A Hardware Design Approach for Merge-Sorting Network

Chun-Yueh Huang
Department of Electronic Engineering
Kung Shan University of Technology

Gwo-Jeng Yu and Bin-Da Liu
Department of Electrical Engineering
National Cheng Kung University

ABSTRACT --In this paper, a hardware design methodology for merge-sorting networks, which uses a fixed size Batcher's sorting network, a data memory module and a memory addressing controller, is proposed. In this method, only by adjusting the data flow of the memory addressing controller, the amount of sorting data can be extended easily. Particularly, the adjustment of data flow is quite regular. Therefore, the proposed method has the following merits: low complexity of parallel sorting networks, low hardware fabrication cost, high extensibility, high regularity and no extra data memory space needed. For verifying the proposed approach, a 128-item merge-sorting network has been designed and simulated by Verilog VHDL.

INTRODUCTION

With many important applications in switching networks [1], image processing [2], and data processing [3], parallel sorting networks have received a lot of attentions from researchers. Recently, the research theme for parallel sorting networks [4, 5] trends to how to extend the number of sorting elements. However, if the number of sorting items increases, the complexity of parallel sorting networks will increase largely at the same time. As far as VLSI implementation is concerned, the cost of fabrication will raise as the number of sorting items increases. In order to avoid high complexity of parallel sorting networks, Olariu [6] proposed a methodology using a fixed size I/O sorting network and a set of memory module to extend the number of sorting elements. Basically Olariu's idea is promising in reducing the hardware cost, but the adjustment of data flow of addressing controller in Olariu's approach is irregular. That is to say, once we need to design some larger size sorting networks, these networks will become too difficult to extend in Olariu's approach.

In this paper, based on Olariu's idea, using a fixed size I/O sorting network and a set of memory module, we propose a modular and regular addressing method to design larger size sorting networks. In this method, the time complexity of comparing and exchanging data is $O(M\log N)$. To verify this approach, we use Verilog VHDL to design a 128-item merge-sorting network in this paper. Experimental results show that the proposed 128-item merge-sorting network can reduce both the complexity of parallel sorting networks and the cost of fabrication. Moreover, no extra data memory space is needed in this method.

BATCHER'S CLASSIC SORTING NETWORK

In General, a sorting network can be modeled by a directed graph whose nodes represent processing elements and whose edges represent the links between the nodes. A Batcher's sorting network with I/O size 8 in Fig. 1 is a typical and widely used example of sorting networks. In Fig. 1, the basic processing element, a comparator used to perform a compare-exchange operation, has two inputs and two outputs. After comparing input a with input b, the output at the top is min(a, b) and the output at the bottom is max(a, b). In accordance with the data flow of the sorting network, after the inputs (I_1, I_2, ..., I_8) pass through a series of comparisons and exchanges, the outputs (O_1, O_2, ..., O_8) will be obtained in ascending order. However, when the number N of data for sorting increases (i.e., N>20), it is too difficult to obtain a corresponding Batcher's sorting network because of high complexity.

MERGE-SORTING NETWORK

In order to reduce complexity of sorting networks, the merge-sorting approach can be used to solve the sorting problem with large number of data, combining a small size I/O Batcher's sorting network with a data memory module to sort plenty of data. Fig. 2 shows the framwork of the N-item merge-sorting network, which consists of three components: a memory module, a Batcher's sorting network with I/O size 8 as shown in Fig. 1, and a controller unite. The memory module, primarily for storing the sorting data, is arranged as $N/4$ rows and each row contains four 1-byte RAMs. The controller unite is used to generate memory address and control data flow in the sorting process.

At the outset of the merge-sorting, the original data for sorting are loaded into the memory module. And then, from the memory module, taking two rows of data to the 8-input sorter to be sorted each time, the sorter outputs will be generated in ascending order. Next, the outputs are further divided into two clusters--the larger one and the smaller one. At last, the data of the two clusters will be written back to the original memory module to replace the original data, according to the sorting sequence represented by the directed arrows as shown in the line

[1] This work was supported by the National Science Council, Republic of China, under Contract NSC -89-2215-E-006-043

0-7803-6685-9/01/$10.00©2001 IEEE

representation of Fig. 3. The direction of the arrows means ascending order while the data are being written back to the original memory module.

In the process of sorting sequence, an N-item data merge-sorting can be divided into three sorting cycles: (1) first local sorting cycle, (2) cross sorting cycle, and (3) second local sorting cycle, as shown in Fig. 3. In the first local sorting cycle, N-item data are divided into two parts, where each part contains $N/2$-item data to do merge-sorting respectively. When this process is finished, N-item data will be arranged in two $N/2$-item sorted clusters in ascending order. Next, in the cross sorting cycle, the smaller data in the up cluster are taken to be compared and exchanged with the larger data in the down cluster in sequence, and vice versa. After this sorting cycle is completed, the $N/2$-item data in the down cluster will be larger than the other $N/2$-item data in the upper cluster. In other words, now the data stored in the memory module are divided into the $N/2$-item larger parts in the down cluster and the $N/2$-item smaller parts in the up cluster. In the second local sorting, the data in the two cluster will be sorted separately again and stored in ascending order. Finally, the data stored in the memory module arranged in the row-major are the sorted results.

Now, we take a 16-item merge-sorting network for an example to demonstrate merge-sorting process in detail. The upper-left block of Fig. 3 is a 16-item merge-sorting. The sorting sequences are (1, 2), (3, 4), (1, 4), (2, 3), (1, 2), and (3, 4). When the first local sorting cycle (1, 2) and (3, 4) is finished, the 16-item data have been separated into two sorted clusters in ascending order. In the cross sorting cycle (1, 4) and (2, 3), the smaller data in the up cluster are taken to be compared with the larger data in the down cluster, and vice versa. After this step is completed, the 8-item data in the up cluster will be smaller than the other 8-item data in the down cluster. In the second local sorting cycle (1, 2) and (3, 4), the data in the two clusters will be sorted again and stored in ascending order. Finally, the 16-item data stored in the memory module arranged in the row major are the sorted results.

Similar to the above discussion, a 32-item merge-sorting network can be constructed as the line representation shown in Fig. 3. To extend further the above methodology to design a 128-item merge-sorting network, we need three components: a 128-byte memory module, a Batcher's sorting network with I/O size 8, and a controller unite. The memory module is arranged as 32 rows and each row contains four 1-byte RAMs, and the line representation of 128-item merge-sorting network as shown in Fig. 4.

The times of comparing and exchanging data for N-item merge-sorting networks can be formulated as

$$f(k) = 4 \times f(k/2) + k/2, \; k = N/4 \qquad (1)$$
$$f(2) = 1$$

where k is the number of row in the memory module. In the 128-item merge-sorting network, the times of comparing and exchanging data are $f(128/4) = f(32) = 496$. According to the proposed method, the times of comparing and exchanging data for a 1024-item merge-sorting network are $f(1024/4) = f(256) = 32640$. In this method, the time complexity of comparing and exchanging data is $O(N\log N)$.

HARDWARE IMPLEMENTATION

For the sake of real-time applications, we design a 128-item merge-sorting network in hardware based on Verilog. The circuit structure of the 128-item merge-sorting network includes 1) an 32-row memory module, where each row contains four 1-byte RAMs, 2) a set of data shift registers used to convert 4-byte data into 8-byte data, 3) an 8-input Batcher's sorting network, 4) an output multiplexer used to select the sorted results that will be written back to data memory, and 5) a controller unite used to generate memory address corresponding to sorting sequence, and the control signal for communicating with system processing units.

As for design methodology, we adopt Verilog VHDL to design the merge-sorting network in RTL level and synthesize by Synopsys tool based on 0.35μm CMOS process cell library. The total gate counts of the 128-item merge-sorting network are 3185. The delay time of the critical path for an 8-input sorting network is about 10ns, and the clock rate is 50MHz. It takes the data shift register and the output multiplexer four clock cycle times respectively to read in and write out the 2-row data between the memory and the sorting network. Every cycle time of comparing and exchanging 8-item data is 320ns. Simulation results show that the sorted results for 128-item data can be obtained in 158.72μs. A set of numerical simulation results for the 128-item merge-sorting network are shown in Table 1. Therefore, we prove that a complex 128-item sorting network can be implemented in hardware for real-time applications.

CONCLUSION

In this paper, a hardware design methodology for merge-sorting networks, which uses a fixed size Batcher's sorting network, a data memory module and a memory addressing controller, is proposed. In this method, only by adjusting the data flow of the memory addressing controller, the amount of sorting data can be extended easily. To further extend this approach for real-time applications, a 128-item merge-sorting network has been designed in hardware based on Verilog. Simulation results show that the sorted results for 128-item data can be obtained in 158.72μs. The proposed method has the following merits: low complexity of parallel sorting networks, low hardware fabrication cost, high extensibility, high regularity and no extra data memory space needed.

REFERENCES

[1] P. Zhou and O. W. W. Yang, "A new design of central queueing ATM switches," *IEEE Global Telecommunications Conf., GLOBECOM '97*, pp. 541 –545, vol.1, 1997

[2] C. C. Lin and C. J. Kuo, "Two dimensional rank-order filter by using max-min sorting network," *IEEE*

Trans. Circuit and Syst. for Vedio Technology, vol.8, no.8 , pp. 941–946, Dec. 1998.

[3] K. E. Batcher, "Sorting networks and their applications," *Proc. AFIPS Conf.*, pp.307-314, 1968.

[4] D. L. Lee and K. E. Batcher, "A multiway merge sorting network," *IEEE Trans. Parallel and Distributed Syst.*, vol.6, no.2 , pp. 211–215, Feb. 1995.

[5] T. Nakatani, S. T. Huang, B. W. Arden, and S. K. Tripathi, "K-way bitonic sort," *IEEE Trans. Computers*, vol.38, no.2 , pp.283–288, Feb. 1989.

[6] S. Olariu, M. C. Pinotti, and S. Q. Zheng, "How to sort N items using a sorting network of fixed I/O size," *IEEE Trans. Parallel and Distributed Syst.*, vol.10, no.5 , pp. 487–499, May 1999.

Table 1 Numerical simulation results for the 128-item merge-sorting network.

	DATA BEFORE SORTING				DATA AFTER SORTING			
ADDR	RAM0	RAM1	RAM2	RAM3	RAM0	RAM1	RAM2	RAM3
0	0	16	100	200	0	0	4	5
1	5	26	130	205	6	8	9	10
2	10	36	160	210	10	14	15	16
3	15	46	190	215	19	20	20	22
4	20	56	220	220	24	24	25	26
5	25	66	250	225	29	30	30	34
6	30	76	24	230	35	36	38	39
7	35	86	54	235	40	40	44	45
8	40	96	84	240	46	49	50	50
9	45	106	114	245	52	54	54	55
10	50	116	144	250	56	59	60	60
11	55	126	174	255	64	65	66	68
12	60	136	204	4	69	70	70	74
13	65	146	234	9	75	76	79	80
14	70	156	8	14	82	84	84	85
15	75	166	38	19	86	89	90	94
16	80	176	68	24	95	96	98	99
17	85	186	98	29	100	100	105	106
18	90	196	128	34	110	112	114	115
19	95	206	158	39	116	120	125	126
20	100	216	188	44	128	130	130	135
21	105	226	218	49	136	140	142	144
22	110	236	248	54	145	146	150	155
23	115	246	22	59	156	158	160	166
24	120	0	52	64	172	174	176	186
25	125	10	82	69	188	190	196	200
26	130	20	112	74	202	204	205	206
27	135	30	142	79	210	215	216	218
28	140	40	172	84	220	220	225	226
29	145	50	202	89	230	232	234	235
30	150	60	232	94	236	240	245	246
31	155	70	6	99	248	250	250	255

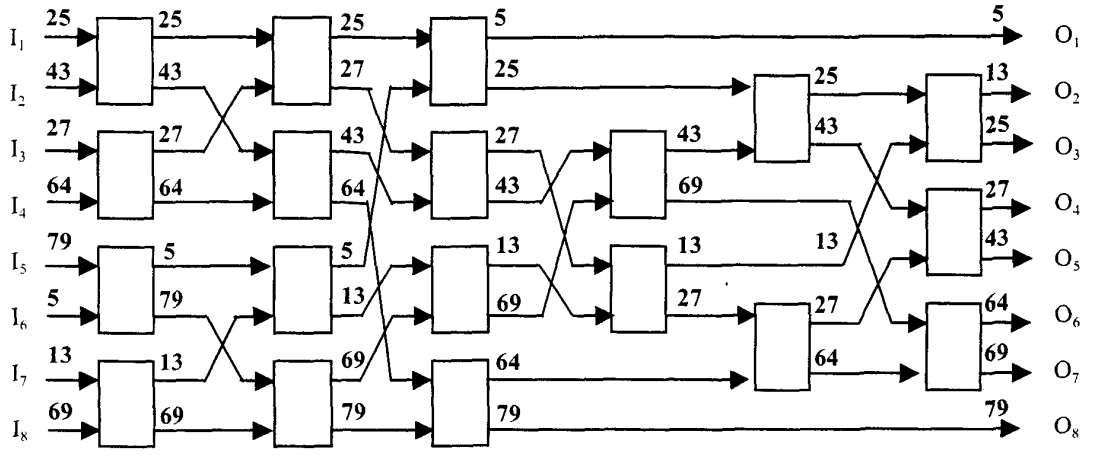

Fig. 1 A Batcher's classic sorting network of I/O size 8.

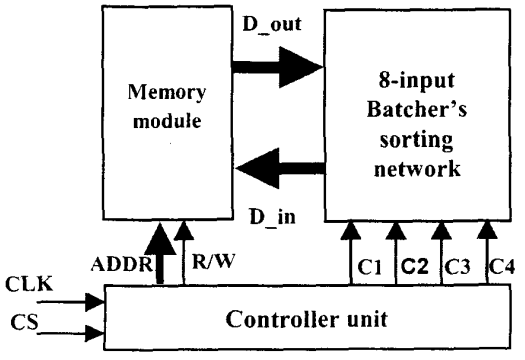

Fig. 2 The framwork of the merge-sorting network.

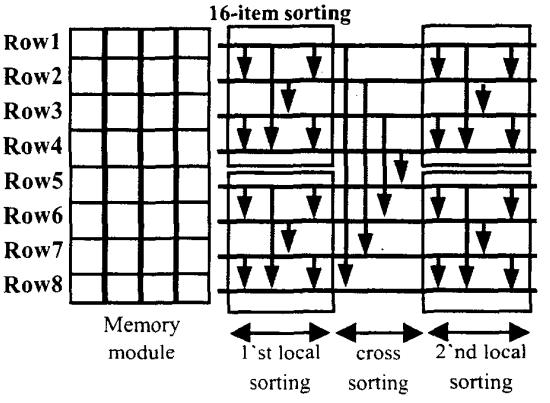

Fig. 3 The line representation of a 32-item merge-sorting network.

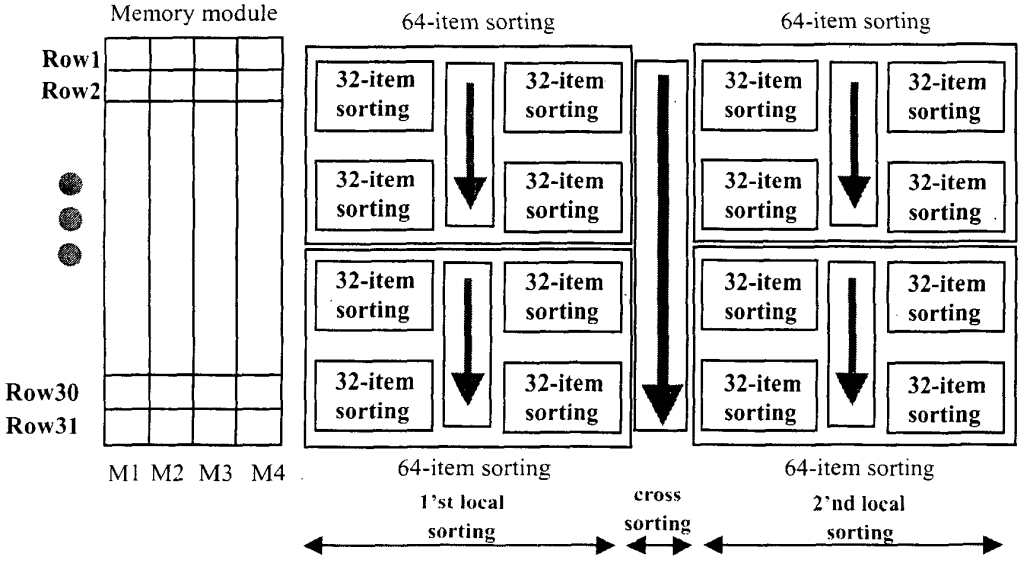

Fig.4 The addressing method representation of a 128-item merge-sorting network.