Analyzer Datasheet, available at http://www.synopsys.com/products/logic/deanalyzer_ds.html, 1997.

[16] IEEE Standards Board, *IEEE standard hardware description language based on the verilog hardware description language*, 1995.

[17] M. Garey, F. Hwang, and G. Richards, Asymptotic results for partial concentrators, *IEEE Transactions on Communications, 36(2)*, 1988, 214-217.

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