Modular Design of a Large Sorting Network

Neeraj K. Sharma
Applied Computing Research Institute
School of Computer Science & Computer Engineering
La Trobe University
Bundoora, Victoria 3083
Australia
neeraj@latcs1.cs.latrobe.edu.au

Abstract

Batcher sorting networks have been extensively used in the design of ATM switches based on Batcher-banyan interconnection network. Batcher sorting networks require large number of stages of sorting elements especially for large network sizes. This results in high delay, difficulty in partition into IC, and difficulty in maintaining synchronization across the entire structure. In this paper, we present a simple design of a sorting network that can be used as a building block to build larger sorting networks of arbitrary size. The proposed design is very modular and can be efficiently implemented using current VLSI technology.

Keywords: ATM switching, Batcher-banyan network, Batcher sorting network, shared memory ATM switch.

1. Introduction

Several broadband packet switches based on Batcher-banyan configuration have been proposed [1-5]. The Batcher-banyan network exploits the fact that a Banyan network is nonblocking if the active inputs are consecutive and the cells at these inputs are destined for outputs in an increasing or decreasing order [3]. An ascending sorter sorts the inputs by the ascending order whereas a descending sorter sorts the inputs by the descending order. The Batcher sorting network followed by a shuffle exchange and a Banyan network results in a nonblocking network, provided that no two inputs have cells destined for the same output port. Figure 1 shows an $N = 8$ Batcher-banyan network. An $N \times N$ Batcher sorting network requires $n(n+1)/2$ stages of 2×2 sorting elements, where $n = log_2N$ [6]. For large N, this results in a large number of stages of sorting elements which make partitioning into *IC* and maintaining synchronization across the entire structure difficult [1]. Furthermore, Batcher sorting networks can sort sequence of N inputs where N is a power of two. If N is not a power of two, part of the sorting network is not utilized.

In this paper, we present a simple design of a sorting network based on a fully shared memory [7-14]. In the proposed sorting network, N is not limited to a power of two. We then present design of a large sorting network. The large sorting network uses small sorting modules as building blocks for the larger sorting network.

The remaining of the paper is organized as follows. In Section 2, we review the Batcher sorting network. The proposed sorting network is discussed in Section 3 and the design of large sorting network using the proposed sorting network is discussed in Section 4. Section 5 concludes the paper.

2. Review of Batcher Sorting Network

A sorting network with N inputs and outputs consists of $n = log_2N$ levels of sub-sorters and is therefore referred to as n-level sorting network [6]. There are 2^{n-i} i-level sub-sorters and each i-level sorter is a banyan network with 2^i inputs and outputs, where $1 \leq i \leq n$ [15]. Thus, sorting networks are also known as banyan sorter. The sorting network has $n(n+1)/2$ stages of sorting elements and each stage has $N/2$ sorting elements as shown in Fig. 2. Each node of the sorting network is a 2×2 sorting element *(SE)* which sorts the incoming packets in the order as indicated by the arrows. The number of *SEs* in the sorting network of size N is $(n^2 + n)N/4$. The network operates synchronously with packets being processed in each stage in parallel. The sorting network sorts the packets according to the destination address of the packets.

3. Design of Proposed Sorting Network

The proposed $N \times N$ sorting network is shown in Fig. 3. It is a simplification of a fully shared memory ATM switch [7-14]. It consists of a $N{:}1$ multiplexer, serial to parallel *(S/P)* and parallel to serial *(P/S)* converters, ATM cell memory *(ACM)*, destination tag memory *(DTM)* for ATM cell memory management, a simple sorting network controller *(SNC)*, and a *1:N* demultiplexer. The function of the switch can be divided into two parts - write and read cycles. In the following discussion, we will assume the sorting network to be an ascending sorting network and descending sorting network will be discussed later. It is also assumed that the sorting network sorts the packets according to the destination address of the packets.

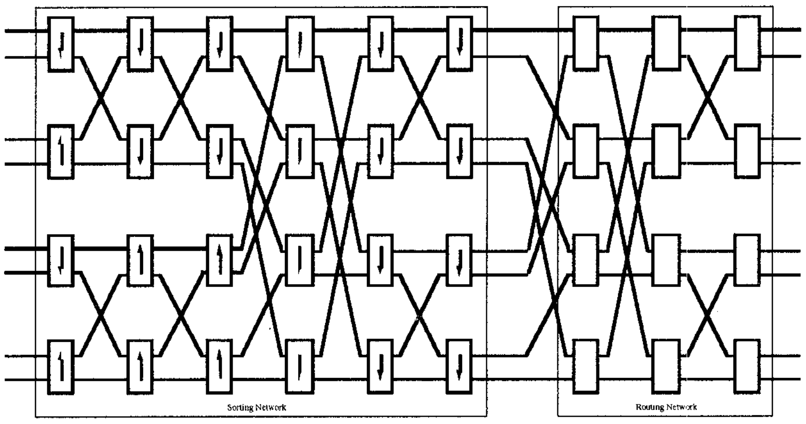

Figure 1. An N = 8 Batcher-Banyan Network

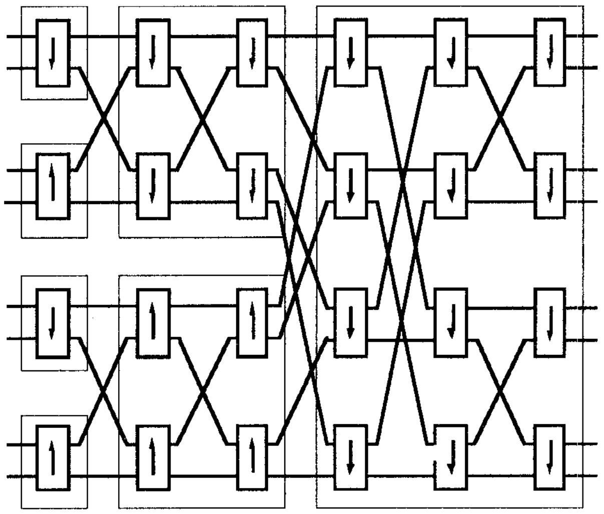

Figure 2. An N = 8 Ascending Batcher Sorter

Write Cycle: During the write cycle consisting of *N* sub-slots, the *N:1* multiplexer provides all the *N* inputs access to the ATM cell memory sequentially. After the serial to parallel conversion of the input cells, the cell from input *i* is written into the i^{th} location in *ACM* by the *SNC*, where $0 \leq i \leq N\text{-}1$. To perform this function, the *SNC* consists of an *modulo N* counter. Initially, the counter is set to *0*. For each sub-slot, the counter is incremented by one. The value of the counter provides the ATM cell memory with the write address *(Waddr)*. The capacity of the ATM cell memory is *53 bytes x N*. The *ACM* is logically organized as *N* link list, one for each output port. The link list is stored in the *DTM* which has a capacity of *2n+1 bits x N*. Both the *ACM* and *DTM* are maintained by the *SNC*. During each sub-slot, the link list in the *DTM* is updated by the *SNC*.

Read Cycle: During the read cycle consisting of another *N* sub-slots, in sub-slot $i, 0 \leq i \leq N\text{-}1$, the *SNC* examines the link list for output port *j*, where $0 \leq j \leq N\text{-}1$. If there are no cells destined for output port *j*, the *SNC* examines the link list for output *j+1*. In the case where there are *k* cells destined for output port *j*, the *SNC* reads each of the *k* cells sequentially in *k* sub-slots. The *SNC* provides the *ACM* with the read address *(Raddr)*. To perform this task, the *SNC* uses a *modulo N* counter. Initially, the counter is set to *0*. As soon as the *SNC* finds a cell to be read from the *ACM* using the link list, the output of the counter is used to set the *1:N* demultiplexer and the counter is incremented. Once the cell is read from ATM cell memory, the cell is sent to the *P/S* converter to be converted to a serial format and is sent to the output by the *1:N* demultiplexer. Note that during the read cycle, an update to the link list for each sub-slot is not required. After *N* read sub-slots, the link list is initialized to begin the next write cycle. The same counter used in the write cycle can be used for the read cycle. The function of the *SNC* during the read cycle is shown in Fig. 4.

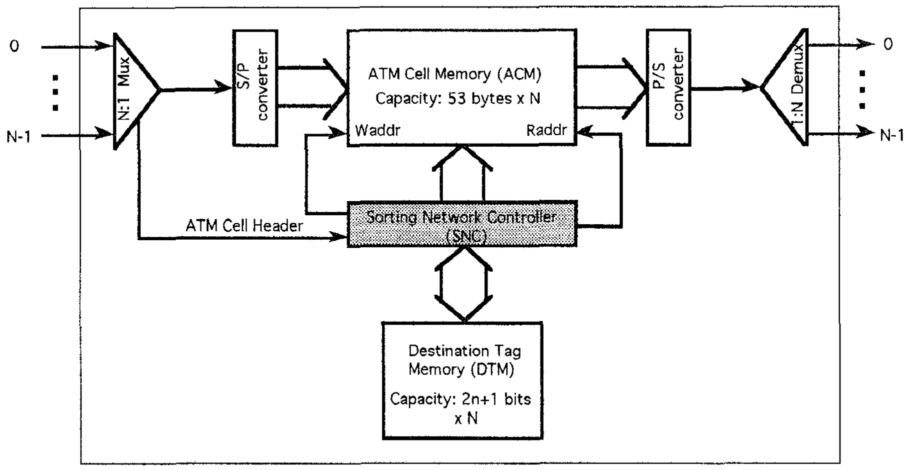

Figure 3. Proposed Sorting Network

```
sub_slot = 0;     /* initialize */
count = 0;     /* initialize */
while (sub_slot ≤ N-1)
  {
     if (there is at least one cell destined to output port 'count')
        while (there are cells destined to output port 'count')
           {
             send the cell to output number 'sub_slot' of the 1:N Demultiplexer;
             sub_slot++;
           }
     else
        count++;
  }
```

Figure 4. Function of the SNC During the Read Cycle

The *SNC* controls the *ACM, DTM, N:1 multiplexer*, and the *1:N demultiplexer*. During the write cycle, the ATM cell header is sent to the *SNC* and the *SNC* provides the *ACM* with the write address *(Waddr)*. For each write sub-slot, the link list stored in the *DTM* is updated by the *SNC*. For each cell stored in memory location *i* in the *ACM*, where $0 \leq i \leq N-1$, the destination address of the cell is stored in the i^{th} location of the *DTM*. For the i^{th} cell added in the link list, the *SNC* also determines the next location in the *ACM* where a cell is stored that is also destined to the same destination address as the cell stored in the i^{th} location of the *ACM*. An example of a link list stored in the *DTM* is shown in Table I. Note that in Table I, '*-1*' indicates the end of the link list, thus requiring *n+1* bits for storing the next location instead of *n* bits. The size of the DTM required is *2n+1 bits x N*.

For the read cycle, the *SNC* uses an output port lookup table as shown in Table II. At the end of the read cycle, the link list and the output port lookup tables are initialized for the next write cycle. The output lookup table requires *2n+2 bits x N* size storage.

The modification of the above mentioned ascending sorter to a descending sorter is straight forward. For the read cycle, instead of using an up-counter, we require a down-counter.

Table I. Link list stored in the DTM

Memory location	0	1	2	3	•	•	N-1
Destination number (n bits)	3	3	1	1	•	•	•
Next location (n+1bits)	1	-1	3	-1	•	•	•

Table II. Output port lookup table

Destination number (i)	ACM location for the first cell destined to i (n+1 bits)	ACM location for the last cell destined to i (n+1 bits)
0	-1	-1
1	2	3
2	-1	-1
3	2	2
•	•	•
•	•	•
N-1	•	•

Batcher sorters [6] can sort sequence of *N* inputs where *N* is a power of two. If *N* is not a power of two, part of the sorting network is not utilized. In the proposed sorting network, *N* can be any size. Thus, the proposed technique can be used to design arbitrary size sorting networks.

For the proposed sorting network, the ATM cell memory *(ACM)* must be able to perform *N* number of write and *N* number of read in every time slot. Thus, the size of the sorting network is limited by the speed of the *ACM*. For example, assuming the I/O line speed of *L* and a packet size of *424* bits, each sub-slot duration has to be less than *424 bits/(2 x N x L bps)*. Thus, if *L = 155 Mbps* and *N = 8*, the sub-slot duration has to be less than *171 ns*. For larger values of *L* and/or larger values of *N*, it may not be technologically feasible to implement a sorting network using the proposed technique. In the next section, we discuss the design of larger sorters using the proposed sorting network as a building block.

4. Design of Large Sorting Network

In this section, we propose the design of a large sorting network that is modular and can be easily implemented using VLSI technology. The proposed design uses the sorting network discussed in the previous section as modules in the Batcher sorting network reviewed in section 2 to build a larger sorting network.

To design a large *N x N* ascending sorting network, we will assume that *M x M* sorting modules are available which can sort a sequence of *M* inputs either in descending or ascending order as shown in Fig. 5. Its assumed that *M x M* sorting modules are designed as discussed in the previous section and *M* is selected such that the *M x M* sorting module can be easily implemented with the ..vailable technology. *M x M* sorting modules in Fig. 5 can also be represented as *d-dilated 2 x 2* sorting modules (with *d = M/2*) as shown in Fig. 6. Note that the function of the *d-dilated 2 x 2* sorting module is to sort sequence of *M* inputs either in descending or ascending order, where $M \geq 2$. When *M = 2 (d = 1)*, we obtain a *2 x 2* sorting element used in the Batcher-banyan network (Fig. 2).

The *d-dilated 2 x 2* sorting modules shown in Fig. 6 can now be used as *2 x 2* sorting elements in the Batcher sorting network in Fig. 2 to obtain larger sorting networks. All *2 x 2* sorting elements and links in the Batcher sorting network are replaced by *d-dilated 2 x 2* sorting modules and *d*-links, respectively, as shown in Fig. 7. The case where *d = 1* results in a Batcher sorting network. An example for *N = 16* sorting network with *M = 8* and *d = M/2 = 4* is shown in Fig. 8.

The proposed design of a large sorting network with N inputs and N outputs using *d-dilated* 2×2 sorting modules consists of $m = log_2(N/d)$ levels of subsorters. Like the Batcher sorting network, there are 2^{m-i} i-level sub-sorters and each i-level sorter is a *d-dilated* banyan network with $d \times 2^i$ inputs and outputs, where $1 \leq i \leq m$. The proposed large sorting network has $m(m+1)/2$ stages of *d-dilated* 2×2 sorting modules and each stage has N/M sorting modules which sort the incoming packets in the order indicated by the arrows. The total number of $M \times M$ sorting modules required to design an $N \times N$ sorting network is $m(m+1)N/(2 \times M)$. The network operates synchronously with packets being processed in each stage in parallel.

The proposed design of a large sorting network is very modular and can be easily implemented in VLSI. Each of the *d-dilated* 2×2 sorting modules can be implemented as an *IC* which can be programmed as an ascending or descending sorting module.

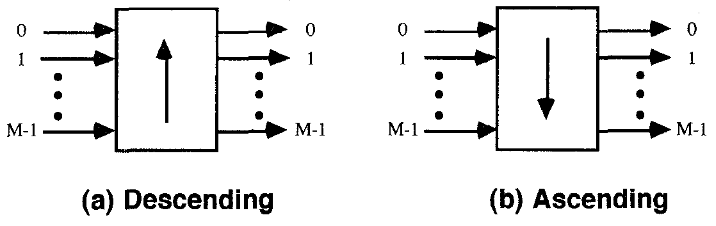

Figure 5. M x M Sorting Modules

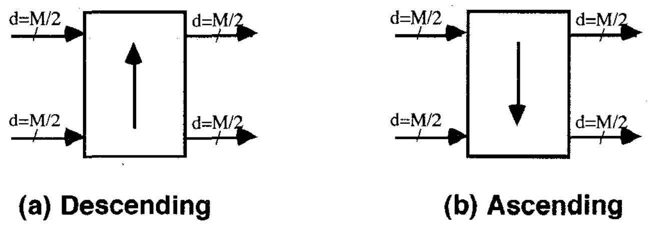

Figure 6. d-dilated 2 x 2 Sorting Modules

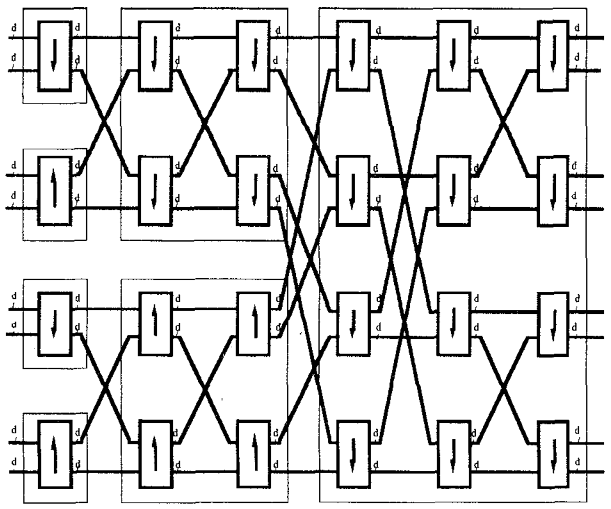

Figure 7. Proposed Design of a Large Sorting Network with 6 Stages of d-dilated 2 x 2 Sorting Modules

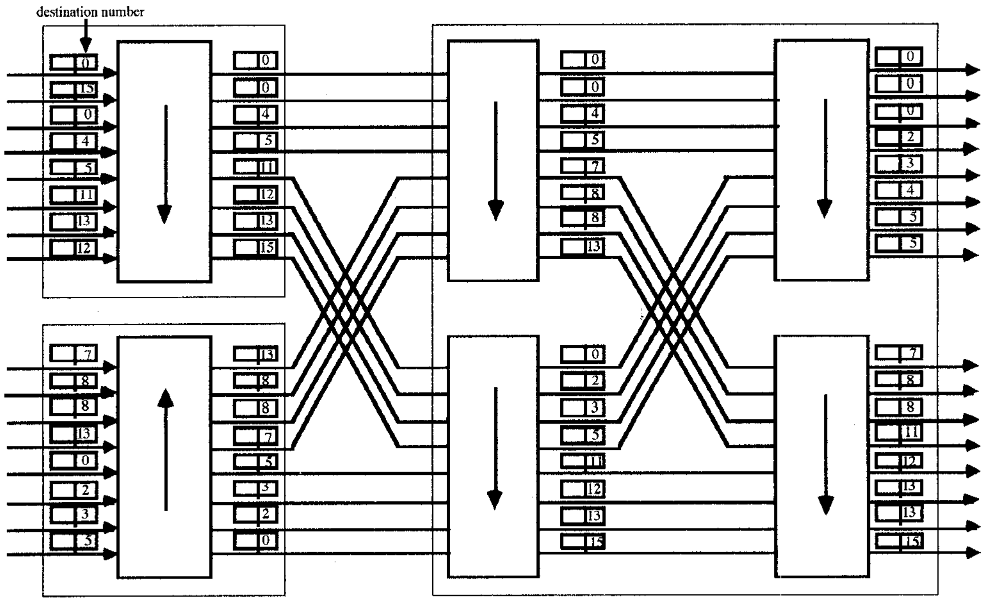

Figure 8. An N = 16 Sorting Network with M = 8 and d = 4

The delay of the proposed design of a large sorter is given by:

Delay of sorter = (delay of a M x M module)(number of stages of M x M modules)
= 2M(sub-slot period)m(m+1)/2

where $m = log_2(N/d)$. The number of stages is proportional to d as shown in Fig. 9. Note that $d = 1$ is the Batcher sorting network. It can be seen that large sorting networks can be designed using fewer stages and reasonable values of d.

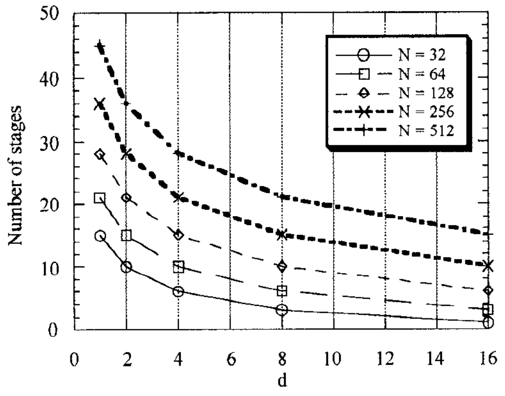

Figure 9. Number of Stages of d-dilated 2 x 2 Sorting Modules Required for Various N

5. Conclusions

In this paper, we first presented a simple design of a sorting network. Unlike the Batcher sorting network, the sorting network size is not limited to power of two. We then presented design of larger sorters. In a large Batcher sorting network, partitioning into *IC* and maintaining synchronization across the entire structure can be difficult. The proposed design overcomes these limitations. Since the proposed design of large sorter uses small sorting modules as basic building blocks, the design is very modular and can be efficiently implemented using VLSI technology.

REFERENCES

[1] R. Y. Awdeh and H. T. Mouftah, "Survey of ATM Switch Architectures," *Computer Networks and ISDN Systems*, 27, 1995, pp. 1567 - 1613.

[2] A. Pattavina, "Nonblocking Architectures for ATM Switching," *IEEE Communications Magazine*, February 1993, pp. 38 - 48.

[3] H. Ahmadi and W. E. Denzel, "A Survey of Modern High-Performance Switching Techniques," *IEEE Journal on Selected Areas in Communications*, vol. 7, No. 7, 1989, pp. 1091 - 1103.

[4] J. Hui and T. Lee, "A large Scale ATM Switching Network with Sort-Banyan Switch Modules," *GLOBECOM*, 1992, pp. 133 - 137.

[5] A. Huang and S. Knauer, " Starlite: A wideband digital switch," *GLOBECOM*, 1984, pp. 121 - 125.

[6] K. E. Batcher, “Sorting Networks and their Applications,” *AFIPS Conference*, 1968, pp. 307 - 314.

[7] J. Garcia-Haro and A. Jayszczyk, “ATM Shared-Memory Switching Architectures,” *IEEE Network*, July/August 1994, pp. 18 - 26.

[8] S. Kumar and D. P. Agrawal, “On Design of a Shared-Buffer based ATM Switch for Broadband-ISDN, *IEEE International Phoenix Conference on Computers Communications*, 1994, pp. 377 - 383.

[9] S. Kumar and D. P. Agrawal, “A Shared-Buffer Direct-Access (SBDA) Switch Architecture for ATM-based Networks,” *IEEE International Conference on Communications*, 1994, pp. 101 - 105.

[10] S. X. Wei and V. P. Kumar, “On the Multiple Shared Memory Module Approach to ATM Switching,” *IEEE INFOCOM*, 1992, pp. 116 - 123.

[11] K. Y. Eng, M. A. Pashan, R. A. Spanke, M. J. Karol and G. D. Martin, "A High-Performance Prototype 2.5 Gb/s ATM Switch For Broadband Applications," *IEEE Globecom*, 1992, pp. 111 - 117.

[12] N. Endo, T. Kozaki, T. Ohuchi, H. Kuwahara and S. Gohara, "Shared Buffer Memory Switch for an ATM Exchange," *IEEE Transactions on Communications*, Vol. 41, No. 1, January, 1993, pp. 237 - 245.

[13] H. Yamanaka, H. Saito, H. Yamada, M. Tsuzuki, S. Kohama, H. Ueda, H. Kondoh, Y. Matsuda and K. Oshima, "622 Mb/s 8x8 Shared Multibuffer ATM Switch with Hierarchical Queuing and Multicast Functions," *IEEE Globecom*, 1993, pp. 1488 - 1495.

[14] A. Jajszczyk and W. Kabacinski, “A Growable Shared-Buffer-Based ATM Switching Fabric,” *IEEE Globecom*, 1993, pp. 29 - 33.

[15] L. Goke and G. Lipsvski, “Banyan Networks for Partitioning Multiprocessors,” *1st Annual Symp. Computer Architecture*, 1973, pp. 21 - 28.